Microsoft

Inside OUT

The ultimate, in-depth reference

Hundreds of timesaving solutions

Supremely organized, packed with expert advice

Microsoft Exchange Server 2013: Mailbox and High Availability

Tony Redmond *Award-winning author + Microsoft MVP*

PUBLISHED BY
Microsoft Press
A Division of Microsoft Corporation
One Microsoft Way
Redmond, Washington 98052-6399

Copyright © 2013 by Tony Redmond

All rights reserved. No part of the contents of this book may be reproduced or transmitted in any form or by any means without the written permission of the publisher.

Library of Congress Control Number: 2013948703
ISBN: 978-0-7356-7858-3

Printed and bound in the United States of America.

First Printing

Microsoft Press books are available through booksellers and distributors worldwide. If you need support related to this book, email Microsoft Press Book Support at mspinput@microsoft.com. Please tell us what you think of this book at http://www.microsoft.com/learning/booksurvey.

Microsoft and the trademarks listed at http://www.microsoft.com/about/legal/en/us/IntellectualProperty/Trademarks/EN-US.aspx are trademarks of the Microsoft group of companies. All other marks are property of their respective owners.

The example companies, organizations, products, domain names, email addresses, logos, people, places, and events depicted herein are fictitious. No association with any real company, organization, product, domain name, email address, logo, person, place, or event is intended or should be inferred.

This book expresses the author's views and opinions. The information contained in this book is provided without any express, statutory, or implied warranties. Neither the authors, Microsoft Corporation, nor its resellers, or distributors will be held liable for any damages caused or alleged to be caused either directly or indirectly by this book.

Acquisitions Editor: Anne Hamilton
Developmental Editor: Karen Szall
Project Editor: Karen Szall
Editorial Production: nSight, Inc.
Technical Reviewer: Paul Robichaux; Technical Review services provided by Content Master, a member of CM Group, Ltd.
Copyeditor: Kerin Forsyth
Indexer: Lucie Haskins
Cover: Twist Creative • Seattle

Contents at a Glance

Chapter 1
Introducing Microsoft Exchange Server 2013 . . 1

Chapter 2
Installing Exchange 2013 43

Chapter 3
The Exchange Management Shell. 83

Chapter 4
Role-based access control 131

Chapter 5
Mailbox management. 169

Chapter 6
More about the Exchange Administration Center. 267

Chapter 7
Addressing Exchange 333

Chapter 8
The Exchange 2013 Store 387

Chapter 9
The Database Availability Group. 457

Chapter 10
Moving mailboxes 567

Chapter 11
Compliance management 641

Chapter 12
Public folders and site mailboxes 765

Table of Contents

Introduction . . . . . **xvii**
Acknowledgments xviii
Errata & book support xviii
We want to hear from you xix
Stay in touch xix

Chapter 1 **Introducing Microsoft Exchange Server 2013** **1**
Understanding development priorities 2
The influence of The Service 5
Exchange Online and Exchange development 7
Transition into the cloud 9
Exchange 2013 architecture 10
The motivation to upgrade 11
Evolving from earlier versions of Exchange 12
Waiting for updates 13
Fundamental questions before you upgrade 15
Selecting the right Windows Server for Exchange 2013 22
Using virtualization 24
Preparing for Exchange 2013 26
A word about transport 27
The test plan 28
Testing for operational processes 29
Testing for programming and customizations 30
Updating earlier versions of Exchange 31
Deploying earlier versions of Exchange servers alongside Exchange 2013 32
Exchange 2013 editions 32
Active Directory 34
Using the strong link between Exchange and Active Directory 34
ADSIEdit 36

What do you think of this book? We want to hear from you!

Microsoft is interested in hearing your feedback so we can continually improve our books and learning resources for you. To participate in a brief online survey, please visit:

microsoft.com/learning/booksurvey

Useful utilities . . . 37
MFCMAPI . . . 38
Exchange Web Services Editor . . . 40
Installing Exchange 2013 . . . 41

Chapter 2 **Installing Exchange 2013 . . . 43**

Approaching the installation . . . 43
Types of Active Directory deployment that support Exchange . . . 45
Preparing Active Directory for Exchange . . . 47
Creating the Exchange 2013 organization . . . 49
Deploying an Exchange 2013 server . . . 50
Installing UCMA . . . 53
Running Setup . . . 53
Setup logs . . . 56
Uninstalling Exchange . . . 58
Repairing Exchange . . . 61
Recovering a failed server . . . 61
Versions, cumulative updates, and service packs . . . 63
Cumulative updates . . . 64
Version numbers . . . 67
Object versions . . . 69
Reporting licenses . . . 70
Security groups and accounts Exchange creates . . . 73
Namespace planning . . . 77
Self-signed certificates . . . 77
Using the Exchange 2013 CAS to handle connections . . . 79
The case for protocol-specific namespaces . . . 80
Contemplating management . . . 81

Chapter 3 **The Exchange Management Shell . . . 83**

How Exchange uses Windows PowerShell . . . 83
Using remote Windows PowerShell . . . 86
Connecting to remote PowerShell . . . 88
Limiting user functionality . . . 91
EMS basics . . . 92
Command editing . . . 95
Handling information EMS returns . . . 98
Selective output . . . 98
Using common and user-defined variables . . . 101
Using PowerShell ISE with Exchange . . . 103
Identities . . . 105
Piping . . . 109
OPATH filters . . . 110
Server-side and client-side filters . . . 110
Transcripts . . . 114
Bulk updates . . . 114
Calling scripts . . . 117

Execution policies . . . 118
Profiles . . . 119
Active Directory for PowerShell . . . 120
Setting the right scope for objects in a multi-domain forest . . . 122
Exploring useful EMS examples . . . 123
Looking for large folders . . . 124
Outputting a CSV file . . . 125
Creating a report in HTML . . . 127
Verbose PowerShell . . . 129
Controlling access to Exchange . . . 129

Chapter 4 **Role-based access control . . . 131**

RBAC basics . . . 132
Roles . . . 136
Role groups . . . 139
Scopes . . . 141
Working with RBAC . . . 143
Creating a new role group . . . 145
Maintaining role group membership . . . 149
Role group management . . . 151
Role assignment . . . 152
Using role assignment policy to limit access . . . 153
Creating roles for specific tasks . . . 155
Specific scopes for role groups . . . 157
Database scoping . . . 158
Special roles . . . 159
Unscoped roles . . . 160
Which role groups do I belong to? . . . 162
Assignment policies . . . 163
Implementing a split permissions model . . . 166
Figuring out RBAC . . . 167
On to management . . . 168

Chapter 5 **Mailbox management . . . 169**

Seeking perfection halts progress . . . 170
Starting EAC . . . 173
How EAC accesses Exchange data . . . 176
Changing EAC columns . . . 177
Naming conventions . . . 178
Managing Recipients . . . 180
Recipient filtering . . . 180
Exporting EAC information to CSV files . . . 181
Some mysterious mailboxes . . . 182
The need for mailboxes . . . 183
Naming mailboxes . . . 185
Creating new mailboxes . . . 187
Completing mailboxes . . . 193

Bulk mailbox creation . . . 195
Creating new room and resource mailboxes . . . 195
Languages . . . 196
Default folders . . . 200
Manipulating mailbox settings . . . 202
Automating mailbox settings . . . 205
What's in a mailbox? . . . 207
Ready-to-go custom attributes . . . 207
Mailbox resources provisioning management agent . . . 209
Setting mailbox quotas . . . 213
Removing or disabling mailboxes . . . 218
Reconnecting mailboxes . . . 220
Discovery mailboxes . . . 224
Creating additional discovery mailboxes . . . 224
Health mailboxes . . . 226
Setting mailbox permissions . . . 229
Mailbox delegation . . . 229
Managing Full Access permission . . . 232
Mailbox auto-mapping through Autodiscover . . . 235
Opening another user's mailbox with Outlook Web App . . . 237
Sending messages on behalf of other users . . . 237
Outlook delegate access . . . 239
Shared mailboxes . . . 240
Recalling messages . . . 241
Moderated recipients . . . 243
Moderated groups . . . 244
Processing moderation requests . . . 247
Moderated mailboxes . . . 249
Mail-enabled contacts . . . 250
Mail users . . . 252
Resource mailboxes . . . 254
Defining custom properties for resource mailboxes . . . 255
Providing policy direction to the Resource Booking Attendant . . . 256
Processing meeting requests according to policy . . . 261
Equipment mailboxes . . . 265
Enough about mailboxes . . . 266

Chapter 6 **More about the Exchange Administration Center . . . 267**

Groups . . . 267
Managing groups . . . 269
Creating new groups . . . 270
Group owners . . . 275
Group expansion . . . 277
Protected groups . . . 278
Viewing group members . . . 281
Security groups . . . 284
Tracking group usage . . . 285

Allowing users to manage groups . . . 286
Room lists . . . 298
Dynamic distribution groups . . . 300
OPATH queries . . . 300
Creating new dynamic distribution groups . . . 302
Validating query results . . . 305
Creating dynamic groups with EMS . . . 306
Using custom filters . . . 308
Sharing . . . 313
Certificate management . . . 313
Mail flow and rules . . . 318
Delivery reports . . . 320
Administrator searches for delivery reports . . . 322
Using EMS to search delivery reports . . . 327
Running EAC without an Exchange mailbox . . . 329
Setting diagnostics for Exchange servers . . . 330
But what will you manage? . . . 332

Chapter 7 **Addressing Exchange . . . 333**

Email address policies . . . 333
Email policy priority . . . 334
Creating a new email address policy . . . 337
Focusing on certain recipients by using filters . . . 341
Creating email address policies with custom filters . . . 343
Address lists . . . 345
Address book policies . . . 348
ABPs and groups . . . 350
Creating and implementing an ABP . . . 351
The Offline Address Book . . . 357
The OAB and Outlook . . . 359
How Exchange 2013 generates the Offline Address Book . . . 362
Creating and using customized OABs . . . 367
Hierarchical address book . . . 371
MailTips and group metrics . . . 373
Client interaction . . . 376
Configuring MailTips . . . 377
User experience . . . 378
Custom MailTips . . . 380
Multilingual custom MailTips . . . 381
OAB support for MailTips . . . 381
The display or Details Templates Editor . . . 382
The next step . . . 386

Chapter 8 **The Exchange 2013 Store . . . 387**

Long live JET! . . . 388
Maximum database size . . . 389
Sizing mailboxes . . . 391

Dealing with I/O . . . 392
Maintaining contiguity . . . 397
The database schema . . . 398
Exchange 2013 I/O improvements . . . 400
Workers, controller, and memory . . . 402
Managed Availability and the Managed Store . . . 407
Database management . . . 407
Creating new mailbox databases . . . 410
Updating mailbox databases after installation . . . 414
Backups and permanent removal . . . 416
Removing a database . . . 417
Transaction logs . . . 419
Log sets . . . 419
Transaction log checksum . . . 427
Transaction log I/O . . . 427
The question of circular logging . . . 429
Reserved logs . . . 431
Background maintenance . . . 432
Database checksums . . . 433
Database defragmentation . . . 435
Database compaction . . . 436
Page zeroing . . . 436
Content maintenance tasks . . . 436
Corrupt item detection and isolation . . . 437
Protection against high latency . . . 442
Protection against excessive database or log growth . . . 443
Debugging swelling databases . . . 444
Online repair cmdlets . . . 445
Rebuilding a database . . . 448
Using ESEUTIL . . . 450
Database usage statistics . . . 451
Mailbox assistants . . . 454
And now for something completely different . . . 455

Chapter 9 **The Database Availability Group . . . 457**

Changes in high availability in Exchange 2013 . . . 457
Database portability . . . 459
Database Availability Group basics . . . 461
The dependency on Windows clustering . . . 463
Migrating an Exchange 2010 DAG . . . 464
Active Manager . . . 465
Automatic database transitions . . . 467
Managed Availability and high availability . . . 469
Best copy and server selection . . . 472
ACLL: Attempt copy last logs . . . 478
Transaction log replay: The foundation for DAG replication . . . 479
Transaction log compression . . . 483

Block mode replication . . . 484
Transaction log truncation . . . 486
Incremental resynchronization . . . 487
Seeding a database . . . 489
Changes in message submission within a DAG . . . 489
Day-to-day DAG management and operations . . . 489
Building the DAG . . . 494
Pre-staging the Cluster Name Object . . . 497
The role of the FSW . . . 499
DAG task logging . . . 502
Crimson events . . . 502
Managing DAG properties . . . 504
DAG networks . . . 506
Cleaning up before creating database copies . . . 510
Using circular logging inside a DAG . . . 510
Adding new database copies to a DAG . . . 514
Monitoring database copies . . . 517
Reseeding a database copy . . . 523
Adding database copies with EMS . . . 525
Using a lagged database copy . . . 525
Activating a mailbox database copy . . . 533
Rebalancing database copies . . . 534
Performing a server switchover . . . 536
AutoDatabaseMountDial and potential issues moving databases . . . 539
Activation blocks . . . 542
Moving database locations within a DAG . . . 544
Removing database copies . . . 546
Removing servers from a DAG . . . 548
Handling storage failures . . . 549
Datacenter Activation Coordination . . . 551
Approaching DAG designs . . . 552
How Exchange 2013 changes the resilience equation . . . 558
Building a DAG with Exchange Standard edition . . . 560
Server maintenance . . . 562
Stressed servers . . . 565
On to protecting data . . . 566

Chapter 10 **Moving mailboxes . . . 567**

The Mailbox Replication service . . . 567
Moving mailboxes . . . 568
Asynchronous moving . . . 571
The Migration service . . . 572
Mailbox Replication service processing . . . 575
MRS and system resources . . . 579
Preventing loss of data . . . 580
Creating and managing migration batches with EAC . . . 581
Using EMS with migration batches . . . 592

Reporting mailbox moves . . . 595
Getting more information about a move . . . 597
Accessing move report histories . . . 599
Managing individual mailbox moves . . . 605
Assigning move priority . . . 607
Moves and mailbox provisioning . . . 607
Handling move request errors . . . 607
Checking and altering move request status . . . 609
Clearing move requests . . . 610
Migration endpoints . . . 611
Enabling MRSProxy . . . 612
Planning mailbox moves . . . 614
Suspending mailbox moves . . . 615
Ensuring high availability . . . 619
MRS configuration file . . . 621
Mailbox import and export . . . 623
Gaining permission through RBAC to execute mailbox import and export . . . 624
Planning the import of PST data . . . 626
Importing and exporting mailbox data with EAC . . . 629
Importing and exporting mailbox data with EMS . . . 632
Alerts . . . 638
Time to comply . . . 640

Chapter 11 **Compliance management . . . 641**

The joy of legal discovery . . . 642
Archive mailboxes . . . 645
Enabling archives . . . 647
The default archive and retention policy . . . 652
Using an archive mailbox . . . 655
Disabling an archive mailbox . . . 656
Messaging records management . . . 657
Types of retention tags . . . 657
System tags . . . 662
Designing a retention policy . . . 663
Managed Folder Assistant and retention policies . . . 665
Naming retention tags . . . 666
Creating retention tags . . . 668
Creating a retention policy . . . 673
Applying a retention policy to mailboxes . . . 677
Modifying a retention policy . . . 681
Customizing retention policies for specific mailboxes . . . 681
User interaction with retention policies . . . 684
Setting a retention policy on a folder . . . 685
Removing tags from policies . . . 687
Removing a retention policy . . . 688
Upgrading from managed folders . . . 688

How the Managed Folder Assistant implements retention policies . . . 689
Behind the scenes with the MFA . . . 691
Retention date calculation . . . 693
Preserving information . . . 696
Putting a mailbox on retention hold . . . 698
Putting a mailbox on litigation hold . . . 699
Searching mailbox content . . . 701
In-place holds . . . 703
Creating a new search . . . 708
Retrieving discovered content . . . 716
Examining search results . . . 720
Resource throttling for searches . . . 724
How in-place holds work . . . 726
Using groups with searches . . . 728
Removing a search . . . 730
Executing searches with EMS . . . 731
What Exchange can search . . . 733
Search syntaxes . . . 736
The value of the Recoverable Items structure . . . 737
The function of the Recoverable Items structure . . . 737
Improvements in Exchange 2013 . . . 739
Single-item recovery . . . 743
Knowing what's in Recoverable Items . . . 745
Managing quotas for Recoverable Items . . . 746
Auditing administrator actions . . . 748
The audit mailbox . . . 749
How administrator auditing happens . . . 750
Auditing mailbox access . . . 754
Enabling mailboxes for auditing . . . 757
Accessing mailbox audit data . . . 759
Other compliance features . . . 763

Chapter 12 **Public folders and site mailboxes . . . 765**

Public folders . . . 765
New concepts . . . 766
Creating public folder mailboxes . . . 767
How many public folder mailboxes are needed? . . . 770
Controlling the root . . . 774
Creating public folders . . . 776
Mail-enabling public folders . . . 779
Moving public folders . . . 783
Redirecting content . . . 785
Removing a public folder . . . 786
Mobile access . . . 786
Organizational forms library . . . 787
Migration to modern public folders . . . 787

Site mailboxes 791
How site mailboxes work 792
The life cycle of site mailboxes 800
Site mailbox provisioning policy 803
Summarizing public folders and site mailboxes 804
Moving on 805

Index 807

What do you think of this book? We want to hear from you!

Microsoft is interested in hearing your feedback so we can continually improve our books and learning resources for you. To participate in a brief online survey, please visit:

microsoft.com/learning/booksurvey

Foreword for Exchange 2013 Inside Out books

Those seeking an in-depth tour of Exchange Server 2013 couldn't ask for better guides than Tony Redmond and Paul Robichaux. Tony and Paul have a relationship with the Exchange team that goes back two decades, to the days of Exchange 4.0. Few people have as much practical knowledge about Exchange, and even fewer have the teaching skills to match. You are in good hands.

Over the past few years, we have seen significant changes in the way people communicate; a growing number of devices, an explosion of information, increasingly complex compliance requirements, and a multigenerational workforce. This world of communication challenges has been accompanied by a shift toward cloud services. As we designed Exchange 2013, the Exchange team worked hard to build a product and service that address these challenges. As you read these books, you'll get an up-close look at the outcome of our efforts.

Microsoft Exchange Server 2013 Inside Out: Mailbox and High Availability covers foundational topics such as the Exchange Store, role-based access control (RBAC), our simplified approach to high availability, and the new public folder architecture. It also covers our investments in eDiscovery and in-place hold. As you read, you'll see how Exchange 2013 helps you achieve world-class reliability and provides a way to comply with internal and regulatory compliance requirements without the need for third-party products.

Microsoft Exchange Server 2013 Inside Out: Connectivity, Clients, and UM explores the technologies that give users anywhere access to their email, calendar, and contacts across multiple devices. It also explains how to protect your email environment from spam, viruses, and other threats and describes how Exchange 2013 can connect with Office 365 so you can take advantage of the power of the cloud.

From our new building-block architecture to data loss prevention, there's a lot to explore in the newest version of Exchange. I hope that as you deploy and use Exchange 2013, you'll agree that this is an exciting and innovative release.

Enjoy!

Rajesh Jha
Corporate Vice President - Exchange
Microsoft Corporation

Introduction

This book is for experienced Exchange administrators who want to get inside the soul of Exchange Server 2013, the latest version of the Microsoft enterprise messaging server first released in October 2012 and updated on a frequent basis since. You might learn how to work with Exchange 2013 by reading this book, but I sincerely doubt that this will happen simply because I have written it with experience in mind.

The book does not cover every possible topic relating to Exchange 2013. In fact, it focuses primarily on the Mailbox server role. Let me explain why. After completing *Microsoft Exchange Server 2010 Inside Out* (Microsoft Press, 2010), it became very clear that attempting to cover all of a complex product such as Exchange in any depth in just one book was a fool's errand. There are too many details to master, too much work to do, too much information that can only be skimmed over to keep to a reasonable page count. The result would probably be a book that weighs 2 kilos, spanning 1,400 pages that takes 2 years to write. All in all, an unacceptable situation in both commercial and practical terms.

Paul Robichaux and I ran a number of Exchange 2010 Maestro seminars in the 2010–2011 period. Despite the infamous cockroach sandwich affair, the events were good fun, and we enjoyed discussing the technology in some depth, even if we tended to ramble on at times. Brian Desmond, an Active Directory MVP who did an excellent job of lab master and stand-in speaker when required, helped us. Because we worked well together and because Paul has an excellent record of writing both books and articles, it seemed like a good idea to consider a joint approach for *Microsoft Exchange Server 2013 Inside Out*. We arrived at the basic idea quickly—we would split coverage into the two server roles. I'd write about the Mailbox role and Paul took on client access, including all the various clients Exchange supports, and unified messaging, which, strictly speaking, is part of an Exchange 2013 Mailbox server. However, Paul is an acknowledged expert in this space, and it would have made no sense to have me write about a subject of which Paul is the master.

Because Exchange 2013 is an evolution of Exchange 2010, we decided to use *Microsoft Exchange Server 2010 Inside Out* as the base for the new book. An evolution it might be, but an extensive level of change at the detail level exists in Exchange 2013. The upshot is that I'm not sure how much of that book remains in the current text—maybe 20 percent. One thing I am glad of is that we did not rush to press after Exchange 2013 first appeared. Given the amount of change that has occurred in updates from Microsoft since, a book that describes the release to manufacturing (RTM) version of Exchange 2013 would have been obsolete very soon after publication. We hope that these volumes will last longer.

I hope that you enjoy this book and that you'll read it alongside Paul's *Microsoft Exchange Server 2013 Inside Out: Clients, Connectivity, and UM*. The two books really do go together. Paul has scrutinized every word in this book and I have done the same for his. We therefore share the blame for any error you might find.

Acknowledgments

I owe enormous thanks to the many people who agreed to look over chapters or portions of the book. Each has deep expertise in specific areas and all contributed greatly to eradicating errors and increasing clarity. These folks include Sanjay Ramaswamy, Jürgen Hasslauer, David Espinoza, William Rall, Todd Luttinen, Tim McMichael, Vineetha Kalvakunta, Fred Monteiro da Cruz Filho, Kanika Ramji, Lokesh Bhoobalan, Astrid McClean, Alfons Staerk, Kern Hardman, Andrew Friedman, Abram Jackson, and Scott Schnoll. Even if they didn't realize it, many of the Exchange MVPs played their part in improving the book by prompting me to look into topics that I had forgotten to cover. I should also acknowledge the huge contribution made by my editor, Karen Szall. We fought many times about page counts, content, and too many other topics to list here but always kept the project moving.

I apologize sincerely if I have omitted to mention anyone who has contributed to making the text of the book as accurate and as informative as possible.

Errata & book support

We've made every effort to ensure the accuracy of this book and its companion content. Any errors that have been reported since this book was published are listed on our Microsoft Press site at oreilly.com:

http://aka.ms/ExIOv1/errata

If you find an error that is not already listed, you can report it to us through the same page.

If you need additional support, email Microsoft Press Book Support at *mspinput@microsoft.com*.

Please note that product support for Microsoft software is not offered through the addresses above.

We want to hear from you

At Microsoft Press, your satisfaction is our top priority, and your feedback our most valuable asset. Please tell us what you think of this book at:

http://www.microsoft.com/learning/booksurvey

The survey is short, and we read every one of your comments and ideas. Thanks in advance for your input!

Stay in touch

Let's keep the conversation going! We're on Twitter: *http://twitter.com/MicrosoftPress*.

CHAPTER 1

Introducing Microsoft Exchange Server 2013

Understanding development priorities. 2
The influence of The Service. 5
Exchange 2013 architecture 10
The motivation to upgrade. 11
Selecting the right Windows Server for Exchange 2013 . 22
Using virtualization 24
Preparing for Exchange 2013 26
Exchange 2013 editions. 32
Active Directory 34
Useful utilities 37
Installing Exchange 2013 41

Many users have worked with email for a long time now, spanning development from its "green screen," beginning when everyone used video terminals connected to a central computer to compose and send messages that were most definitely text-only, to when many companies quite happily use cloud-based email, meaning that they connect to central computers over the Internet. Although the clients are very different and the contents of the messages are much more varied, the rush to embrace highly centralized services almost seems like a return to the mainframe-dominated past.

As good as cloud-based services are, the need still exists for on-premises deployments, where servers, clients, and all the supporting infrastructure are firmly under the control of an IT department. A new version of a popular software product such as Microsoft Exchange Server 2013 generates different emotions for the different players who participate in the cycle of product development, deployment, operations, and support, not to mention a flurry of excited commentary from those who observe developments in the industry. This book explains the impact of the release of Exchange Server 2013 for those who are involved in Exchange deployment, operations, and support. Much of the insight into the product comes from the other key players, the team that develops Exchange and keeps working to improve it on a daily basis. They have their view about what's important; most of the time I agree with their position (if only after arguing until I understand where they are coming from), and sometimes I disagree. You'll see this dichotomy of views as you go through the different topics presented in this book. I'll begin by presenting the case for Exchange 2013 and explore what Microsoft wanted to accomplish in this release of the product.

Microsoft divides Exchange 2013 into two server roles: Mailbox (sometimes called back-end) and Client Access Server (sometimes called CAS or front-end). This book is dedicated to a discussion of the mailbox server role, although it does touch on the subject of the CAS when necessary to maintain continuity or completeness in a discussion. A companion volume, *Microsoft Exchange Server 2013 Inside Out: Connectivity, Clients, and UM* by

Paul Robichaux, covers the CAS in detail. Both books are intended to stand on their own merits and do not depend on each other, but you can gain a more complete picture of Exchange 2013 by reading both, which I hope you will do.

Microsoft hopes that the quality of Exchange 2013 merits its introduction and that customers consider the new and enhanced features to be compelling enough to warrant a fast upgrade. In addition, Microsoft likes to see an improvement in its competitive situation, something that is especially important in the new era of cloud-based services in which Google in particular has proven to be a worthy competitor in both functionality and setting new expectations for customers for the price point that should be paid monthly per mailbox and the service level that is delivered. Customers want a product that meets their requirements and is easy to deploy and manage; one that isn't too different from previous versions; and deployment of which won't cost an enormous amount in terms of personnel effort and new hardware. Partners hope for new business, whether it's an increase in product sales or an uptick in services revenue to help customers analyze, assess, and then decide how best to use the new software. All this is true for Exchange 2013, which sits at the center of a large ecosystem spanning 360 million deployed mailboxes (the number estimated by the Radicati Group in 2011). The majority of these mailboxes remain on-premises, although there is no doubt that erosion will occur over time as companies move to Office 365 and other hosted offerings.

Understanding development priorities

It's tough to drive innovation into a product that has been around for so long, and it's tough to satisfy all the different constituencies that use Exchange, from the small business that deploys one or two servers to the world's largest enterprises that support hundreds of thousands of mailboxes. Each time Microsoft releases a new version of Exchange, it has to include enough new stuff in the product to create a compelling case for an upgrade.

All development projects have priorities. Microsoft reveals its goals for each version of Exchange when it meets with customers or makes presentations at major industry conferences, such as the return of the Microsoft Exchange Conference (MEC) in September 2012. These goals include:

- **Supporting a multigenerational workforce** The first generation of workers that experienced email was in the 1980s. Each generation since has added its own expectations of how email should work in the mix. The current multigenerational workforce is more diverse and demanding and expects information to be integrated and available through more devices than ever. Microsoft points to the way Exchange 2013 combines information from multiple sources (including LinkedIn and Facebook) to present a unified view of personal contacts ("People") and the way Smart Search works from the way users collaborate to improve search results as evidence of how it is providing better access to information.

- **Providing an engaging experience** The evidence here often includes the reworked user interfaces in Outlook 2013 and Outlook Web App, including support for touch devices. Although some will enjoy features in the new client applications (such as the way Outlook and Outlook Web App permit inline editing of replies to messages), many users will continue to use earlier versions of Outlook because of the difficulty and expense involved in deploying new software to desktops.

- **Improving integration with Microsoft SharePoint and Lync** Exchange 2010 can store Lync conversations for individual users but has no real integration with SharePoint. Lync contacts can now be stored in Exchange 2013, and Lync archiving is subject to Exchange compliance features such as in-place hold. For SharePoint, Exchange 2013 introduces site mailboxes to bridge the gap between how people collaborate through email and document authoring. Search capabilities are also enhanced by using Search Foundation as a common platform across Exchange, SharePoint, and Lync so that information can be located in all repositories. These features depend on SharePoint 2013, Lync 2013, and Outlook 2013 to provide the necessary points of integration and user interface.

- **Meeting evolving compliance needs** Microsoft made a huge investment to provide a broad range of compliance features such as archive mailboxes and retention policies in Exchange 2010. Real-life experience has helped Microsoft refine these features. Search is improved as previously described, and retention holds are expanded to allow multiple query-based holds to be placed on user mailboxes when essential information must be retained. In addition, the new data loss prevention (DLP) feature helps users exert better control over important forms of data that often travel in email, such as credit card information.

- **Providing a resilient solution** There's no doubt that the introduction of native high-availability features embodied in the Database Availability Group (DAG) was the major success story of Exchange 2010. Unlike other features, high availability is based on the heart of Exchange, the Store databases. Experience revealed how automation of the resolution of failure conditions could be improved, and new capabilities were introduced to make it easier to introduce and manage truly resilient mailbox servers. Making the CAS a more stateless server also helps because these servers can now be moved into and out of operational environments more easily. In addition, because a version dependency no longer exists between front-end (CAS) and back-end (Mailbox) servers, it should be possible to update servers of one type to a new version of Exchange without updating the others at the same time.

Because Exchange 2013 builds on the architecture and priorities established for Exchange 2010, it is valuable to review the priorities for that release in the context of Exchange 2013. Because you know how Exchange 2010 has been used in the intervening period, you can assess how the development priorities turned out in reality and how the different

investments in technology made several years ago have developed in Exchange 2013. When it was developing Exchange 2010, Microsoft concentrated on three major areas:

- Increasing operational flexibility through easier deployment, higher availability, and simpler administration. Simplicity comes about in many areas, including the reduction in server roles to just two: the Mailbox server and the CAS. Administration is simplified in that you now have a single management console: the browser-based Exchange Administration Center (EAC). Microsoft Windows PowerShell continues to provide the basis for automation. After a slow start, the Exchange community has truly embraced Windows PowerShell, and the number of useful scripts that are shared on the Internet is growing at a rapid rate.

- Streamlining communications by supporting larger, better-organized mailboxes; investing further in unified communications; and allowing users to work more easily together no matter which device or client they use. Exchange 2010 focused on 10-GB mailboxes with up to 100,000 items in a folder; Exchange 2013 considers a world in which a 100-GB mailbox and 1,000,000 items in a folder might be common.

- Delivering greater visibility and control with protected communications, in-built compliance and archiving functionality, and better reporting and management alerts. A large range of compliance features, including archive mailboxes and retention policies, was introduced in Exchange 2010 to assist companies in complying with various legal and regulatory directives. As explained earlier, features such as discovery searches are refined further in Exchange 2013 and enhanced in new ways such as the provision of site mailboxes, which also create a closer connection between SharePoint and Exchange. The DLP feature comes from experience gained with transport rules and MailTips to enable organizations to define and implement policies to control the transmission of sensitive information through email.

This is not an exhaustive list of the improvements in Exchange 2013. For example, the advent of modern public folders is welcome because it addresses a nagging problem that has existed in Exchange for at least a decade. This book explores the challenges of migrating existing public folder deployments to the new infrastructure in Chapter 12, "Site mailboxes and public folders."

In scanning the development priorities for Exchange 2013, it's interesting that many of the same points could have been made about Exchange 2010. Perhaps it's good that development priorities have remained reasonably consistent, or maybe the same influences that guided Microsoft to make these the priority areas for Exchange 2010 have not abated.

These areas of investment have to work as well for hosted environments as they do when deployed onsite. Security and privacy are big challenges for hosted environments because all communications have to be routed from a customer's own network across the Internet

to a data center Microsoft or another provider hosts. It's not just a matter of transporting messages anymore; directory synchronization and administrative commands have to flow as easily as messages, and everything has to work in dedicated environments and in the multitenant shared environments that are becoming more common because of their cost efficiencies. The debate that erupted following the PRISM controversy in mid-2013 is an example of the sensitivities that exist around security and privacy.

Many of the changes in Exchange 2013 are highly influenced by recent developments in hardware. For example, managed availability imposes a certain overhead on a server because it consumes resources to verify that components are functioning correctly. The overhead might have been a problem for older servers but should not be an issue for the kind of multicore servers available now. Exchange trades memory for disk I/O in a number of versions, based on the principle that memory is becoming cheaper, and it's better to cache data than to go to disk. Exchange 2013 uses larger caches than Exchange 2010, and this, along with the other changes made to reduce or manage I/O better, make it feasible to deploy mailbox databases on low-cost, high-capacity drives. Hardware will continue to evolve, and the Exchange developers keep a keen eye on the possibilities enabled by new capabilities. They also know that how Exchange uses hardware resources has to be as efficient as possible to make it an economic platform for cloud deployments, whose major selling point is often a low monthly cost per mailbox.

At the time of writing, Exchange has been under development for nearly 20 years, and its source code encompasses some tens of millions of lines of code. At one time, the code base amounted to 21 million lines, but a rewrite of the Exchange Information Store into managed code for Exchange 2013 eliminated a large amount of redundant code that handled conditions that are no longer valid. No engineering group stays constant over such an extended period. Different engineering leadership, internal Microsoft politics, and competitive pressure have all contributed to elevating different priorities for the product over the years. Working in a world of cloud services is just the latest influence on Exchange.

The influence of The Service

Since Exchange 2010, Microsoft has had to walk a thin line to develop software that can run as well in a traditional on-premises deployment as in its Office 365 cloud service. Companies have offered hosted Exchange services for years, and many continue to compete successfully against Office 365 with products based on Exchange 2013. The big difference is that Microsoft now runs a massively scalable cloud service that exerts a huge influence over the engineering roadmap. Microsoft is more likely to create new functionality if it is important to Exchange Online, the email component of the Microsoft Office 365 cloud platform, than if it is important to a few on-premises customers. This is the downside of the cloud for on-premises customers; the upside is that Microsoft gains enormously from the experience

of operating Exchange Online. That experience flows back into the on-premises software in the form of better management and automation, a factor this book explores.

The Service is the name the Exchange development group uses for Exchange Online. The name gives insight into how the developers and testers think about Exchange Online; it provides a cloud-based email service to end users.

The early history of Microsoft with cloud-based email was inconsistent. Hotmail (now Outlook.com) was acquired in 1997 and has been a great success as a consumer email service. "Consumer" means that Hotmail offers all the functions and features required by home users but lacks the features, such as compliance, that have become increasingly important for businesses. Although Hotmail gained tens of millions of users, its underpinnings were not suitable to provide a foundation for a business-oriented service. For example, although you can connect Outlook to Hotmail, most of the advanced features in Outlook do not work, among them calendaring. For this and other reasons, Microsoft had to evolve Exchange to become cloud-capable.

The first Microsoft attempt at delivering a business-oriented, hosted email service was based on Exchange 2007, launching as Microsoft Business Productivity Online Services (BPOS) in late 2008. Microsoft was operating other online email services at that time, including the Live@EDU service that focused on the U.S. education market, but BPOS was the first email service targeted at the enterprise market.

The fundamental problem with Exchange 2007 is that it was not designed to run at the kind of massive scale demanded by cloud services. Exchange 2007 was well suited to on-premises deployment, but it had problems that were revealed at scale. CAS was new, the Microsoft front-end proxy technology available at the time (ISA Server) was fragile and limited by its 32-bit platform, and Exchange management tools had only begun the process of automation enabled by the adoption of Windows PowerShell. Hard work and many endless nights that Microsoft support personnel worked closed more of the gaps in functionality. Nevertheless, deficient as BPOS was in many respects, it was a superb learning experience for the architects, developers, and operations personnel who ran the online email service.

Many of the benefits were realized in Exchange 2010, the first version of Exchange that can be regarded as cloud-capable. The improvements and extra stability gained in Exchange 2010 enabled Microsoft to launch Exchange Online as part of Office 365 in July 2011, preceded by a long beta period during which Microsoft gained additional operational experience. However, soon after the formal launch, Office 365 revealed some of the immaturity in processes and procedures surrounding the service. Major outages in August and September 2011 were publicly embarrassing but highly informative. The problems seemed to spur Microsoft to additional effort. No further significant outage was encountered in the next 18 months, an achievement that matched Google's record and exceeded the service level that most companies are capable of delivering in terms of IT service availability.

Exchange Online and Exchange development

The Exchange development group manages Exchange Online. Developers and testers are held accountable for problems that occur in Exchange Online. It is therefore in the developers' and testers' interest to make sure that no code is released for production use in Exchange Online that is not as robust, secure, and scalable as is humanly possible to create. Everyone in the development group knows that if a problem is found, the responsible developer will be called to duty with an automated call from a vice president, commanding him to fix a bug. In this instance, accountability truly drives results.

This approach is invisible to many customers, and those who are responsible for running on-premises Exchange servers might wonder whether there is any evidence that this approach delivers value to them. Indeed, given the investment Microsoft has made in its cloud platforms, on-premises customers might well conclude that they gain no advantage from all this work and are unlikely to see new features appearing in future on-premises versions of Exchange. However, a positive feedback loop exists to ensure that good ideas implemented for the cloud platform find their way back into the on-premises code base. The best example in Exchange 2013 is the managed availability subsystem, by which a series of probes deployed across the product analyze how the service is functioning and take automated action to address problems as soon as possible after they are detected. Automation is the key to successful scaling and operation of cloud-based systems. You cannot afford manual intervention when you manage thousands of servers supporting millions of users, so having code that detects and fixes problems without administrator intervention is clearly a very good idea.

Another example is the evolution of the CAS to become a stateless server, so much so that you can't deploy an Exchange 2013 CAS server in an organization without having at least one Exchange 2013 mailbox server as a companion because the CAS proxies all connections to a suitable mailbox server, including Windows PowerShell commands. If a mailbox server isn't available, CAS can't function.

CAS has been the source of many problems since it first appeared in Exchange 2007. Its scalability was weak, and load balancing and affinity have always presented challenges. These issues are increasingly important, given the growing number of devices that connect to Exchange. Users once had a single device (the PC) that ran a client connected to Exchange to access mailboxes, but they now cope with a profusion of smart phones, pads, tablets, ultra-notebooks, laptops, and other devices that can run email clients. Moreover, if a device supports ActiveSync, IMAP4, or POP3 (listed in order of feature availability), it can connect to Exchange. All these connections have to go through a CAS before they can be redirected (through a proxy) to the correct mailbox server that currently holds the active database containing the user's mailbox. Exchange 2013 moves away from the previous requirement to use layer 7 load balancing to support layer 4 (Transmission Control

Protocol, or TCP), a change that makes the load balancing of incoming connections across a pool of available CAS servers much easier.

At the same time, all client connections now use HTTPS, even those from internal Outlook clients that traditionally have used remote procedure call (RPC) over TCP to connect to mailboxes. Internal Outlook clients connect to Exchange 2013 using RPC over HTTPS, just as external Outlook clients do. (The internal connections use HTTP.) No client makes a direct connection to an Exchange 2013 mailbox server because all connections are forced through a CAS. This change does not mean that Exchange has eliminated its use of Messaging Application Programming Interface (MAPI) RPCs. Instead, these calls are encapsulated inside HTTP packets.

Collectively, these changes make CAS deployment and management easier while also making it possible for individual CAS servers (which can still be deployed in arrays) to be removed from service without causing an impact on clients. Giving the CAS a simplified set of tasks also means that the functionality of mailbox servers can be upgraded without imposing the need to upgrade the CAS, so future product upgrades should be much simpler because you'll be able to run Exchange Server 2013 CAS alongside Exchange 2013+1 mailbox servers. At least, that's the plan.

Microsoft gains many operational advantages through these changes because Exchange Online is the largest deployment of CAS servers in the world. However, so do on-premises customers who have complained about the fragility of the CAS for years.

Many other examples exist to testify to the transfer of improvements made to streamline and strengthen the cloud platform to on-premises servers. The continuing refinement and capability of the Mailbox Replication Service (MRS) is one instance because mailbox moves are performed continually to rebalance load across available servers in Microsoft datacenters; the growing maturity in the high availability of Exchange Server is another. You could not run mailbox databases on low-cost disks if Microsoft didn't need this capability to achieve the necessary operational cost level required to make money when charging $6 a month per mailbox. Features such as single-page patching and autoseed of failed databases are other examples of the kind of functionality that becomes hypercritical when operating at scale while also being extremely useful inside a classic on-premises deployment.

However, the pace of change that occurs in the cloud version of Exchange and its subsequent push-through effect of features that appear in updates for on-premises customers can be difficult to manage. Over the years, on-premises administrators have become accustomed to a relatively predictable and steady pace for feature updates, which usually only appeared in service packs. Over the lifetime of Exchange 2010, as development accelerated for Exchange Online, Microsoft began to ship updated functionality in slipstream or roll-up updates. This took some customers by surprise, but it was really only a pointer to the situation that now exists with Exchange 2013 and Exchange Online, where updates proven

in the datacenter are subsequently released to on-premises customers. Microsoft will not retreat from providing new features as quickly as it can put them into user hands. Microsoft and Google are locked in an ultracompetitive battle in which Google has declared that its "goal is to get to the 90 percent of users who don't need the most advanced features of Office."[1] Given the presence of a very large competitor who wants to grab most of a very valuable franchise, every reason exists for Microsoft to continue to press forward with new features while making the service as economically attractive for customers as possible.

Transition into the cloud

The Service is not going away. Writing about the introduction of Exchange 2013 in December 2012, Gartner[2] said that, although we are in the early days of movement to cloud-based email systems, by 2020, these systems would have 50 percent of the market. Office 365 will not take all of this share, but given the size of the Microsoft installed base, it is fair to expect that Office 365 will support hundreds of millions of active mailboxes at that time.

Another way of looking at the situation is to debate just how deeply cloud services will penetrate across the sectors representing small, medium, and large companies. It is obviously easier for a small-to-medium company to move to a cloud service, especially now that the migration and interoperability tools have been refined. It is harder for a large and complex company to move, but overall, cloud email systems will occupy an increasing portion of the market as time goes by. In the immediate future, companies that meet some or all of the following criteria are most likely to move to Office 365:

- Run Exchange 2003 or Exchange 2007 servers
- Have fewer than 5,000 mailboxes (or fewer than 1,000 in some smaller markets)
- Have no real reason to develop expertise in email
- Struggle to keep email systems updated and functioning properly in such a way that they attain a service level agreement (SLA) of 99 percent or more

In addition, no startup company should ever deploy an on-premises email server unless it has a very clear and unarguable need to do so.

An increasing percentage of the overall Exchange installed base will be hosted by the service in the coming years simply because it is easier and cheaper for many companies to host their mailboxes on the service than to grapple with the complexities of an on-premises deployment. They now have a choice. Even better, the experience Microsoft has gained from Office 365 is demonstrably flowing back into the on-premises product.

1 *http://www.businessinsider.com/google-grabs-office-users-from-microsoft-2012-12*
2 *http://www.gartner.com/id=2231415*

Exchange 2013 architecture

Given the development of different features over the past 10 years and the growing influence of The Service, these factors coalesced into the Exchange 2013 architecture. Briefly, the architecture used in Exchange 2013 can be distilled into four guiding principles:

- Decouple version dependencies.
- Communicate at the protocol layer of the network stack.
- Concentrate functionality into the mailbox role.
- Create building blocks for deployment.

Microsoft wanted to break the close coupling between server roles that exists in previous releases to enable different components to be upgraded in the future without creating the need to upgrade everything. Sometimes, the situation is referred to as a "tight versioning alignment," meaning that components had dependencies that prevented operation if all the components were not upgraded together. Exchange 2013 therefore requires servers to communicate at the protocol layer of the network stack, using a well-defined set of protocols. Direct communication is not permitted even if two components reside on the same physical server. If the implementation works, it should mean that you can update mailbox servers to the latest version of Exchange while continuing to use Client Access Servers that run older software. Figure 1-1 shows how communication between Exchange 2013 Mailbox servers is accomplished using three protocols. Unlike previous versions, when a component such as the Store could communicate directly with the transport service running on another service, all communications are forced to flow up to the top of the stack and then over the most appropriate protocol to a receiving component on another server. This arrangement makes the roles loosely coupled because the only dependency that exists is at the protocol layer.

Alongside an insistence on using protocols to communicate, Microsoft radically simplified the CAS role by moving all rendering and data access functionality to the mailbox role. In many respects, the mailbox role is the core of Exchange because the CAS now acts as a proxy for incoming client connections.

The building blocks for deployment are the DAG for Mailbox servers and the Client Access Array for CAS servers. Both can be deployed independently, but you still need to ensure that a CAS is deployed in every Active Directory site that hosts Mailbox servers.

Of course, you do not have to deploy these building blocks if you don't want to or need to. A single multirole server can provide an excellent email service to a small company. It doesn't provide the kind of resilience that additional servers can provide, but it will work.

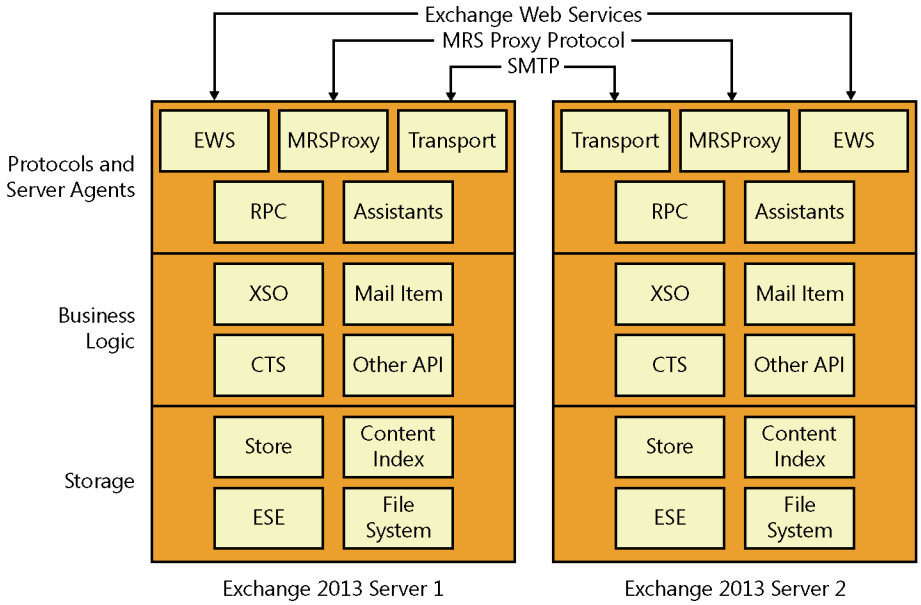


Figure 1-1 Exchange 2013 inter-server communication

The motivation to upgrade

The first point in a deployment project is to understand why you want to deploy Exchange 2013. Different circumstances dictate the ability and willingness of companies to proceed with the deployment of a new version of Exchange, including these common scenarios:

- They might currently run a very early version of Exchange, including Exchange 5.5 (released in 1997 and still in use in limited circumstances today).
- They might have declined to upgrade from Exchange 2003 to Exchange 2010 because their current infrastructure met their needs. They might have wanted to avoid buying new hardware on which to deploy Exchange 2010 or did not want to grapple with understanding the new architecture, perhaps because of other priorities within their overall IT infrastructure.
- They might want some compelling new feature that is available only in Exchange 2013. For example, companies that have standardized on Apple iOS mobile devices might consider the Outlook Web App for iOS apps a must-have capability. Companies that make extensive use of SharePoint might consider site mailboxes a way to maximize the investment they have made in Exchange and SharePoint.
- They might run another mail system and now want to move to Exchange.

Some companies still operate very early Exchange Server versions. Exchange 2003 is a good example; it was deployed by many small companies that installed one or two servers to handle the load of a few hundred mailboxes. It's possible that these companies didn't see the need to move to a new release because the project involved complexity, effort, and expense that could be more usefully deployed elsewhere. Companies running Exchange 2003 now face the fact that the software has reached the end of its formal life cycle, and support will become increasingly difficult and expensive.

In the past, the only available option was to move to a newer version of the on-premises software, perhaps not to the cutting edge but certainly to a more modern version. An extra choice exists now because Office 365 is a good option for many companies that run earlier Exchange Server versions, especially if they will struggle to assign the appropriate IT resources to plan and deploy the new software, including all the attendant updates such as Active Directory, Windows, and third-party software. Companies that run Exchange 2003 servers face the need to move to Exchange 2007 or Exchange 2010 (the preferred option) first if they plan to upgrade to Exchange 2013 eventually, so Office 365 offers a way to avoid a potentially drawn-out and complex migration. This option also enables companies to access evergreen technology; that is, they don't have to worry about upgrading software to be able to use new features because the hosting provider will plan and deploy upgrades as new software becomes available.

Moving to Office 365 also demands resources and might not be a simple switchover, especially if the company decides to operate a hybrid deployment, but it offers advantages in that many of the mundane IT operations required to keep servers running are relinquished to the hosting provider, including the responsibility for keeping software updated. The Office 365 alternative is discussed further in *Microsoft Exchange Server 2013 Inside Out: Connectivity, Clients, and UM* by Paul Robichaux.

Evolving from earlier versions of Exchange

Only two versions support direct migration to Exchange 2013. These are:

- **Exchange 2010** The minimum version required to operate with Exchange 2013 is Exchange 2010 SP3, updated with whichever update is current.
- **Exchange 2007** The minimum version required is Exchange 2007 SP3 RU10 (or later).

No version of Exchange 2003 can exist inside an organization that contains Exchange 2013 servers. Any Exchange 2003 servers must be upgraded to a supported version or removed from the organization before you can deploy Exchange 2013.

If you run Exchange 2007 today, there is less fear of the unknown elements of a new version because much of the Exchange 2013 architecture is at least recognizable in the

context of Exchange 2007 and is therefore not as unfamiliar as it would be if all you know is Exchange 2003. Features that made their debut in Exchange 2007, such as continuous log replication, are in their third iteration. The features might be unrecognizable from what you know in Exchange 2007, but the basic concepts remain much the same. To bridge the knowledge gap, there's a mass of published information from Microsoft and third parties covering topics from basic design approaches to Windows PowerShell code examples.

Companies that do not currently operate Exchange and want to migrate from another email system often have the easiest transition because they have already decided to move to Exchange, and the decision now is which version to deploy. Based on current support policies and previous practice, you can expect Microsoft to provide extended support for Exchange 2007 (assuming the latest update is deployed) until at least November 2017, so plenty of time is available to deploy and use what is now well-understood technology. See *http://support.microsoft.com/lifecycle/search/* for details of the current support availability for previous versions of Exchange.

A move to a new version of Exchange is often combined with a deployment of the latest version of Microsoft Office on the desktop, largely because the latest version of Outlook is usually required to provide the necessary user interface components to expose some of the Exchange functionality. For example, it's necessary to use Outlook 2010 alongside Exchange 2010 if you want users to see MailTips and retention tags. The same is true with Exchange 2013 because Outlook 2013 is required to see DLP prompts and to use site mailboxes. You can certainly use Outlook 2007 SP3 or Outlook 2010 SP1 (both updated with the November 2012 update; see *http://technet.microsoft.com/en-us/library/aa996719(v=exchg.150)*.*aspx*) with Exchange 2013 as long as you don't mind users missing these features.

Waiting for updates

Some might ask: Why wait for a year or so after the original release of Exchange 2013 before issuing a book covering the new version? It's a fair question that deserves a considered response.

All software projects have schedules. The imperative to ship products to customers means that some features are often incomplete or missing in the original, or released to manufacturing (RTM) software. For example, the version of Outlook Web App provided with Exchange 2010 RTM wasn't as good as it should have been and was rewritten to deliver a much better user experience in Exchange 2010 SP1. The same is true of retention policies and tags, which functioned in Exchange 2010 RTM but were much improved in Exchange 2010 SP1.

Product schedules often include nontechnical components such as marketing that influence dates. In the case of Exchange 2013, the situation is further complicated because Exchange is only one of the server applications that form part of Office Wave 15, a term that

describes the concurrent release of new versions of all the server and client applications within the Microsoft Office suite. On the server side, the applications include SharePoint 2013 and Lync 2013; on the client side, the applications include Outlook 2013 and Word 2013. Coordinating the release of such a broad range of applications while also watching other influences such as the release of new server and client operating systems (Windows Server 2012 and Windows 8) requires a huge amount of project management. In fact, it's sometimes surprising that Microsoft manages to release a wave of products at one time.

Given that some compromises are likely to keep a product synchronized with the others in the Office suite, it's not surprising that Exchange 2013 RTM had some failings. The core is solid, but some new features such as site mailboxes depend on other products (SharePoint 2013) or cannot be accessed by users unless new client applications are deployed, and inadequate time was available to gain experience with the migration techniques such as those designed to move older-style public folders to their modern counterparts.

The presence of some software bugs and lack of completeness does not mean that customers who put Exchange 2013 RTM into production gain no advantage. A flaw in one person's eyes might be unimportant to another and, in fact, if you can deploy a complete Office 15 infrastructure, you probably would be very happy with the Exchange 2013–SharePoint 2013–Outlook 2013 combination.

However, most companies that have existing Exchange deployments have waited for Exchange 2013 updates to arrive. The updates are shipped in the form of cumulative updates (CU), similar in many respects to the roll-up updates used for previous versions. The difference is that roll-up updates largely concentrated on providing a set of bug fixes to update Exchange, whereas cumulative updates are really mini-service packs because each update is a complete version of Exchange that can be used to install the product onto a new server.

The anticipated advantages of not rushing to install new software and waiting for updates include:

- **Software feature completion** For example, the RTM version of Outlook Web App could not access public folders, a deficiency that was addressed in Exchange 2013 RTM CU1; other improvements have since been delivered in other updates.
- **Bug fixes and other improvements made since the RTM version shipped** Even though all parts of Exchange are extensively tested during development, including through production deployments by customers in the Technology Adoption Program (TAP), it's inevitable that some flaws will be exposed only after the software is released.
- **Knowledge about deployment techniques** "Best practice" is a curious term. It purports to be the best way to do something, in this case to deploy and operate

Exchange 2013. However, best practice evolves over time as knowledge of a product grows. It is therefore true that anyone who waits to deploy Exchange 2013 rather than using the RTM version will gain from the knowledge acquired since the RTM version was released.

- **Knowledge of migration** In particular for Exchange 2013, this is experience in how to migrate complex or large-scale deployments of traditional public folders to their modern equivalents.
- **Knowledge of the interaction between Exchange 2013 and other Office 2013 components, including SharePoint 2013, Lync 2013, and Outlook 2013**

Largely for these reasons, it has become common practice to wait for the first updates to appear before seriously considering deployment of a new version of Exchange. You could make the point that even more benefit might be gained by waiting even longer. This is true only with respect to bug fixes because the product is now essentially feature-complete.

Fundamental questions before you upgrade

No matter what the situation is, companies have to answer some fundamental questions about why they want to deploy Exchange 2013 before they can proceed:

- Will Exchange 2013 lead to a reduction in existing operational costs?
    - Consolidation might result in fewer servers, leading to cheaper support and administration costs.
    - Virtualization might reduce the number of physical servers that need to be deployed.
    - Cheaper just a bunch of disks (JBOD)-class storage might replace storage area network (SAN) technology.
    - Add-on software might be eliminated because the desired features are now included in Exchange 2013. For example, third-party data replication products can be replaced with native DAGs.
    - Other reasons might also exist.
- What new costs will the company take on to move to Exchange 2013?
    - New hardware (servers, storage infrastructure) might be needed.
    - New or upgraded software licenses for Windows 2008 R2 SP1 or Windows 2012, Exchange 2013, and any associated products (third-party and Microsoft) or a backup product are required. To use specific functionality with Exchange (such as archive mailboxes), you might have to purchase enterprise CALs.

  - Now-discontinued products such as Threat Management Gateway (TMG) and Forefront Protection for Exchange (FPE) might need to be replaced. Some cloud-based alternatives are available, such as Exchange Online Protection (EOP), but in other instances, a replacement product must be sought. In the case of TMG, you have until April 2015 before Microsoft drops mainstream support for the product.
  - If you use load balancers, you might need to reconfigure these to accommodate the different affinity for inbound transactions that Exchange 2013 uses. At this stage, most vendors have well-documented recommendations about how to configure their equipment to work with Exchange 2013. It makes sense to ask your vendor for advice.
  - Client upgrades (Windows Phone and other mobile devices, Outlook 2013, and so on) need to be made. Outlook 2003 is no longer supported, so any devices running this release must be upgraded to run Outlook 2007 SP3 or Outlook 2010 SP1 (both running the November 2012 cumulative update or later). Outlook for Mac users should run Outlook for Mac 2011 updated with the latest update. Note that Outlook for Mac still lacks the functionality available to its Windows counterpart. This is largely due to missing features in Exchange Web Services, the protocol used by Outlook for Mac to connect to Exchange.
  - Training for administrators, help desk personnel, and users must be provided.
  - Consulting is advisable to help make the transition.

- Apart from basic email functionality, which features in Exchange 2013 does the business need?
  - Will you use Unified Messaging (including integration with other Microsoft products such as Lync)?
  - Is better high availability required?
  - Will you use any of the compliance features, including archive mailboxes, discovery searches, or DLP rules?
- What are the major roadblocks to deployment?
  - The need to upgrade other applications, including rewriting code that depends on now-unsupported application programming interfaces (APIs) such as Web Distributed Authoring and Versioning (WebDAV) into code that uses Exchange Web Services could cause difficulty, including the need to train programmers to rewrite code.

  - There is also a need to test third-party applications that integrate with Exchange or wait for vendors to release new versions of their applications that are certified to work with Exchange 2013.
  - Outlook 2013 must be deployed to take full advantage of some of the features of Exchange 2013 such as site mailboxes.

- Can you get the same functionality at the same or better price point elsewhere?
  - Exchange Online includes the option to deploy in a hybrid model by which some mailboxes are supported on classic on-premises servers (running Exchange 2010 or Exchange 2013) and some run in the cloud. You can also place archive mailboxes in the cloud while keeping their active counterparts on-premises. Moving to the cloud seems to be a simple decision, but considerable complexity lurks under the surface.
  - Microsoft is not the only hosting provider that offers Exchange 2013 as the basis for its email service. Many third-party hosting providers, some with many more years of experience than Microsoft in the field, offer a comparable service. These providers might charge a little extra per mailbox per month, but they base their offering on the assertion that they will provide a higher level of support and customized deployment compared to the service available for Office 365.
  - A different email platform might be selected, although this introduces additional work in terms of platform selection, clients, and migration.

After you understand the full context of your current situation and know what the motivation is to deploy Exchange 2013, you can proceed to the planning phase.

Building a business case

No business manager will write a blank check for an upgrade to a new version of any software product. The answers to the list of fundamental questions listed previously will provide good data to help justify the expense involved in the work and additional investment required to deploy Exchange 2013. However, it's also worthwhile to examine some of the technical changes made in Exchange 2013 to discover whether these can deliver additional benefits for your company. Following are some of the major new features in Exchange 2013 to show how they might form the basis of a business case to support the upgrade. There is no guarantee that any of these features are of interest to you or that they support a case for deployment when placed in the context of your company.

Tighter integration with SharePoint and Lync

Exchange 2013 has a much tighter integration with SharePoint 2013 and Lync 2013, two of the other applications in the Office suite. These applications share a common search platform in the Search Foundation (previously known as FAST), so discovery searches can now be performed across SharePoint and Exchange. (Lync conversations have always been storable in Exchange mailboxes.)

The prospect of conducting searches that are more extensive makes compliance officers and lawyers happy. A better business case can be made from the introduction of site mailboxes because these enable much better collaborative document management than has been previously available. With Exchange 2010 and SharePoint 2010, information relating to projects is typically located in two places: mail messages and other items such as calendar meetings stored in the mailboxes of the project members and documents held on SharePoint sites. SharePoint provides facilities such as document versioning that are often used to create and revise project documentation.

Site mailboxes, which depend on Outlook 2013 to provide the necessary user interface, provide an overarching layer to combine the best of Exchange and SharePoint. Internally, a site mailbox is composed of an Exchange mailbox and a SharePoint site, but to the user, the two merge seamlessly so that items can be moved from one repository to the other.

Site mailboxes are still new, and best practice for their design, deployment, and management is still evolving. However, given that we live in a world where electronic documents are the foundation of business, it is reasonable to assume that better document management capabilities are important and that this feature might be valuable to one or more departments within a company. Site mailboxes are discussed in Chapter 12.

Greater compliance

Exchange has been making its way gradually to becoming a platform that is capable of handling the most demanding of compliance requests. Beginning with features such as journaling and moving through transport rules and the introduction of Messaging Records Management (MRM), Exchange 2010 marked a major investment in compliance features with archive mailboxes, retention policies, litigation and retention hold, and multimailbox discovery searches.

Exchange 2013 builds on this foundation with the introduction of data loss prevention (DLP), intended to assist users in understanding and complying with organizational requirements concerning the transmission of sensitive data such as social security numbers and credit card information in email. You can think of DLP as specialized forms of transport rules that scan outbound messages for known patterns representing confidential data. DLP also builds on the MailTips functionality first seen in Exchange 2010 to generate its own informational messages for users who do not comply with policy.

Changes are also made in how administrators can place user mailboxes on hold. Exchange 2010 supports retention hold (for a specified period) and litigation hold (until released). Information cannot be removed from mailboxes when they are on hold. Exchange 2013 refines the set of holds that are possible to set on a mailbox to Indefinite (equivalent to litigation hold), Query (defines the type of information to hold), and Time-based (similar to retention hold except that the time is based on the received or creation date of the items). Items that meet the criteria for holds are retained in mailboxes until needed by eDiscovery searches, hence the name "in-place holds."

Achieving compliance with regulatory or legal requirements is a major business focus today. Being able to achieve that compliance within the email system is critical because so much information flows through email. It's hard to put an exact value on achieving compliance until you experience the costs involved when a company does not satisfy compliance standards, resulting in large legal and other bills. Selling better compliance for email to the business is usually possible, especially if you can replace a third-party add-on product at the same time.

Information Store improvements

The Information Store service is at the heart of Exchange. Simply put, without the Information Store, you would have no access to mailboxes. Compared to its equivalent from Exchange 2003, Exchange 2013 demonstrates just how far Microsoft has come in 10 years:

- The Exchange 2013 Store has been rewritten using C# managed code to make it more efficient and reliable. One of the positive side effects of the rewrite is the elimination of a lot of older code that was present but no longer used because its intended function has been superseded by new developments. The new Store is referred to as the Managed Store to differentiate it from its predecessors.

- The Managed Store is no longer a monolithic process by which a problem can affect every database on a server. Instead, the Managed Store is broken into a series of worker processes, one for each mounted database on a server, and a control process that orchestrates the overall operation of the Store.

- Exchange 2010 introduced native data protection for the Store in the form of the DAG. Despite some initial bugs, the DAG has been very successful in Exchange 2010 deployments. Microsoft has enhanced the DAG with new features such as the ability to host multiple databases on a single disk, easier cross-datacenter transitions, more functionality for lagged database copies, and database autoreseed.

- The Managed Store continues the work begun in Exchange 2007 to drive down the I/O requirements for databases to a point at which Microsoft claims a 99 percent reduction in I/O over Exchange 2003. It's probable that such a reduction can be attained only when storage is very tightly managed. However, it's true that Exchange

2013 is more efficient in terms of I/O than any of its predecessors, so it is possible to deploy Exchange 2013 on very inexpensive disks, providing that databases are protected with multiple copies (at least three) within a DAG.

Because data storage is often the largest single hardware cost for Exchange, these changes mean that Exchange 2013 can deliver better and more reliable performance on lower-cost storage. It is difficult to put an exact value on this benefit because you have to consider existing investment. For example, if your company is already heavily invested in SAN technology that provides storage to many applications, the company might be unwilling to invest in new storage solutions to host Exchange 2013 because it wants to derive a return from the existing investment. Deploying Exchange 2013 on a SAN is a more than acceptable solution, but in this case, you won't gain any additional business advantage. You can find additional information about the Exchange 2013 Store in Chapter 8, "Exchange 2013 Store."

Modern public folders

Some companies could not care less about public folders because they have never used public folders. Others, mainly those that have used Exchange for a considerable time, have very large public folder infrastructures that provide rudimentary sharing and collaboration facilities for users.

Up to now, public folders have always used separate databases from mailbox databases. They have had their own replication mechanism, designed to make data in public folders close (in network terms) to the end user by directing connections to replicas of public folders that are kept synchronized through replication. This scheme made perfect sense when networks were not as capable as they are today, and no one could contemplate having users in one part of the world connect to a remote server located in a data center a continent away to retrieve data.

Exchange 2010 introduced a new form of data replication for mailbox databases. Exchange 2013 now introduces modern public folders that use mailboxes as the basis for their storage and replication. This approach is sensible because it uses the investment Microsoft has made in keeping mailbox databases healthy and reliable through features such as single-page patching. However, the downside is that a migration must be planned to move an existing public folder hierarchy and all its public folders to the new structure. This is a one-time operation, so when it is done, there is no way back.

Companies that have invested in public folders and have much data stored in these folders will be relieved that Microsoft is finally providing a solution to an issue that has been lingering without resolution for a number of releases. Considering the value of the data that is often stored in public folders, it should be possible to make the transition part of a business

case for the deployment of Exchange 2013. After you move the information into modern public folders, you might then consider whether site mailboxes are a better long-term solution for your company's collaborative requirements.

Modern public folders are discussed in Chapter 12.

Management interfaces

Management interfaces is an area that has been simplified in Exchange 2013. Exchange 2010 offers administrators the Exchange Management Console (EMC), the Exchange Control Panel (ECP), and the Exchange Management Shell (EMS); Exchange 2013 rationalizes the set into the EAC and EMS. Much has been written about the demise of EMC, but the loss of a Windows-based console some sixteen years after the original ADMIN.EXE shipped with Exchange 4.0 is not much of a loss because the console, although feature-rich, was also slow and, in some cases, unwieldy.

The introduction of EAC delivers another advantage in that a single administrative interface now operates across on-premises and cloud deployments. Yet another advantage is gained through the cross-browser capabilities of EAC, which enables Exchange to be managed from a wide range of mobile devices, including Apple iPads and Android-based tablets.

Rationalization also removes a dependency that existed within EMC on components maintained by other Microsoft engineering groups. EMC is based on the Microsoft Management Console (MMC) framework, so Exchange depends on the group that builds MMC to fix any problems found in that code. Given the interconnected nature of modern software engineering, in which shared code libraries and components are used extensively, it should come as no surprise to discover that MMC, in turn, has other dependencies. An MMC console is still used by Exchange 2013, but only for a small number of utilities that remain in the Exchange toolbox.

Internally, all the commands executed by EAC are run as Windows PowerShell cmdlets on the server that is being managed. The number of Windows PowerShell cmdlets available to manage Exchange 2013 RTM CU2 is 965, with nearly 200 new cmdlets introduced to deal with features such as site mailboxes and DLP policies. Windows PowerShell is the basis of automation for Windows-based servers, and automation means that a single administrator can become more productive by being able to manage more servers. It also means that operational processes are carried out reliably each time because they can be encapsulated in a Windows PowerShell script that can be executed across a set of servers at one time. True economic value can therefore be derived from automation, but the same value might be gained by using Windows PowerShell with Exchange 2007 and Exchange 2010.

Apart from the potential represented by automation, putting a price on better management interfaces is difficult. Having one (EAC) interface rather than two (EMC and ECP) is good because it's easier to train people to deal with one tool than with several. On the downside, EAC will take time for all to learn and master.

Selecting the right Windows Server for Exchange 2013

Microsoft supports the deployment of Exchange 2013 on Windows Server 2008 R2 SP1 Standard, Enterprise, or Datacenter editions or Windows Server 2012 Standard or Datacenter editions. Microsoft will release Windows Server 2012 R2 on October 18, 2013. No indication is yet available as to what version of Exchange 2013 will run on Windows 2012 R2, but you can expect Microsoft to release an update to support the new operating system around the same time. The remainder of this discussion focuses on the choice between Windows 2008 R2 SP1 and Windows 2012. However, the arguments presented here can be extended to accommodate Windows 2012 R2 after it is available.

The usual choice for deployment of Exchange 2013 mailbox servers is Windows 2008 R2 SP1 Enterprise edition or Windows 2012 Standard edition because these editions enable the servers to participate in DAGs. If you don't need this high-availability feature, for instance when you deploy CAS servers, it is appropriate to use Windows 2008 R2 SP1 instead. Exchange 2013 supports the Datacenter edition of both operating systems, but this option is more expensive in terms of software licensing and does not provide any additional benefit to Exchange.

The Windows Server Core, Essentials, or Foundation server editions remain unsupported and are unlikely to be supported because they are limited versions of Windows designed to be deployed to meet specific needs. No support exists for Exchange 2013 to run on the Itanium (IA64) version of Windows.

Selecting the version of Windows Server for deployment is a critical decision because Microsoft does not support in-place server upgrades (with Exchange 2013) from Windows Server 2008 R2 SP1 to Windows Server 2012. Considering the relative age of the operating systems, you are likely to use Windows Server 2012 sometime in the next couple of years. Therefore, consider using Windows Server 2012 as the basic operating system for your Exchange 2013 deployment. This is much better than creating a situation in which the only way you can upgrade to Windows Server 2012 is by deploying a set of new Exchange servers on new Windows Server 2012 servers, moving mailboxes over to them, and then decommissioning the former Windows Server 2008 R2 SP1 servers. It also makes sense to run the same version of the operating system and Exchange on every server in the organization; this makes support and administration much easier. For these reasons, this book uses Windows 2012 as the example platform for Exchange 2013.

Another point to take into consideration is that Windows engineering has made improvements in some of the critical components that affect Exchange that make Windows Server 2012 the best choice for specific servers. For example, you can achieve a more comprehensive level of automation on a server running Windows 2012 using Windows PowerShell simply because more of the Windows components support Windows PowerShell. (More than 3,000 Windows PowerShell cmdlets are available to interact with Windows components in Windows 2012, compared to 700 in Windows 2008 R2 SP1.) Automating daily operational procedures with Windows PowerShell is a good way to avoid errors and ensure that the same processes are used to maintain servers across an organization.

Many factors need to be considered before selecting Windows 2008 R2 SP1 over Windows 2012 or vice versa, including:

- The cost to upgrade to Windows 2012. For instance, you might have the right to upgrade under the terms of a Microsoft Software Assurance agreement.
- The operating system used in production for other applications today and the experience that exists within the company of Windows 2008 R2 SP1 or Windows 2012.
- The availability of third-party products (such as backup software) used alongside Exchange for the preferred operating system.
- The compatibility of operating processes and procedures with Windows 2008 R2 SP1 or Windows 2012.

In most cases, the choice comes down to personal preference. Some companies enjoy the prospect of pushing ahead with a new operating system to support the new email server; others prefer the comfort of a well-known (and in their minds, well-sorted) operating system combined with a new email server on the basis that it's wise to limit the number of changes in an environment.

Although it is possible to recycle older hardware to support a new version of Exchange, it has become common practice to use new server hardware as the basis for deployment. At the same time, you could also deploy the latest version of Windows Server to create a platform that will last as long as possible. The logic here is that using new hardware matches software and server capabilities as anticipated by the Exchange developers. However, it is true that any modern server manufactured in the past few years is more than capable of handling the load generated by Exchange 2013.

INSIDE OUT **Upgrading workstations used for management**

The Exchange 2013 Management Shell (EMS) can run on either Windows 7 SP1 (x64) or Windows 8–based workstations. Unlike Exchange 2010, workstations running Vista SP2 (x64) are not supported, so you might need to upgrade workstations you want to use for management. However, remember that the Exchange Administration Center (EAC) is now browser-based and capable of running on a wide range of platforms, including Windows Surface RT devices, Apple iPads, and Android tablets. These devices suffice for the vast majority of administrative activities performed against Exchange 2013 servers, and you'll only need a more capable workstation if you want to run EMS either interactively or to execute a shell script.

Using virtualization

Virtualized platforms have made enormous strides in capability, performance, and robustness over the past five years, yet Exchange has not always been desirable in this sphere. In the past, two major factors were at play that caused Exchange to maintain almost a hands-off approach when virtualized servers were contemplated. First, Microsoft did not have a competitive hypervisor, and VMware dominated the market. This is not a technical reason for not supporting virtualization, but it is certainly a good commercial reason for going slowly, especially because Microsoft knew that Hyper-V, its own hypervisor, was coming. Second, Exchange 2003 and Exchange 2007 exert heavy demands on storage subsystems, and users doubted that Exchange servers could run well on virtualized servers when put under strain.

VMware took the opposite attitude of Microsoft and went about the business of proving that it was quite possible to run Exchange 2007 (in particular) on virtual servers. Microsoft didn't support customers who took the advice of VMware that virtual servers were a viable platform, and customers who used VMware were forced to duplicate any problems on a production server before they could look for support. Overall, a highly unsatisfactory situation existed.

The advent of Hyper-V changed the situation because Microsoft could not say that virtual servers worked on one hypervisor but not another. In any case, after a technology company moves into an area, it makes sense for the different parts of the company to expand and embrace the new area, which is what happened when Microsoft development groups began to work with virtual servers and establish just how well (or not) their products functioned. In the case of Exchange, formal support for VMware came from Exchange 2007 SP1 onward with some notable caveats concerning clustered mailbox servers running

continuous cluster replication or standby cluster replication, the predecessors of today's DAG. The potential clash between the high-availability features enabled through virtual servers and those Exchange uses are still problematic. Virtual servers don't know how DAGs work, and DAGs don't know when their member servers are virtual machines, leading to strong guidance from Microsoft to let the application-level high-availability software do its work rather than depending on the hypervisor. Therefore, don't let the virtual platform move DAG member servers around, especially between clustered root servers; you might end up with unpredictable and unsupportable situations. Developments and improvements will come in this area over time, and it's certainly a situation worth watching.

The second issue concerning the demands Exchange placed on storage subsystems was largely addressed by the major improvements in I/O performance in Exchange 2010 with even better performance coming in Exchange 2013. The work done to enable Exchange to run on low-cost disks delivered considerable benefits for virtual servers, so virtual platforms have become very popular in the Exchange community.

Even though the latest versions of VMware and Hyper-V are very capable hypervisors, many still prefer to use physical computers for Exchange mailbox servers. Virtual servers are good choices for CAS servers and Edge servers because these servers have little or no data on them that cannot be recovered easily in case of failure. Virtual servers can also serve as excellent platforms for other parts of the Windows infrastructure, such as print servers or even domain controllers, as long as the core of Active Directory is running on some highly available physical computers.

My attitude toward mailbox servers is very possibly rooted in years of experience with physical computers and the consequences of hardware failure for the core of an email system. The choice to deploy Exchange on physical or virtual machines will be guided by the architectural choices a company makes and the knowledge of the IT administrators. Both Microsoft and VMware publish interesting and valuable documents to provide guidance about how best to deploy applications such as Exchange 2013 on virtual servers. Virtualization is a fast-moving area of technology; it is wise to track developments and be open to advances in that technology.

INSIDE OUT A natural choice for test machines

Although I might be less positive about using virtual machines for Exchange 2013 mailbox servers, I certainly think that virtualized machines are the best platform for all types of Exchange test servers. They provide ease and speed in setting up and deploying different kinds of servers to test various scenarios. The exception to this rule is when you need to stress test Exchange against storage subsystems. In this instance, there's no substitute for testing against the identical configuration that you intend to deploy in production, right down to the microcode that runs on storage controllers.

Preparing for Exchange 2013

Apart from deciding on the operating system, what actions can you take to prepare for an eventual deployment of Exchange 2013, assuming that you run an earlier version of Exchange today? The following is a list that should be supplemented with details of your particular environment, including items such as applications that depend on Exchange:

- Check for required upgrades and hotfixes before you install servers. Exchange affects many parts of the operating system and can expose weaknesses. A check with the TechNet social forum for Exchange 2013 can sometimes reveal a particular combination of Exchange 2013, an operating system, and another Microsoft or third-party product that don't work well when deployed together. A fix might be available for Exchange, the operating system, or the other product—or someone might know of a workaround that will save you time and effort during or after the upgrade.

- If you haven't already done so, move Active Directory to Windows 2003 forest functional mode (or higher). Exchange 2007 and Exchange 2010 share the same requirement. Many consultants recommend that it is best to upgrade the forest to Windows 2008 level if this does not affect other applications. Deploy Active Directory domain controllers and global catalog servers running 64-bit Windows Server 2008 SP2 or R2 SP1. (This is a change from Exchange 2010, which easily accesses Active Directory running on Windows 2003 servers.) Note that Exchange does not support domains that have an underscore in their name because of an internal dependency on X.509 certificates that cannot contain this character.

- Remove any server running Exchange that runs Exchange 2003 or earlier versions; they cannot be installed in a forest that supports Exchange 2013. If you still run Exchange 2007, make sure that these servers run at least SP3 patched with the latest roll-up update (RU10 or later) to ensure that the Exchange 2007 servers can coexist with Exchange 2013 inside an organization.

- For the same reason, servers running Exchange 2010 must be upgraded to SP3 (or a later release). This topic is discussed in more detail in Chapter 2, "Installing Exchange 2013."

- Decide on the version of Exchange 2013 you will use. The choice is between the Standard edition and the Enterprise edition. See "Exchange 2013 Editions" later in this chapter for more information about the features each version supports. You can upgrade from the Standard edition to the Enterprise edition, but you can't downgrade from the Enterprise edition to the Standard edition. If you intend to use the Database Availability Group high-availability feature, you need to run the Enterprise edition of either Windows 2008 R2 SP1 or Windows 2012; you can't upgrade an

existing Windows installation from the Standard edition to the Enterprise edition without a reinstall.

- Use the latest cumulative update for Exchange 2013. There is no point in deploying code that includes well-known bugs. Microsoft releases cumulative updates for Exchange 2013 on a regular basis, and that's the software that you should install.
- Client Access Licenses (CALs) are also required for every user who connects to Exchange 2013. Standard and enterprise versions are available. The enterprise version is additive, meaning that you must also buy a standard CAL for each user. You need the enterprise CAL to use features such as Unified Messaging, advanced journaling, and archive mailboxes.

A word about transport

This volume of *Exchange Server 2013 Inside Out* chiefly discusses aspects of the mailbox server role, whereas *Exchange Server 2013 Inside Out: Connectivity, Clients, and UM* considers the other important elements of an Exchange deployment such as client connectivity, unified messaging, and transport. However, transport is also part of the mailbox server role because Exchange 2013 has eliminated the hub transport role. Transport also turns up on Exchange 2013 client access servers, so this volume must clarify what goes where and what it covers in terms of the mailbox server.

The situation is relatively simple for client access servers, all of which run a front-end transport service that is designed to handle all inbound and outbound SMTP traffic going to or coming from external systems. The transport service running on these servers primarily handles getting mail in and out of Exchange and performs very little content inspection (domains, connections, senders, and recipients). The transport service on a client access server communicates with the transport service on mailbox servers to send and receive messages, but in line with the intention that the client access server should be as stateless as possible, it never queues any messages locally.

If you're upgrading from Exchange 2010 or Exchange 2007, you would consider the Transport service that runs on mailbox servers to be very similar to the processing that hub transport servers perform in those versions, with one extremely important difference. The Transport service handles SMTP mail flow and performs the kind of message categorization and content inspection that you expect from previous versions. However, it never communicates directly with mailbox databases, which is a task assigned to the separate Mailbox Transport service. You can think of the Transport service as the intermediary between the front-end Transport service that runs on client access servers and the Mailbox Transport service that deals with mailbox databases. It is also responsible for managing message queues and stores any queued messages in its transport database.

The Mailbox Transport service is designed to route messages to databases. It actually consists of two components. The Mailbox Transport Delivery service processes inbound SMTP messages from the Transport service and connects and delivers the messages to the currently active copies of the mailbox databases that hold the destination mailboxes by using remote procedure call (RPC). On the outgoing side, outbound messages that originate in Exchange mailboxes are processed by the Mailbox Transport Submission service, which connects to the database and retrieves the messages before submitting them through SMTP to the Transport service to be categorized and routed.

This sounds very straightforward, but the details are explored in depth in *Exchange Server 2013 Inside Out: Connectivity, Clients, and UM*. For now, you know enough to continue with a discussion about the mailbox server.

The test plan

Successful preparation always involves dedicated effort to test more than once to ensure that new software works well in the environment run by a specific company. You must protect yourself from unexpected behavior by designing and executing a comprehensive test plan. The plan should address these points:

- All clients your company uses (in all versions) have to be verified against Exchange 2013. The list might include:
  - All versions of Outlook you currently use.

    Note

    Exchange 2013 does not support any version prior to Outlook 2007 SP3 updated with the November 2012 cumulative update, and not all versions of Outlook are created equal; only the Outlook Professional edition includes the user interface necessary to reveal archive mailboxes.

  - The features and functionality available in Outlook Web App 2013 for the browsers you use (Internet Explorer, Chrome, Opera, Firefox, or Safari), including the various platforms on which these browsers run, such as Windows, Linux, UNIX, and Apple Mac.
  - IMAP4 and POP3 clients (Eudora, Thunderbird, and so on) on whatever operating system platforms you use.
  - Entourage and other Mac solutions. Although earlier Exchange Web Services (EWS)-based Mac clients can connect to Exchange 2013, Outlook 2011 for Mac is the best solution and should be used whenever possible.

  - Mobile clients (Windows Phone, other ActiveSync clients, Apple iPhone, Android devices, and so on).

- The outcome of the client test plan might result in a number of steps you have to take before or during the Exchange 2013 deployment, including:
  - Some Exchange 2013 features (such as DLP Policy Tips and site mailboxes) do not work with Outlook unless you deploy Outlook 2013, so consider how your plans (if any) to deploy Office 2013 might influence your plans to introduce Exchange 2013.
  - Although older Windows Mobile devices can connect to Exchange 2013 with ActiveSync, the latest Windows Phone devices (7.5 and higher) are better partners for both Exchange 2010 and Exchange 2013. If you don't use Windows Mobile devices, select ones that support ActiveSync rather than depending on the IMAP or POP3 protocols to support mobile access to mailboxes. Apple iOS devices can use the Outlook Web App for iOS apps to connect to Exchange 2013 or Exchange Online.

Testing for operational processes

Apart from the purely functional testing, you should also test how well Exchange 2013 will function in your operational environment. There are many changes in the software that will cause you to change operational processes and procedures in different areas. For example:

- Unless you plan to use Exchange 2007 for an extended period, do not deploy additional single copy cluster (SCC) or local continuous replication (LCR) instances for high-availability solutions because both features were deprecated in Exchange 2010. Use cluster continuous replication (CCR) or standby continuous replication (SCR) instead because these are closer to the technology used in the new DAG that replaces both CCR and SCR in Exchange 2010 and Exchange 2013.
- If you use tape-based backup solutions for Exchange, consider how to use a solution based on Volume Shadow Copy Services (VSS) instead. Exchange 2013 does not support backups made with the streaming backup APIs that have been around since Exchange 5.0, so tape backups are no longer possible.
- If you use a third-party archiving and compliance solution, have a discussion with the vendor to understand the vendor's plan to work with or move to the archiving and compliance functionality in Exchange 2013.
- Discuss the permissions model your company uses to control access to Windows resources and applications to ensure that the role-based access control model Exchange 2013 introduces meets the company's security and organizational needs.

Exchange 2013 includes support for a split permissions model (see Chapter 4, "Role-based access control") that will interest companies that like to keep a distinct separation between Windows and Exchange administration.

Testing for programming and customizations

Not everyone wants to exploit the range of APIs and programmable interfaces available to access Exchange data, but you might be surprised when you start to analyze the range of features that people use to work with Exchange in your company and see that different ways of accessing information are important to different groups. Test to ensure that you can continue to deliver the same degree of service to end users after Exchange 2013 is deployed, no matter how they access data. End users include administrators, so interfaces between Exchange and other products (both Microsoft and third-party) are important areas to explore and understand. Some items to keep in mind include the following:

- Understand what customizations you have applied to previous versions of Exchange to ensure that everything is transferred over to Exchange 2013. Microsoft makes sure that customizations such as transport and journal rules are transferred on the deployment of the first Exchange 2013 server in an organization, but you should know what you have so that you know what you should check is still working for Exchange 2013. You will have to reapply customizations such as changes to Outlook Web App themes or front-end logon screens manually. Exchange holds most of its administrative settings in Active Directory, so these are always available even if a server suffers a catastrophic failure. However, settings such as Secure Socket Layer (SSL) certificates are stored outside Active Directory and must be manually updated for Exchange 2013.

- Understand how your company uses older Exchange APIs such as WebDAV, which has been replaced by Exchange Web Services (EWS). You might have to rewrite any custom code that is not using EWS or look for another solution that can be used with Exchange 2013 such as a Windows PowerShell script. As an alternative, you can consider keeping an earlier version of Exchange Server in the organization to support an application that uses a deprecated API until you have the chance to rewrite the code.

- Check any code that uses Windows PowerShell to perform such tasks as system monitoring or account maintenance to ensure that it will continue to work with Exchange 2013.

- Prepare a deployment plan to introduce Exchange 2013 servers into the organization, taking into account advice from Microsoft. The suggested order of deployment is CAS servers and then Mailbox servers (which include Unified Messaging functionality in Exchange 2013). A certain amount of namespace planning and certificate management is required before the first Exchange 2013 CAS can be installed to maintain

external connectivity and to ensure that client connections are correctly referred to their destination.

- Remember that all management components (including the Exchange Management Shell) run on the Mailbox server, so you must introduce at least one Exchange 2013 Mailbox server into the organization at the same time that you install an Exchange 2013 CAS. This is done best by making sure that the first Exchange 2013 server is a multirole server. In fact, the needs of all but the largest or most complex deployments are met best by running Exchange 2013 on multirole servers.

The prospect of going through a migration from one version of an email server to another is never appealing. Depending on your requirements, it might be better to plan for a very different approach such as outsourcing Exchange to a third party, using a traditional on-premises deployment, taking a cloud-based approach through Office 365, or using another hosted offering based on Exchange 2013.

Updating earlier versions of Exchange

If you run Exchange 2007 today, it's likely that you have deployed SP3, the last major update, released in mid-2010. Apart from essential maintenance delivered through regular updates containing cumulative bug fixes and some improvements to features, Exchange 2007 is approaching its natural end of life and doesn't receive much development attention. To prepare for the introduction of Exchange 2013, you need to deploy Exchange 2007 SP3 RU10.

Exchange 2010 is a newer product, only recently replaced by Exchange 2013. Nevertheless, at the time of writing, it also has had three service packs. Exchange 2010 SP3 is the minimum version you need to install to coexist with Exchange 2013. Apart from bug fixes and other code updates, these versions of Exchange understand the new Active Directory schema introduced to define new objects such as site mailboxes used in Exchange 2013. Schema upgrades have to be replicated around the complete Active Directory forest, so this task has to be factored into your deployment plans. Another advantage of deploying Exchange 2010 SP3 is that this release supports Windows Server 2012. This might be important to you if you want to deploy all your servers running Exchange on the same operating system. No version of Exchange 2007 supports Windows 2012.

If you have to update earlier versions of Exchange Server to coexist with Exchange 2013, you might have to update third-party products, too, with a release that vendors have validated to work well with the earlier software. Third-party products might also have to be updated for Exchange 2013 (and verified against the latest cumulative update), so it's a good idea to create a full list of software that runs on your servers that run Exchange and check each product to establish which version is required to support Exchange 2013.

INSIDE OUT **A general rule**

The general rule is that all earlier versions of Exchange servers on the Active Directory site in which you first introduce Exchange 2013 must be upgraded to a version that supports Exchange 2013. You also need to upgrade any earlier Mailbox servers from which you intend to move mailboxes to Exchange 2013.

Deploying earlier versions of Exchange servers alongside Exchange 2013

The possibility exists that you might want to deploy some earlier versions of Exchange servers inside a brand-new Exchange 2013 organization. On the surface, this shouldn't be a problem because Exchange 2013 can coexist with Exchange 2007 and Exchange 2010 in the same organization. However, the order in which you deploy servers is important. If you begin an organization with a server running Exchange 2013, you can install only Exchange 2013 servers in that organization because beginning with Exchange 2013 will render the organization incapable of supporting earlier versions. Therefore, if you think you will need to use Exchange 2007 or Exchange 2010 now or in the future, perhaps to support an application that hasn't yet been upgraded to support Exchange 2013, start by installing a server running the earliest required version (Exchange 2007 or Exchange 2010; you can't run Exchange 2003 alongside Exchange 2013), and then add Exchange 2010 servers if required. Then deploy Exchange 2013. The earlier versions of Exchange servers don't have to be active. They simply have to exist in the organization as markers to create the conditions to permit support of a multi-version organization.

When you reach the point at which the earlier versions of Exchange servers will never be required again, you can remove the servers.

Exchange 2013 editions

Exchange 2013 is available in two editions. The same code is used in both editions, but the license you load by installing a product key determines which edition is active on a server. You can mix and match servers running the two editions within the same organization.

The Standard edition is the least expensive to purchase and contains all the necessary functionality an email server requires, including the ability to participate in a Database Availability Group (DAG). However, it is limited to five databases. If you attempt to mount a sixth database, Exchange 2013 Standard edition will gracefully refuse and issue event

9591, "Exceeded the max numbers of 5 MDBs on this server." You must also operate under the same restriction if you install a trial version of Exchange 2013. Because there is no need for these servers to mount mailbox databases, Exchange 2013 Standard edition is the right choice for standalone CAS servers or as a multirole server when a relatively small number of mailboxes need to be supported. For some companies, the number of mailboxes will be fewer than a thousand. Others will differ, and the choice eventually comes down to the degree of high availability and resilience the organization wants to provide to the mailboxes.

Exchange 2013 Enterprise edition is designed to meet the needs of large enterprises. A server running the Enterprise edition can mount up to 100 databases, so it is the obvious choice for large-scale deployments, deployed as single-role mailbox servers or as multirole servers. The ability to mount up to 100 databases is the only difference between the two editions.

To use a DAG, you need to deploy Exchange on Windows Server 2008 R2 Enterprise edition with SP1 or on Windows 2012 to provide the Windows failover clustering feature that underpins the DAG. Although it is possible to construct and operate a DAG using Exchange 2013 Standard edition, the five-database limit restricts the number of mailboxes that can be supported and the degree of resilience that you can incorporate into the DAG through database copies. For this reason, unless you are confident that a DAG based on Exchange 2013 Standard edition is capable of meeting all your needs, it is better to use Exchange 2013 Enterprise edition within DAGs.

Any server that has not had a license loaded is deemed to run the trial version. After 120 days of running a trial server, you'll begin to receive notifications that it's time to install a proper license. Installing a valid product key is sufficient to transform a trial version of Exchange into a licensed (and therefore supported) version. Exchange also enables you to input an Enterprise edition product key to upgrade a Standard edition to the Enterprise edition. However, it does not allow you to downgrade a server from Enterprise to Standard, so you must ensure that the correct product keys are installed on the servers. If you make a mistake and need to downgrade a server from Enterprise to Standard, the supported method is to uninstall the server from the organization and then reinstall Exchange.

Apart from deciding on the version of Exchange to deploy, you'll also need to be sure that you buy the right user CALs. Use of some Exchange features such as archive mailboxes require an enterprise CAL per mailbox, as does voice mail integration, the discovery and retention features, and extended Exchange ActiveSync policies. The enterprise CAL is additive; first you buy a standard CAL to license the standard features, and then you purchase an enterprise CAL if you need to use the extended features. Refer to the Exchange Server 2013 licensing page at *http://www.microsoft.com/exchange/2013/en/us/licensing.aspx* to get an idea of what licenses are required for the specific features you want to use.

Active Directory

The successful deployment and management of Active Directory has been a fundamental prerequisite for Exchange since Exchange dropped its own directory service in Exchange 2000. In fact, Active Directory shares many of the characteristics of the earlier Exchange directory services, and Microsoft learned a lot about how to design and operate directories from the experience it gained with Exchange 4.0 to Exchange 5.5.

Using the strong link between Exchange and Active Directory

By any measure, Exchange makes the most extensive use of Active Directory of any Microsoft application. When Exchange is installed for the first time, the installation procedure does the following:

- Extends the Active Directory schema to add a large number of new objects and attributes to support Exchange features and allow the creation of objects such as connectors in Active Directory. New attributes such as email addresses extend existing objects such as users, contacts, and groups to enable them to work with Exchange. Other attributes are created for new objects and are used for those objects only. For example, the ability to define a maximum number of active databases that a server can support depends on the *msExchMaxActiveMailboxDatabases* attribute (almost all the objects and attributes that belong to Exchange are prefixed with "msExch") that the Exchange installation program adds when it prepares the Active Directory schema before servers can be deployed.

  Most of the attributes are single valued, such as the database that hosts a mailbox. Others are multivalued, such as the email addresses for a mailbox. *LegacyExchangeDN* is the most famous attribute Exchange added because it has served as an X.500-based identifier for Exchange objects since the first version of the product and now provides backward compatibility for applications that rely on it to identify objects such as mailboxes. Today, Exchange relies on globally unique identifiers (GUIDs), 16-byte numbers, when it needs to assign an identity that won't change to objects.

- Creates a container called Microsoft Exchange under the Services root in the Configuration Naming Context. All the configuration data about objects Exchange manages are located here. (You can see the structure of the container in Figure 1-2.) The objects include servers, policies, address lists, connectors, and so on. Data about mail-enabled objects are not held in this container. Instead, they are stored as attributes on the mail-enabled objects themselves, so they usually are found in the organizational units (OUs) in the default naming context. Mail-enabled objects use the extended schema to populate attributes such as email addresses. The Exchange Administration Center (EAC) relies heavily on data fetched from the Exchange container.

- Adds a set of attributes to the Partial Attribute Set (PAS), which is the subset of Active Directory data that is replicated to global catalog servers and is available to every domain within the forest. For example, Exchange adds many mailbox attributes, such as the GUID that allows the Store to locate a mailbox or the full set of Simple Mail Transfer Protocol (SMTP) addresses for a mailbox, to ensure that message routing works throughout the forest. Other user data form the basis of the Global Address List (GAL) and are added to the PAS so that it is available everywhere.
- Adds a set of extended rights (permissions) to enable administrators to manage objects such as databases. These rights are defined in the form of the roles and permissions managed through Exchange's Role Based Access Control (RBAC) system.
- Adds and extends the set of Active Directory property sets (groups of properties) to make management easier by managing sets of properties as single entities rather than handling the individual properties that make up a set. For example, the Exchange Personal Information property set contains all the attributes Exchange holds about individual users, such as their phone numbers.

Note that Exchange 2013 cumulative updates might also require a schema update or some other preparatory work to be done with Active Directory before the update can be applied. For instance, Exchange 2013 RTM CU2 required administrators to update the schema and then update the Exchange configuration data held in Active Directory with new RBAC role definitions and other information. Table 1-1 summarizes the position by showing where Exchange stores the data on which it relies within Active Directory.

TABLE 1-1 Where Exchange stores information in Active Directory

Partition	Active Directory Location	Exchange data stored	Replication scope
Domain	Dc=domain, DC=parent domain	Mail-enabled recipients (groups, contacts, accounts)	Every domain controller in the domain
Configuration	CN=Microsoft Exchange, CN=Services, CN=Configuration, DC=domainroot	Exchange configuration objects such as policies, global settings, address lists, templates, and connectors	Every domain controller in the forest
Schema	CN=Schema, CN=Configuration, DC= domain root	Exchange-specific classes and attributes	Every domain controller in the forest

INSIDE OUT **Coming to terms with Active Directory schema upgrades**

Active Directory schema upgrades are not popular with Active Directory administrators because after a domain controller learns that a schema upgrade is in progress, it halts all other forms of Active Directory replication until its schema is upgraded. The fear is that schema upgrades will interfere with the normal work of Active Directory and affect other applications.

This fear can be addressed by understanding that schema upgrades will replicate quickly throughout a forest if Active Directory replication is healthy, and the risk to other applications can be reduced by scheduling the schema upgrade for a time of low user demand. One thing is certain: Schema upgrades are a fact of life when you work with Exchange. The best thing to do is to ensure that Active Directory replication is healthy and to agree on operational procedures to apply schema upgrades in your environment.

ADSIEdit

ADSIEdit has proven its worth to Exchange administrators to fix problems with Active Directory objects or simply to help them understand the complex relationship between Exchange and Active Directory. Figure 1-2 illustrates the utility in use to browse the Exchange container and review the values of properties of an object (in this case, details of the administrative audit policy for the organization; see Chapter 8 for more information). Microsoft refers to ADSIEdit as a Lightweight Directory Access Protocol (LDAP) editor to manage attributes in Active Directory. It is installed on servers running Windows Server 2008 R2 and Windows Server 2012 along with the Active Directory Domain Services role, after which you can access ADSIEdit in Administrative Tools in the Start menu. After you start ADSIEdit, you must connect it to the Configuration Naming Context to access Exchange data.

ADSIEdit began in Windows 2000 as an add-in for Microsoft Management Console (MMC) that Microsoft didn't really support, probably because it didn't want administrators to have the ability to mess up Active Directory by editing objects and properties. Remember that ADSIEdit allows you to work with raw data through a user interface that doesn't do much to protect you with warnings about the potential consequences of certain actions such as editing an attribute.

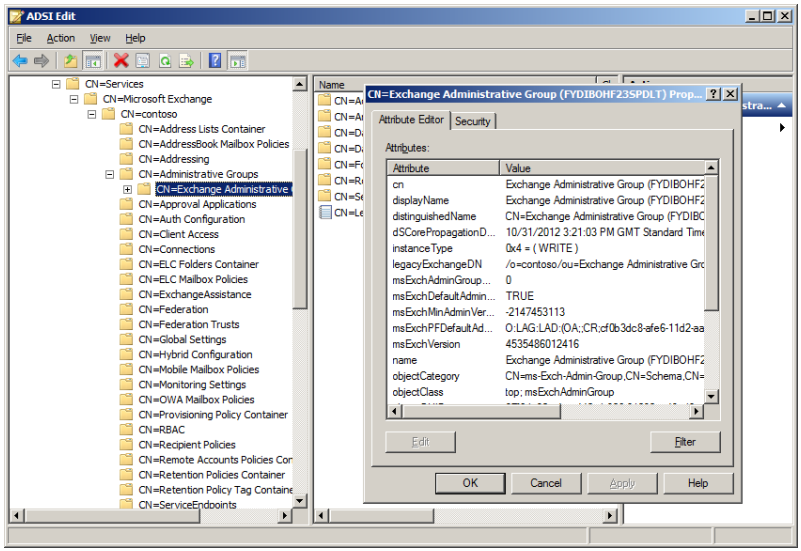


Figure 1-2 Exchange 2013 objects in Active Directory

CAUTION!

You can do great harm with ADSIEdit if you don't know what you are doing. It's best to be cautious and use ADSIEdit to gain an insight into the internal workings of applications such as Exchange that depend on Active Directory to hold their configuration data.

Useful utilities

Every Exchange administrator has her own collection of utility programs to which she returns on a regular basis to solve problems or otherwise move issues forward. ADSIEdit is an obvious example, but others appear on an ongoing basis. In this section, I describe two utilities that I consider worth having in your collection. Other utilities that I like very much include the Remote Connectivity Analyzer (RCA), which is described in depth in *Microsoft Exchange 2013 Inside Out: Connectivity, Clients, and UM* by Paul Robichaux (Microsoft Press, 2013), and Microsoft's Office Configuration Analyzer Tool (OffCAT), available for download from *http://support.microsoft.com/kb/2812744*.

MFCMAPI

As you read this book, you might be struck by the number of times I use the MFCMAPI utility to access information held in mailboxes that is not usually exposed to clients. MFCMAPI is available from the Codeplex (*http://mfcmapi.codeplex.com/*) website and is available in both 32-bit and 64-bit variants. Executables and the source code can be downloaded, and new versions are released on a regular basis. Because of its unique ability to unveil the mysteries of mailbox contents, MFCMAPI is an essential component of a serious Exchange administration toolkit. The program was called Microsoft Exchange Server Messaging API Editor; now it's just called MFCMAPI. Neither name does the program justice; its name should be something like Exchange Mailbox Internals Spy.

MFCMAPI depends on the availability of Outlook on the workstation on which it runs, and you need to match the correct versions of Outlook and MFCMAPI before MFCMAPI can run. It can be run alongside both Outlook 2010 and Outlook 2013 without any problems. It's best to run MFCMAPI when Outlook is configured in online mode because this exposes all the mailbox contents to MFCMAPI. Much of the more interesting mailbox contents such as the various subfolders of the Recoverable Items folder structure are not replicated to the PC when Outlook runs in cached Exchange mode.

MFCMAPI began as a training exercise in 2001 for Microsoft developer Stephen Griffin to learn how to program in the Messaging Application Programming Interface (MAPI). The MFC prefix indicates that he used the Microsoft Foundation Class Library as a framework. MFC is a library that encapsulates parts of the Windows API in C++ classes and thus makes it easier for programmers to work with the Windows API (so the theory goes).

Over the past decade, Stephen Griffin has done an incredible job of maintaining MFCMAPI to keep it up to date with the latest versions of Exchange, fix bugs, and add new features. Although Stephen has been working to improve the MFCMAPI user interface, it does not work like Outlook. The advantage delivered by MFCMAPI is that it exposes mailbox internals by using MAPI, so you can expect to encounter MAPI structures and names. It's easy to figure out, but don't attempt to make changes unless you know exactly what you are doing. As when operating on Active Directory data with ADSIEdit, it's possible to cause problems in a mailbox if you make a change in the wrong place.

When you start MFCMAPI, you select an Outlook profile to use to connect to Exchange. MFCMAPI reads the profile data and presents a screen similar to Figure 1-3. In this instance, you can see that a number of mailboxes are available because they are listed in the profile, including mailboxes that have delegate access. The mailboxes include archive mailboxes linked to the primary mailboxes. The default store—the one to which Exchange delivers email—is marked. You can access on-premises and Office 365 mailboxes and those such as Outlook.com that Outlook accesses through IMAP. You're not going to send or receive email. All you do with MFCMAPI is investigate and interrogate MAPI structures within mailboxes.

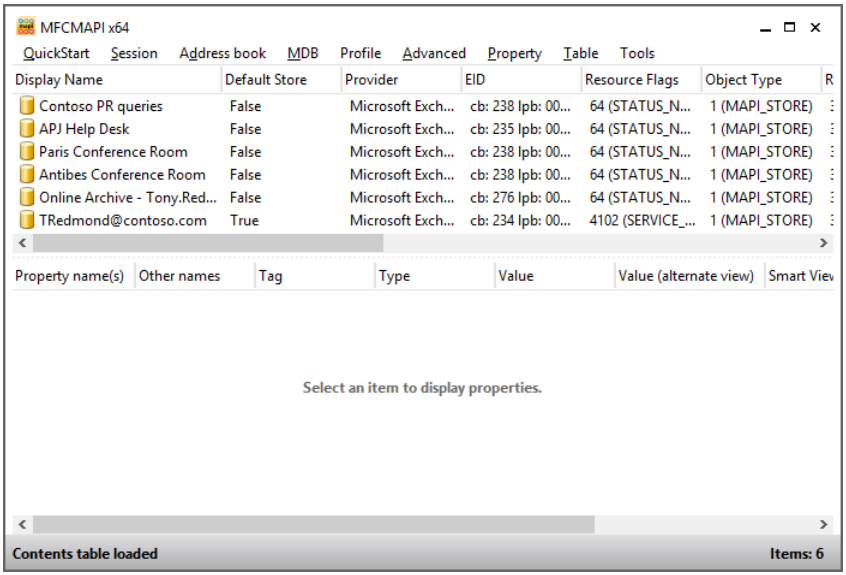


Figure 1-3 MFCMAPI opening a set of mailboxes

When you open a mailbox, you can navigate through its contents and examine the folders Outlook displays (click Top Of Information Store to reveal your folders) and those it doesn't, such as Recoverable Items and its subfolders. In Figure 1-4, you can see the opened Sent Items folder in a mailbox. Twenty-eight items are in the folder. The MAPI properties for the selected message are shown in the bottom pane, and you view full details (or edit) many of these properties by clicking them.

The best way to learn MFCMAPI is to run it to examine mailbox structure and contents. You might like to run the program against a test mailbox and experiment with making changes just to see what you can and cannot do and the effects of making the changes. (The program does prevent you from making changes to some essential data.) Using a test mailbox is probably better than running MFCMAPI against your own production mailbox unless you like to live life on the edge of a razor.

Apart from examining mailboxes to learn how Exchange organizes data, when should you use MFCMAPI to solve problems? One common example is if a mailbox has become corrupt for some reason, and you need to work with Microsoft support to resolve the problem. In this scenario, you're likely to be told exactly what to do, and you might have to remove items that are suspected of being corrupt. As you will find in Chapter 8, the Information Store can protect itself from the side effects of dealing with corruption within a mailbox by isolating the mailboxes and preventing users from logging on to them for six hours. Exchange might deem a mailbox to be corrupt if too many threads freeze when a client attempts to retrieve data. In this case, the only access to the mailbox is possible through

a program such as MFCMAPI. If the user can identify the items that cause the freeze, you could remove them by using MFCMAPI to see whether your action addresses the problem.

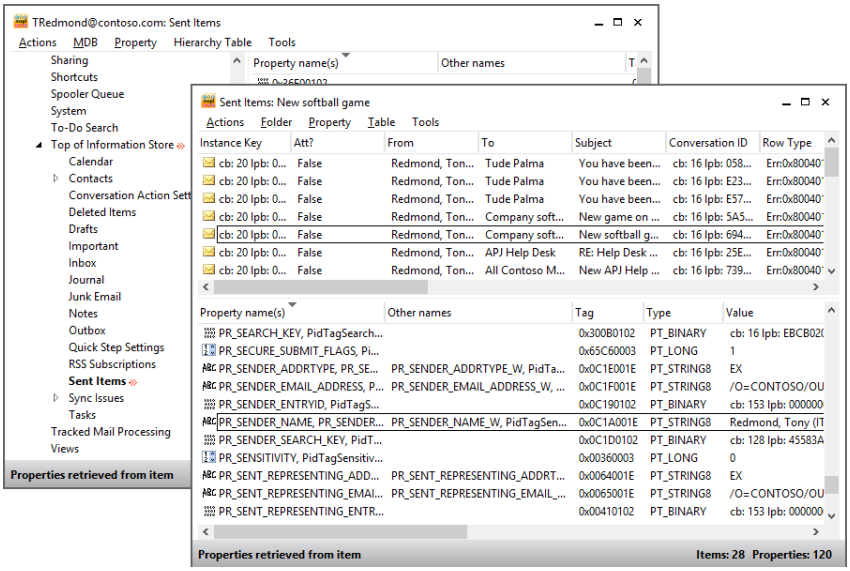


Figure 1-4 Navigating within a mailbox with MFCMAPI

Exchange Web Services Editor

The best thing about MAPI is the extensive feature set it enables for clients such as Outlook and, in its server-side variants, Exchange server. I guess you could argue that MAPI has been very persistent in that the name first appeared as "simple MAPI" in Microsoft Mail way back in the dim and distant early nineties. The functionality you can build on the 12 functions supported by simple MAPI is barely sufficient to create and send messages and can't approach the kind of feature set in Outlook 2013, but it was a start.

Powerful as it is, MAPI has never attracted much developer enthusiasm outside Microsoft and has been viewed as an API that should be approached only when necessary. The lack of good documentation is one possible reason, and the result is that only heavy-duty engineering projects, such as those to create MAPI providers that allow Outlook to access non-Microsoft servers, have used MAPI. (*Inside MAPI* [Irving De La Cruz and Irving Thaler: Microsoft Press, 1996] has been out of print for many years.)

Microsoft has shipped many other messaging-related APIs in an attempt to make Exchange data more accessible or usable, including esoteric offerings such as Collaborative Data Objects (CDO) Routing (released with Exchange Server 5.5) and WebDAV (which was

supposed to be the next great thing when Exchange Server 2000 was released). However, MAPI remains optimal, which is why tools such as MFCMAPI remain so important.

Over the past few years, Exchange Web Services (EWS) has transformed matters. EWS is now the recommended API for developers who want to access and work with the contents of the Exchange Store. Microsoft uses EWS for clients such as Outlook 2011 for Macintosh and Outlook Web App, and plans are in place to extend EWS further as new versions of Exchange appear, probably to close the functionality gap between EWS clients and the MAPI-based Outlook. As far as I am aware, no plan exists to discard MAPI for EWS in the short-to-medium future, but this is a possibility when EWS supports the same feature set as MAPI does today.

In the interim, it seems that EWS is a good thing to know something about. The Exchange Web Services Editor (EWSEditor, *http://ewseditor.codeplex.com/*) is another Codeplex project that's built on top of .NET Framework 3.5. Like MFCMAPI, versions of EWSEditor are released on a regular basis and work with on-premises servers running Exchange 2007, Exchange 2010, or Exchange 2013. The utility also supports Office 365. If you have questions about how to program using EWS, you can find many examples and advice on MVP Glen Scales's website (*http://www.gsexdev.blogspot.com/*).

Installing Exchange 2013

You now know the landscape for Exchange 2013, how to prepare for deployment, the operating system to use, the versions that are available, and how Exchange relies on Active Directory to provide its foundation. The next step is to install the software on a server, which is discussed in Chapter 2.

CHAPTER 2

Installing Exchange 2013

Approaching the installation 43

Types of Active Directory deployment that support Exchange 45

Preparing Active Directory for Exchange 47

Creating the Exchange 2013 organization 49

Deploying an Exchange 2013 server 50

Versions, cumulative updates, and service packs 63

Security groups and accounts Exchange creates 73

Namespace planning 77

Contemplating management 81

Installing Exchange Server 2013 is not a difficult operation if you do the necessary work to prepare for the task. You can install and configure a test server in well under an hour; production-class servers take longer because of the additional care required when working in an environment in which a mistake can affect the service delivered to users. Whatever the circumstances, the fundamental steps to install Exchange 2013 are very similar. You have to prepare Active Directory, install prerequisite software, and then deploy the required Exchange server roles on target computers. This chapter covers all these topics and discusses the next step in a server's evolution, which is when an upgrade becomes available.

Approaching the installation

If you have prepared the environment by installing the various prerequisites on the servers that will host Exchange, the Exchange installation process is straightforward and painless. The steps you'll take include the following:

- Select a name for the organization if you are deploying Exchange for the first time. The name can be up to 64 characters and cannot be blank. You can include spaces in the name; if you include spaces and install Exchange from the command line, enclose the name in quotes. You don't need to enclose the name in quotes if you install with the graphical user interface (GUI). Users don't see the organization name. Usually, the organization is named after the company that owns the system. However, this is not compulsory. Indeed, because of the number of corporate mergers, some consultants recommend choosing a nonspecific name such as Email or Exchange.

- Identify the target servers on which to install Exchange and the roles each server will support.

- Decide on a name for each server. A good naming convention conveys the purpose and use of a server without forcing the administrator to examine server properties to discover its purpose.

- Install prerequisite components such as Microsoft .NET Framework 4.5, Microsoft Windows PowerShell 3.0, Windows Unified Communications Managed API 4.0, and Active Directory remote management tools on the servers on which you want to install Exchange 2013. Windows 2012 servers include Windows PowerShell 3.0, .NET Framework 4.5, and Windows Management 3.0, so less work needs to be done to prepare these servers, but it is always a good idea to consult the Exchange 2013 content on TechNet to validate the current list of prerequisite features and to understand whether any additional Windows hotfixes or other software must be installed before Exchange. As discussed in the "Deploying an Exchange 2013 Server," section later in this chapter Exchange 2013 can install prerequisite software when you install a server.

- Remove or upgrade any servers that run unsupported versions of Exchange because these will stop the Exchange 2013 installation program. Exchange 2013 will not install if it detects a server running a version earlier than Exchange 2007 SP3 in the organization. Exchange 2010 servers must run Exchange 2010 SP3. In addition, Exchange 2007 and Exchange 2010 servers must be updated with the latest available roll-up update.

The example server used for the remainder of this chapter runs Windows 2012 Standard edition.

Most companies find it easiest to deploy multirole servers, and this approach is described in the remainder of this chapter.

INSIDE OUT **Server names are not quite so important anymore**

You should use a naming convention for servers to identify the role and purpose of the server. However, ever since Exchange 2010 broke the link between server and mailbox with the introduction of the Database Availability Group (DAG), you can no longer associate a user's mailbox with a particular mailbox server. In fact, although Exchange 2013 continues to stamp attributes such as HomeMTA and msExchangeHomeServerName with the names of mailbox servers to maintain backward compatibility with previous versions, the Get-Mailbox cmdlet no longer reports these values. (You can view them with ADSIEdit.) Instead, Exchange 2013 uses a globally unique identifier (GUID) to locate the active copy of a user's mailbox. Because GUIDs don't usually make sense to humans, the new method might be less easy on the eye, but it makes perfect sense to Exchange.

Types of Active Directory deployment that support Exchange

Active Directory implementations in use today extend from a single forest to multiple forests, and there are many variations in between. Some companies deployed Active Directory early and have an implementation that features a root domain that contains a small number of servers, such as the schema master, with applications and users deployed into a set of geographic or business-centric domains. Other companies have deployed multiple Active Directory forests because different businesses require separate security contexts. Yet other companies have acquired multiple forests through buying companies and are merging applications, domains, and forests into a more streamlined design. Some companies have deployed a single domain forest that supports everything. In large companies, it is quite common to find Active Directory under the control of a team separate from the group that manages Exchange. From the previous discussion about how Exchange uses Active Directory, you can see that close coordination of all concerned is necessary to achieve a smooth deployment.

Exchange supports three basic Active Directory designs. Apart from understanding that a forest can support only a single Exchange organization (mapping is always one to one between a forest and an Exchange organization), remember that a forest (and not a domain) is the security boundary for Active Directory. Thus, after you deploy multiple forests, you have to make sure servers and user accounts can access the resources they need to do work, regardless of the forest in which the resource is located:

- **Single forest** This is the simplest design and is normally most appropriate for small to medium deployments because it removes a great deal of complexity. All servers in the forest share the same schema and configuration, even if multiple domains are deployed, and difficulties are not usually encountered with security and permissions because all objects operate within a single security context.

- **Multi-forest** Sometimes referred to as cross-forest implementation, this deployment features two or more forests that operate independently from each other in terms of the accounts and applications that are deployed inside each forest. It is possible to synchronize user account information between forests so that users can access a common directory that includes everyone. Exchange can be deployed in one or more forests. A hybrid deployment, where some parts of Exchange are on-premises and some are located on a cloud platform, can be considered similar to a multiforest configuration.

  When Exchange is deployed in multiple forests, the organizations have different names, SMTP addresses, configurations, and so on and a messaging flow between the organizations can be established by SMTP connections to complement the common directory. Multi-forest deployments usually occur as the result of company

acquisitions. Less commonly, companies have deployed Exchange in multiple forests to satisfy particular requirements for specific parts of the company that were inconsistent with the needs of other parts. Exchange 2013 supports the ability to move mailboxes between organizations, so this is not an obstacle. It's also possible to use the Exchange 2013 Exchange Administration Center (EAC) or Exchange Management Shell (EMS) to access multiple organizations at one time. However, the different security contexts created by multiple forests make management more difficult and complex than in a single forest.

- **Resource forest** In this scenario, user accounts and groups are deployed in a root forest, and applications such as Exchange are deployed in a special resource forest. Exchange mailboxes exist in the resource forest and use disabled user accounts that belong to the resource forest. Users who log on to the root forest gain access to their mailboxes in the resource forest through an association with the mailboxes.

  Resource forests have some attraction because they allow a very clear separation of administrative responsibilities between Windows (the root domain) and applications, each of which is assigned its own resource domain. Resource domains are often used in a hosting environment in which the hosting provider creates a resource domain to host and manage Exchange. A one-way trust is usually used to connect the resource forest with the root forest because this allows the hosting provider to manage Exchange without having any control over Active Directory.

There are many variations on these three basic themes in production today. For example, a design composed of root and resource forests is often referred to as a hybrid forest if mail-enabled accounts exist in both forests. You then get into the discussion about whether to deploy one domain per forest, multiple domains per forest, or a mixture of both. Multiple domains are commonly used to isolate Exchange management and operations or to distribute administrative activity across business units or geographic regions. Cost is another factor to include in the planning debate. Additional domains normally imply the need to deploy extra servers as domain controllers. Extra servers increase the complexity of the environment that you have to manage, monitor, and operate, and they drive cost in terms of hardware (servers and storage), datacenter power and cooling, network bandwidth, and software licenses.

INSIDE OUT **Think before you leap**

It is important to settle on the most appropriate design for your company before you begin to deploy Exchange because it is very difficult to change the basic shape of an Active Directory forest after it has been deployed. If you're unsure about how to proceed, it's advisable to start simple and deploy a single forest to support Exchange. The old KISS (Keep it simple, stupid) adage comes to mind here! Later, if you find that this model does not meet business requirements, you can evolve it by adding forests or even Exchange organizations. Remember that every additional domain and forest adds complexity—and therefore cost—to your implementation, so think long and hard before you venture away from a single forest.

Preparing Active Directory for Exchange

Out of the box, Active Directory knows nothing about Exchange, so you have to prepare Active Directory to support the deployment of your Exchange organization. Running any setup operation that updates Active Directory can only be executed on a server that has the Active Directory Domain Services (AD DS) feature installed. To install this feature on a server running Windows Server 2012, start Windows PowerShell and type the command:

```
Install-WindowsFeature RSAT-ADDS
```

The basic framework for preparation of Active Directory for a new Exchange organization is as follows:

1. Make sure that the forest runs in Windows Server 2003 functional mode or higher. The computers used for domain controllers and global catalog servers should run Windows Server 2008 Standard edition or Enterprise edition at a minimum; Windows Server 2008 R2 (or greater) is the preferred operating system for Active Directory.

2. Prepare the forest by adding or modifying elements in the Active Directory schema to support Exchange. This operation has to be done once for the forest and should be run in the domain that contains the schema master for the forest. You can do this by running the Exchange setup program with the /PrepareSchema or /PS switch. After this command completes, the Active Directory schema contains all the classes and attributes required to support Exchange.

3. Introduce Exchange configuration data into the forest by running the Exchange Setup program with the /PrepareAD and /OrganizationName switches. After this command completes, you will find a Microsoft Exchange container created to hold

details of the organization, from servers to databases to connectors. This process also creates the universal groups required to manage Exchange and sets appropriate permissions on objects so they can be managed.

4. Prepare every domain that will contain an Exchange server or an Outlook client by running the Exchange setup program with the /PrepareDomain switch on a server in the domain you're preparing. You can also run Setup with the /PrepareAllDomains switch to prepare all the domains in the forest.

You cannot perform these operations unless your account has administrator permission for the target forest or domain. You'll also need to be a member of the Schema Administrators or Enterprise Administrators group to have sufficient permission to modify the schema before you can run the installation procedure with the /PrepareSchema switch. In addition, your account needs to be a member of the Exchange Organization Administrators security group (which is created when you prepare the organization) to be able to prepare a domain. Replication has to occur after each step to ensure that all domain controllers are at the same level, so it's wise to leave an interval between each step to allow this process to complete.

Although the most common need for schema extensions arises when Exchange 2013 is first deployed for an organization, it's important to recognize that schema updates might be required before a build-to-build upgrade can be performed. For example, Exchange 2010 SP3 uses a slightly different schema from Exchange 2013 RTM CU2, so one update must be applied to Active Directory before you can deploy the first Exchange 2010 SP3 server, and another update must be applied before Exchange 2013 RTM CU2 can be installed.

Remember that edge servers are informed about the version of the Active Directory schema through the subscription they have with the Exchange organization, so you need to take out a new subscription for each of these servers after you update the schema.

Although each of the steps described above can be executed separately, if you are installing Exchange from scratch to build a new organization, it is sufficient to run the installation procedure because it will do all the work to extend the Active Directory schema, add the necessary Exchange objects to instantiate the new organization to the configuration naming context, and install the new server. Subsequent server installations add their own objects to the organization data.

Creating the Exchange 2013 organization

After the Active Directory schema has been successfully prepared, you can create the new organization unless you already have an Exchange organization in place.

The Setup command to create a new Exchange organization called contoso is:

```
Setup /OrganizationName:contoso /IacceptExchangeServerLicenseTerms
```

Note the IAcceptExchangeServerLicenseTerms parameter. This must be input to indicate that you accept the terms of the Microsoft license to run Exchange. Previous versions of Exchange output some legal text that you had to read and then input a key if you disagreed with the terms. This method was deemed unsatisfactory because the administrator did not explicitly agree with the terms of the license, hence the new parameter, which administrators soon become masters at typing.

A number of Active Directory user accounts are created to support the operation of different Exchange 2013 components. Figure 2-1 highlights the accounts Setup creates. Users are unaware of these objects because they are used for background or housekeeping purposes, but Exchange 2013 swiftly runs into difficulties if they are not available.

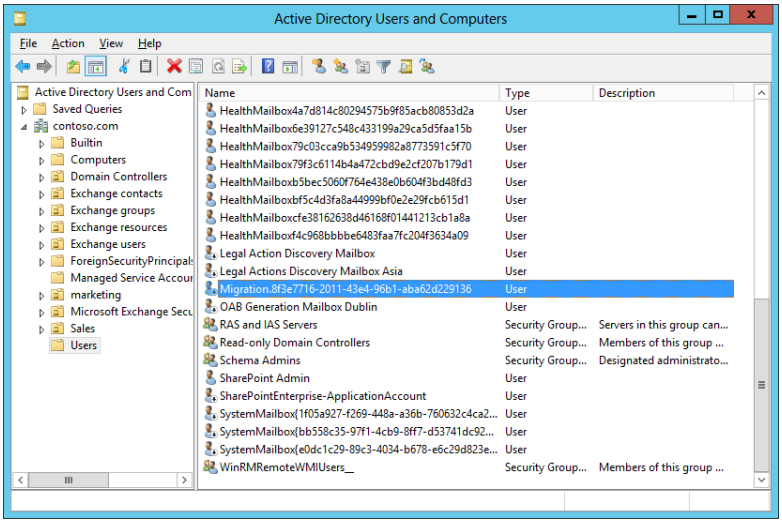


Figure 2-1 Active Directory accounts Exchange 2013 creates

In Figure 2-2, Setup checks to see whether any earlier servers are present and warns that no further Exchange 2010 servers can be installed after Setup completes.

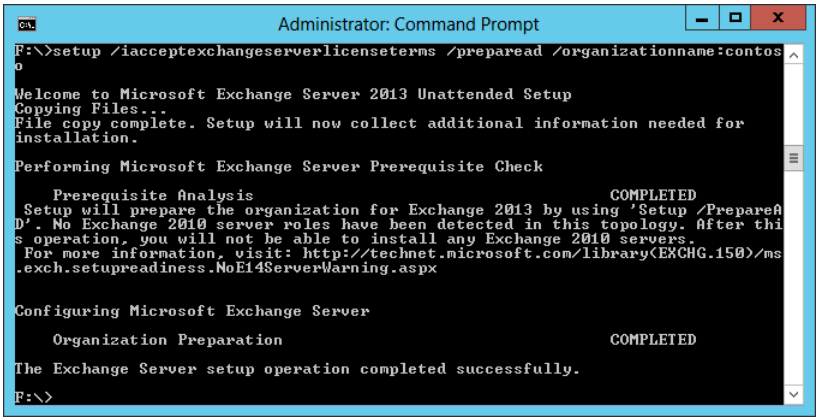


Figure 2-2 Running command-line Setup to prepare Active Directory

Deploying an Exchange 2013 server

Now that you have prepared an organization to support Exchange 2013, you can deploy some servers. The first task is to install the set of prerequisite software on the servers you plan to use for Exchange 2013. Windows 2012 servers are easier to prepare because they include many of the necessary prerequisites.

After .NET Framework 4.5, Windows Management Framework 3.0, and Windows PowerShell 3.0 are installed on a server, both the GUI and command-line versions of the Exchange 2013 installation program offer the option to configure all the remaining prerequisites thereafter automatically. Figure 2-3 shows how the GUI version of Setup offers the option to install any required Windows roles and features. You'll be prompted if any of the roles and features Setup installs requires a server restart. If a Windows hotfix or something else is missing that Setup can't install, you have the opportunity to install the missing item and then retry, or if you have to exit Setup, you can restart from the place you left off.

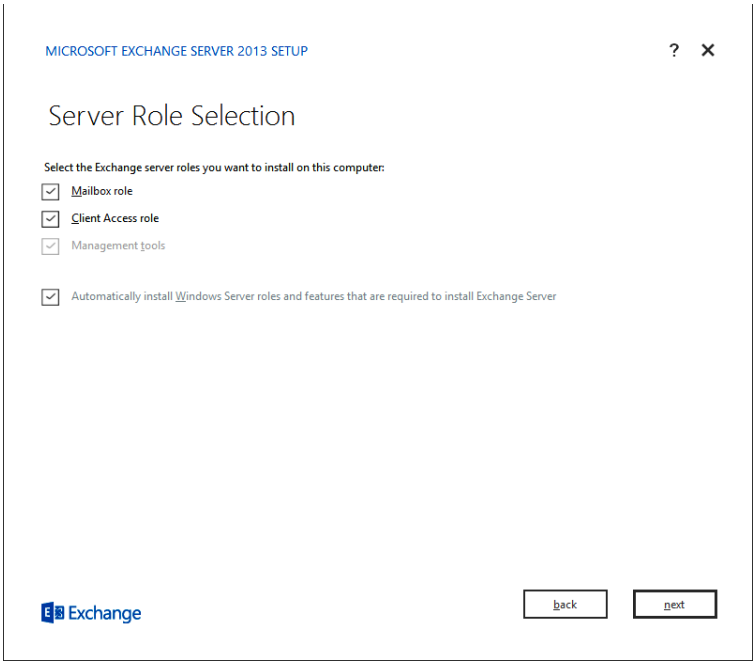


Figure 2-3 Exchange 2013 Setup offering to install required roles and features

By comparison, to run setup.com to install a typical Exchange 2013 server including any required Windows server roles and features, you type this command into a CMD session:

```
Setup.com /Mode:Install /roles: "ClientAccess, Mailbox" /InstallWindowsComponents
/IacceptExchangeServerLicenseTerms
```

It's usually better to prepare a server by installing all the necessary Windows features before moving on to tackle Exchange. You can add all the necessary features by:

1. Selecting and installing each feature with Windows Server Manager. This is the slow but steady option.
2. Running one of the installation configuration files provided with Exchange. This can be a good option when you build an automated deployment script.
3. Installing the set of required features with the Add-WindowsFeature cmdlet (Windows Server 2008 R2 SP1) or Install-WindowsFeature cmdlet (Windows Server 2012). This option takes just a few minutes to complete, including the invariable system restart.

Microsoft includes a set of XML configuration files in the \Setup\ServerRoles\Common folder on the Exchange installation kit that you can run to configure a server. These

files (such as Exchange-MBX.xml) can be used only for installations on Windows 2008 R2 SP1 servers. The commands they contain are directives for the Windows PowerShell ServerManagerCmd module that tell it how to install the required Windows features for different types of Exchange 2013 servers.

To configure the required features with one of the provided XML files, run Windows PowerShell as an administrator and use the Import-Module cmdlet to add the ServerManager module to the set of modules available in the session. You can then select the most appropriate XML file from those provided in the \Scripts directory (or copy the selected file to a suitable location for use during your installation) and input that in the ServerManagerCmd cmdlet, which then installs the features specified in the configuration file. Any features that are already configured will be ignored. Table 2-1 lists the available files on the installation kit. For example, the command to configure a typical multi-role Exchange 2013 server is:

```
ServerManagerCmd –ip \Scripts\Exchange-TYPICAL.xml
```

Table 2-1 **XML configuration files to prepare Exchange 2013 servers**

SERVER ROLES	SCRIPT
ALL (All)	ServerManagerCmd –ip \Scripts\Exchange-ALL.xml
BASE (Basic prep)	ServerManagerCmd –ip \Scripts\Exchange-BASE.xml
CADB (Admin and Mailbox)	ServerManagerCmd –ip \Scripts\Exchange-CADB.xml
ECA (Admin tools)	ServerManagerCmd –ip \Scripts\Exchange-ECA.xml
TYPICAL (Multi-role server)	ServerManagerCmd –ip \Scripts\Exchange-TYPICAL.xml

The InstallWindowsComponent.ps1 script is provided for use on Windows 2012 servers. This script ignores the XML files because it specifies the necessary Windows features in its code. The script is in the \Setup\ServerRoles\Common folder. To run the script, open a Windows PowerShell session and type the name of the script plus the type of role you want to configure. The supported values are MBX (mailbox), CAS (Client Access Server), and AdminTools (Exchange 2013 Administration tools). For example:

```
.\ "C:\Program Files\Microsoft\Exchange Server\V15\Bin\InstallWindowsComponents.ps1"
MBX, CAS, AdminTools
```

Another switch, called /ADToolsNeeded, is provided to enable you to install the Active Directory administration tools on a server that you want to use to run Setup /PrepareSchema or Setup /PrepareAD. To use this parameter, specify –ADToolsNeeded $True.

Because the script is written in Windows PowerShell, it provides an opportunity for repurposing the script by either editing it to satisfy your purposes or including it in whatever other code you use to configure servers.

A list of the full set of Windows features you must configure for each role is available at *http://technet.microsoft.com/en-us/library/bb691354(v=exchg.150).aspx*, including commands that you can cut and paste into a Windows PowerShell session to configure different types of servers (Figure 2-4). This avoids typing mistakes and ensures that you include all the necessary components. If you want, you can create your own scripts to automate the deployment process and use the Add-WindowsFeature cmdlet (for Windows 2008 R2 SP1) or Install-WindowsFeature cmdlet (for Windows 2012) to add the different components for the server roles you require. Some idea of the close relationship that exists between Exchange and Microsoft Internet Information Services (IIS) is shown by the number of IIS features that must be installed before you can deploy Exchange. These extensions support critical components such as Remote PowerShell, Outlook Web App, and the Exchange Administration Center application.

It's easy to start a Windows PowerShell session and install the necessary Windows components (Figure 2-4) by just copying the commands from the Exchange documentation on TechNet and pasting them directly into Windows PowerShell. This avoids typing mistakes and ensures that you include all the necessary components.

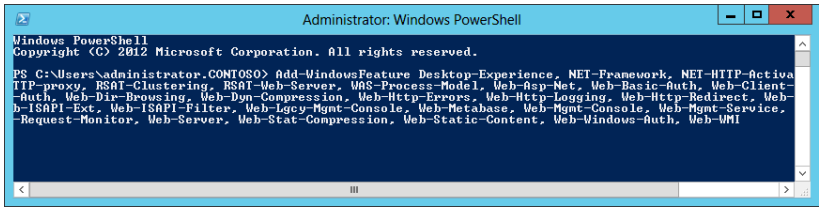


Figure 2-4 Installing Windows components with Windows PowerShell

Installing UCMA

Exchange 2013 Mailbox and multirole servers require the installation of Microsoft Unified Communications Managed API 4.0 Core Runtime 64-bit. This name is a bit of a mouthful, and it's easier to refer to it as UCMA. Unified Messaging functionality is now incorporated in the mailbox role. Even if you have no intention of ever using the Unified Messaging functionality or connecting Exchange with Lync, you still need to install UCMA.

Running Setup

After Active Directory is prepared and you have installed the necessary prerequisite software and features, you can install Exchange by using the GUI version of the setup program, or you can use the command-line version.

Exchange 2013 Setup supports four distinct modes:

- **Installation of a new server** This mode includes the option to install a new role on an an existing server.

> **Note**
>
> **If you use a cumulative update to install Exchange on a new server, setup performs a complete installation from scratch. You do not have to apply any other updates after the base installation finishes. A cumulative update can also be used to update the version of Exchange running on a server.**

- **Uninstall** Exchange 2013 cannot remove a role from a server; you have to remove the complete server.
- **B2B (build to build), which is the installation of a new build of Exchange on top of an existing server** Use this mode to apply cumulative updates; it's also used extensively during the development phase of the product when the engineering group generates a new build nightly. All server roles are upgraded at one time during a B2B run; due to the dependency among various shared components, you cannot apply a new software build to just one role on a server. The command-line switch to apply a B2B update is Setup /Mode:Upgrade.
- **Disaster recovery, which rebuilds a server based on its configuration data stored in Active Directory** This mode does not recover any of the mailbox or public folder databases that might have been present on the server; these have be restored separately. The command-line switch for this mode is Setup /Mode:RecoverServer.

> **INSIDE OUT** **Command-line vs. a GUI version setup**
>
> **The command-line version (also known as an unattended install) is most often used to execute an unattended installation, but experienced administrators who have installed Exchange many times usually prefer the brevity, speed, and simplicity of the command-line setup, whereas those who are less experienced or want a little more guidance through the various steps prefer the GUI. A tip is to perform installations (initial and subsequent upgrades) with elevated permissions because this avoids any potential problems with Windows User Account Control (UAC).**

As an example of using the command-line version of setup, this command installs both roles on a server and creates a default database called DB1 by using the specified paths for the database and transaction logs:

```
Setup.exe /Mode:Install /Roles:Mailbox, CAS /MdbName:DB1 /DbFilePath:
C:\Databases\DB1\DB1.EDB /LogFolderPath: C:\Databases\DB1
```

Command-line switches can be reduced to just the few characters required to make the switches unique. For example, the previous command can also be passed as:

```
Setup/m:i/r:cas
```

The GUI version of the installation program boots into an entry screen Microsoft internally refers to as the Can Opener, perhaps because it opens all the possibilities that exist when Exchange is correctly deployed. This version of the setup program depends on the .NET Framework and Windows PowerShell, which is why these components must be installed on a server before you can even attempt to run the program.

After initialization, the setup program takes you through multiple stages to gather information about the environment and to validate that all the necessary prerequisites exist to allow the installation to proceed before creating the files on disk to build the chosen roles for the new server. As mentioned earlier, Setup installs any missing Windows server roles and features Exchange requires if you allow it to do so.

My experience is that the first installation of Exchange 2013 on a server invariably requires a system restart to complete the process. The time required to install a brand-new server from start to finish varies according to server size and configuration, but you can certainly expect to be finished in well under an hour, especially if you install only a single role on a server. All the steps to deploy prerequisite software and then to install Exchange can be scripted for unattended installs. Installing Exchange on a virtual server (Hyper-V or VMware) follows the same course as on a physical computer, assuming that you have done the necessary work to prepare the virtualization platform for Exchange by configuring the virtual servers with appropriate CPU, disk, and memory resources according to the recommendations published by the virtualization platform vendor.

As it runs, Setup captures details of its progress in a watermark stored in the system registry to enable it to restart from the point of failure. As components are installed, Setup updates the watermark in the system registry to identify the exact state in the procedure. Setup also captures state information such as the options that you selected for the installation. Together, the state information and watermark mean that you can restart Setup from the point of failure (including a cancelled installation) without having to input all your options again.

For more information on how Exchange uses watermarks to track the progress of installations, see *http://blogs.technet.com/b/exchange/archive/2007/05/21/3402900.aspx*. Although this article is based on Exchange 2007, the same concepts are used with Exchange 2010 and Exchange 2013.

TROUBLESHOOTING

A failure occurs during setup

If a failure occurs, you will not be able to run Setup in a different mode until you complete the process that you started in the original mode. For example, if a server fails in the middle of installing a new role, you have to go back and complete or undo the installation before you can execute Setup in another mode (such as applying an upgrade).

Setup logs

If any problems occur during the installation, you can usually find the reason in the files the installation procedure places in the \ExchangeSetupLogs folder. The contents of the main installation log, ExchangeSetup.log, are quite interesting to review because you can find details there of all the commands used to install Exchange on a server, including the changes applied to Active Directory. Exchange is tremendously verbose in terms of the information written into the setup log. The log is broken into major activities, which are indicated by a long line of asterisks, and minor activities, which are indicated by a shorter line. Initialization of the installation process is a major activity; running a Windows PowerShell script to perform a single task is a minor activity.

As a glance at Figure 2-5 reveals, there's a mass of data within the setup log. Most of the data captured in the logs are not useful unless you encounter a problem during an installation and need to find out why the problem occurred. Even so, most of the information you'll need is right at the end of the log, where you'll see details of the problem the setup program experienced and the actions it took to unwind the installation. The remainder of the data in the log is unlikely to be looked at by anyone except a support specialist who's tracking down an elusive problem.

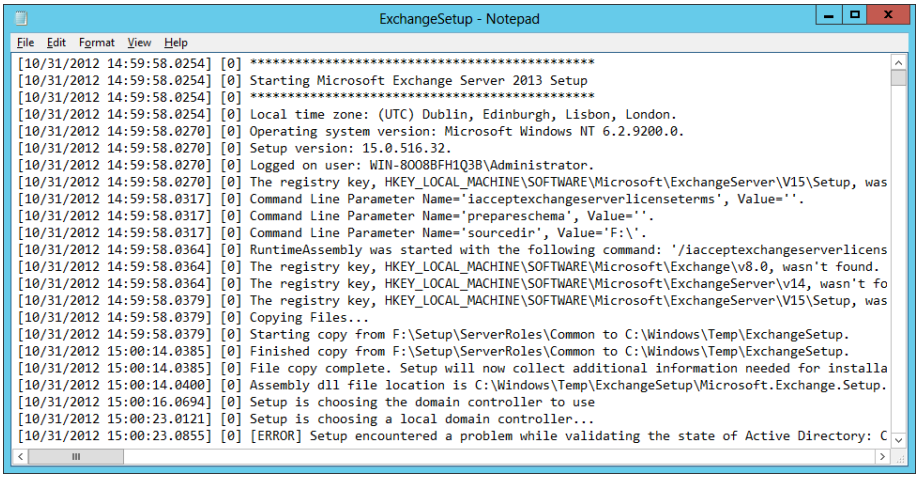


Figure 2-5 Examining the contents of ExchangeSetup.log

In addition to the setup logs, the ExchangeSetupLogs folder contains a number of Windows PowerShell scripts that the installation procedure generates to perform different steps, including the configuration of the various server roles installed on the server. The fact that Exchange uses Windows PowerShell in this manner underscores the importance of scripting to the product, as do the many examples of scripts built to perform uninstalled installations.

If problems persist and you can't install Exchange, or if Exchange doesn't work as expected after the installation, Microsoft support is likely to ask you to provide information such as the installation logs and the server's application event log to enable the support team to understand what happened and fix any problems. The team might also ask you to run the Exchange Trace Analyzer utility (ExTRA) to gather debugging information. ExTRA runs in the background to gather information about Windows and Exchange at an additional level of detail.

TIP

One small detail that occurs during an installation is that the path for executables is updated with the location of the Exchange binaries. This is a nice feature because it avoids the need for administrators to search file locations when they want to run Exchange utilities such as Eseutil.exe.

Uninstalling Exchange

No one would ever remove Exchange from a computer unless it was a test box or it was time to decommission the server after upgrading to a new release. If you need to remove Exchange, however, two methods are available to uninstall a single role, multiple roles, or a complete Exchange server:

- Run Setup.com from the command line. For example, to remove Exchange 2013 from a server, run the Setup.com /Mode:Uninstall command.
- In Control Panel, click Uninstall A Program and then select Exchange 2013 from the list of programs. This invokes the GUI setup program in uninstall mode.

Note that unlike Exchange 2010, Exchange 2013 does not support the removal of a single role from a server; you must remove Exchange completely and then reinstall the role you want to keep. When the time comes to remove roles or a complete server, the first step is to ensure that data is transferred off the server before you run Setup. Although this is not a comprehensive checklist, these steps are among those you should review before you decommission a server:

- Move mailboxes to databases on other servers (or move complete databases if the server is a member of a DAG). Disable automatic database activation on the server to prevent the DAG from activating any copies that remain on the server.
- Exclude the mailbox databases on the server from automatic provisioning so that an administrator does not inadvertently create new mailboxes in those servers. (See Chapter 5, "Mailbox management.")
- After all mailboxes, including system mailboxes such as discovery search and arbitration mailboxes, have been moved from the databases, dismount and remove the databases.
- Remove the server from its DAG if it is a member.
- Clear any outstanding move requests for databases hosted by the server.
- Disable group metrics generation and make sure that another server in the organization takes on this task.
- Remove the server from any send connectors for which it is a source.
- If the server being removed is a CAS, ensure that connectivity (external and internal) for all supported client types will continue to function.

After you relocate mailboxes, it's wise to leave the server in place for a couple of days to check that everything continues to run smoothly. You should check the event logs on the server to be decommissioned and on the servers to which you have transferred work to ensure that no problems are flagged. You could then take the server offline for a period to see whether anything breaks. Finally, after you are sure that everything is ready to proceed and no lurking problems exist, you can run Setup in removal mode.

INSIDE OUT Moving pesky arbitration mailboxes

Sometimes you'll be ready to remove a database from a server that you want to decommission when Exchange complains that the database contains one or more mailboxes. When you run the Get-Mailbox cmdlet to check, nothing is returned. This probably occurs because one or more arbitration mailboxes are held in the database, and these don't show up when Get-Mailbox is run unless you add the /Arbitration switch. The solution is to move the mailboxes before proceeding. You can do this by finding all the arbitration mailboxes in the database and piping the results into the New-MoveRequest cmdlet to move them to another database. After the moves are complete, you should be able to remove the database and uninstall the server.

```
Set-ADServerSettings –ViewEntireForest $True
Get-Mailbox –Database DB1 -Arbitration | New-MoveRequest –TargetDatabase DB2
```

If a lingering arbitration mailbox isn't the issue, Exchange might be protesting because a public folder is homed in the database. You can check with:

```
Get-Mailbox –PublicFolder – Database DB1 | Remove-Mailbox
```

If these steps don't work, and Exchange is still complaining about one or more lingering mailboxes, you can use ADSIEdit to examine the HomeMDBBL attribute (Figure 2-6) of the database to see the list of linked mailboxes that regard the database as their home. (Enable view read-only backlinks by using the Filter button when looking at the database properties to see this information.) If all this seems to be too much work, you can just remove the database by using ADSIEdit.

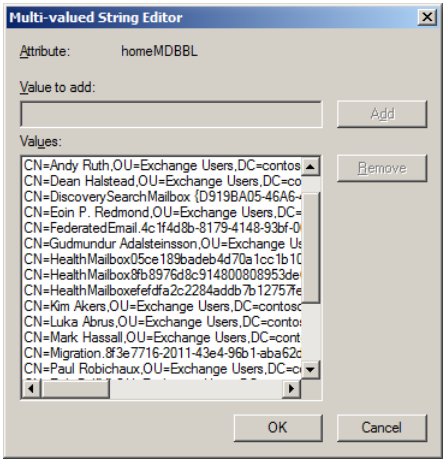


Figure 2-6 Viewing the list of mailboxes back-linked to a database

INSIDE OUT Entering the dark zone

You enter the Exchange dark zone when a problem exists that cannot be resolved with an option available through EAC, a Windows PowerShell command, or some other common utility. It might be that a software or hardware bug has caused incorrect data to be written into the Exchange configuration data in Active Directory or that an administrator has updated a program and caused a conflict with Exchange. Whatever the case, it is a bad situation when Setup won't proceed and you've tried all suggestions, including those found on the Internet.

At this point, some advocate using the ADSIEdit utility (see Chapter 1, "Introduction to Exchange 2013") in an attempt to edit or remove Active Directory that is inaccessible through other means. For example, if Setup is unable to remove Exchange from a server, and you really want to remove it from the organization, you could use ADSIEdit to remove the server object from Active Directory and then delete the computer account. This is a radical step, and it's certainly not something to attempt without a comprehensive understanding of what you're about to do, but it is a method that has often been used to remove a stubborn server.

The Exchange development group has been asked to help by providing a new setup mode to delete a server when an administrator is absolutely sure that this is what he wants to do. Unhappily, they still haven't given us a command something like this:

```
Setup /m:Remove /YesIReallyMeanIt /TakeNoExcuses /BlowitAwayServerName
```

Repairing Exchange

The Exchange 2013 setup program does not support a repair mode; you cannot run Setup in a repair mode to have it scan a server for any missing files, registry keys, Active Directory objects, or other elements that might be affecting the ability of the server to run properly. If a server is not functioning correctly and nothing you do seems to make any difference (and no help is gained by searching the Internet), you might be forced to uninstall all the roles from the server to remove Exchange and then reinstall the server from the beginning. This is not so bad; experience with broken software demonstrates that a "flatten and rebuild" approach usually delivers a more robust solution than attempting to repair individual flaws that might be hiding other problems that only appear after you repair the apparent problem.

An alternative approach is to wait for Microsoft to provide a fix in a cumulative update and install it as an upgrade to a more recent build. This will usually fix any problems caused by a missing component, if only because a build-to-build upgrade includes code that will create objects such as a missing registry key or permission that it detects as the server is upgraded.

Recovering a failed server

If you're unfortunate enough to suffer a catastrophic hardware failure that renders a computer completely unusable, you can recover the configuration for the server from Active Directory to rebuild Exchange. To do this, you must first provide new hardware that runs the same operating system as the failed server and has the same drive letters available for the databases that the server supports. The new server should be as powerful as the failed computer in terms of its ability to support user load. Because new hardware usually has improved performance, it shouldn't be a problem to replace the failed computer with new hardware, but replacing it with older hardware might be problematic if the configuration of that computer isn't as capable and is therefore less likely to support the same load.

After the new hardware is installed, you must do the following:

1. Reset the Active Directory computer account for the failed server.
2. Install and configure Windows with the prerequisite roles and features, including any hotfixes Exchange requires; the setup program cannot install any required software when it runs in recovery mode.
3. Ensure that network connectivity works properly and that the new computer is configured with the correct IP addresses and other settings.
4. Join the replacement computer to the domain hosting Exchange with the same name as the failed server.
5. Authenticate the Windows server license.

You can then run Setup.com with the /m:recoverserver command. In this mode, the installation procedure reads the details of the server configuration from Active Directory and reconstructs an exact replica of the failed server. Exchange keeps details of the software versions installed on a computer in the system registry, and the setup program will only recover a server to the same version of Exchange that was installed on it before it failed. You cannot recover and upgrade at the same time; for instance, it's not possible to recover a failed Exchange 2013 server by using an Exchange 2013 SP1 software kit, so be sure you keep a share or DVD with the version of Exchange that you've actually installed.

After the recovery is complete, all the server roles defined in Active Directory will be operational on the replacement server. In the case of a failed mailbox server, you must still restore databases to their location on disk. The up-to-date database and transaction logs might be accessible from the disks the failed server used. If not, you'll have to recover from backup and replay whatever transaction logs are available to bring the databases as up to date as possible. Some data loss is inevitable in this case. In the case of CAS servers, you might also have to restore configuration files that are not included in Active Directory. For example, if you update the Mailbox Replication Service (MRS) configuration file on mailbox servers for some reason, you will have to:

- Copy the updated configuration file from another mailbox server (if available) in the organization. All configuration updates should be applied consistently across the organization.
- Recover the configuration file from another backup.
- Manually apply the update again. This assumes that you have details of the edits that should be applied to the configuration file, which underlines the need for careful documentation of these types of changes.

Special steps are required for failed servers that are members of a DAG. The installation procedure won't allow you to run /m:recoverserver if it detects that the server is a member of a DAG. To proceed, you must remove database copies and then evict the failed server from the DAG before you can run the steps described earlier. After the server is restored to full health, you can bring it back into the DAG and create new database copies.

See *http://technet.microsoft.com/en-us/library/dd638206(v=exchg.150).aspx* for more information about the steps required to restore a failed DAG member server.

Remember that the operation you've just performed is a recovery of Exchange based on information in Active Directory. Any information or configuration setting that is stored outside Active Directory, such as customizations made to configuration files, must be restored separately.

INSIDE OUT Delegated setup

Exchange 2013 supports the ability for an administrator to delegate setup activity for a server to a nonprivileged user. This can be done for any server after the first Exchange server is installed in an organization. In outline, you do this by running the Exchange setup program in a special mode that creates the server object in Active Directory and prepares it for completion at some point in the future. You then delegate the completion task to another user, who can then run the Exchange setup program to complete the installation.

The first task is to run Setup to create the server object in Active Directory and prepare the server for future completion.

```
Setup.com /NewProvisionedServer:ExServer7.contoso.com
/IAcceptExchangeServerLicenseTerms
```

The target server must exist and be properly prepared for Exchange to be installed. After a successful setup run, you can check Active Directory to see whether the new server is listed with the other Exchange servers.

To complete server provisioning, the user's account must be a member of the Delegated Setup role group. The Exchange role-based access control system and the various role groups are defined in later chapters. For now, it's sufficient to know that you have to run an Exchange Management Shell (EMS) command to add the mailbox to which you want to delegate the server completion task. You can delegate roles to mailboxes or universal security groups only, so the user must have a mailbox or be a member of a universal security group before you can delegate a role. For example, to assign the task of performing the delegated setup to the user who owns the mailbox named Redmond, you open EMS and type this command:

```
Add-RoleGroupMember –Identity 'Delegated Setup' –Member 'Redmond'
```

Afterward, this user can run Setup on the provisioned server to complete the installation, providing that she holds local administrator permission for the server.

Versions, cumulative updates, and service packs

Exchange 2007 marked a new approach to support, delivered in the form of updated code. Previous versions of Exchange focused on regular service packs, which became very large updates similar in many respects to the release of a new version of the product. The current approach divides updates into two kinds of releases for which you can plan:

- **Service packs (SP)** Microsoft releases a new service pack for Exchange on average once a year. A service pack typically includes new functionality as well as fixes and is intended to mark a distinct point in a product's development. In addition, a service pack provides customers with a release that they can target for installation at a suitable point in their maintenance cycle. At the time of writing, Microsoft has indicated that it will release a service pack for Exchange 2013, but code is not yet available.

- **Cumulative updates (CU)** Cumulative updates appear roughly every three months and are cumulative in that CU3 includes all the fixes issued in CU1 and CU2. Therefore, you can apply the latest CU to update a product with all the current released fixes or use the latest CU to install Exchange 2013 on a new server. Cumulative updates are issued on the web and announced on the Microsoft EHLO blog. Microsoft might also make some hotfixes or security updates that pertain to specific situations available separately, and it's worth checking with Microsoft about any important hotfixes that exist for functionality that is important to your deployment before you plan to deploy a cumulative update.

It is sometimes difficult to understand the differences among a cumulative update, a service pack, and a new version of a Microsoft server product. We'll tease this issue out in this section.

INSIDE OUT **A best practice for server software levels**

It is best practice to ensure that all servers in an organization run at the same software level, including roll-up releases. There is no good logic suggesting that servers should run at different software levels for any sustained period. This is especially important when the Active Directory schema is updated by a service pack to introduce new properties for objects Exchange uses. In some cases, feature changes or bug fixes in service packs require you to upgrade all servers to the same service pack level; Microsoft does a good job of documenting these requirements, and you should follow them.

Cumulative updates

Software maintenance is an ongoing process. In the case of Exchange, the outcome is a regular update that represents the best current state of the software as updated with whatever fixes are known at the time the update is produced. From Exchange 2013 onward, Microsoft operates a policy of running the same code base in both Exchange Online and on-premises deployments. Microsoft deploys updates within Exchange Online on a

frequent basis to fulfill one of the promises of cloud platforms, which stipulates that users have access to evergreen technology that is continually updated and refreshed.

On-premises customers gain access to the code through a complete software build called a cumulative update (CU). Each CU represents the most recent update for Exchange 2013, including all the fixes for known software issues at the time the CU was built. A CU can be used to bring an existing server up to the latest software version or to install Exchange 2013 from scratch on a brand-new server.

Gaining fast access to new functionality can be extremely valuable, especially if the new feature addresses a long-held need for a company. The downside is that changing the way a product works can affect users, sometimes for the worse. Microsoft simply reacts to the competitive demands of an online world when it slipstreams new features into products outside major releases. Users might not realize that an update has happened unless the new software reveals a new feature in the user interface, but many changes might have occurred in the way Exchange functions.

For example, Microsoft released Exchange 2010 SP2 RU4 (roll-up update number 4 for Exchange 2010 service pack 2) in August 2012 and changed the way the Managed Folder Assistant (MFA) processes calendar and task items in user mailboxes. Up to that point, an administrator could create a retention policy that included a default retention tag that, for example, caused all items that did not have a more explicit tag assigned to them to be moved into the Deleted Items folder after 120 days. The MFA did not evaluate calendar and task items and thus did not apply the action assigned by the default retention tag. Exchange 2010 SP2 RU4 updated the MFA to enable it to evaluate calendar and task items; any mailbox that came under the control of a retention policy that included a default retention tag would have items in the calendar and task folders processed the next time the MFA ran following the installation of Exchange 2010 SP2 RU4. The upshot was that MFA could remove items from mailboxes in an unexpected manner.

The servicing model used for Exchange 2013 means that Microsoft issues cumulative updates on a quarterly basis. Note that "quarter" does not mean "every three months" because it is quite possible that Microsoft will issue an update in July (the first month of the third quarter) and follow it up with the next update in December (the last month of the fourth quarter). The different intervals allow Microsoft to test an update in production by deploying it into its Office 365 datacenters, which means that the code running in Exchange Online is usually a couple of months ahead of what is available to on-premises customers. Assuming that all goes well, the update is then released to on-premises customers with the inherent promise that because it works for Office 365, customers should be confident that the update will deliver acceptable quality and performance when deployed on-premises. In addition, customers who run hybrid deployments will know that after they deploy the latest cumulative update on-premises, essentially they then run the same software version for the on-premises and cloud portions of their deployment.

The new servicing model encourages customers to install updates as they are released by Microsoft. If a customer reports a bug that is fixed in a later release to the software that they run, they must install the latest update before Microsoft will investigate a problem further. If a bug is discovered in an update, Microsoft will fix the problem and release the fix as part of the next cumulative update with the aim being to reduce the number of individual hotfixes released to customers. Put another way, given the regularity of the update release cycle, customers should not have to wait long before they receive an update that contains fixes for any problems that they report. Best practice is to deploy the same update on all servers within an organization because it doesn't make sense to have different versions in use in different places. In particular, make sure that all the servers in a DAG are brought up to the same software version as soon as practically possible.

Because Exchange 2013 cumulative updates appear on a regular basis, administrators need to be aware that functionality can change and be prepared to test the updates as they appear with this in mind to assess whether changes will affect the way Exchange is used within their company. In addition, administrators must be sure that any third-party software that operates alongside Exchange 2013 is verified to work properly with a CU before it is deployed into production. Note that the installation of a cumulative update will wipe any trace of customized configuration files (for example, to customize Outlook Web App), so some care has to be taken to preserve and reapply these kinds of changes when you deploy these updates.

Given that cumulative updates allow customers to keep their servers updated more frequently than having to wait for an annual service pack, the role of a service pack is different for Exchange 2013 than it was for previous releases. With Exchange 2007 and Exchange 2010, you had no option but to wait for a service pack to appear before you could access new functionality or see new documentation. Now we have quarterly cumulative updates that include bug fixes, new functionality, and documentation updates. The new servicing model is excellent for customers who can cope with such a rapid release cadence, but it can pose very real difficulties for customers who operate highly regimented environments with strict limits on when software updates can be applied. In addition, the need to validate new releases against realistic test environments that mimic the essential characteristics of the operational systems, including alongside third-party and home-grown applications, is often a demanding exercise, especially when considered against the need to provide a service to end users, dealing with vacation, training, other absences of personnel, and other day-to-day problems. This is where a service pack proves valuable because customers can plan for the release of a service pack and incorporate the need for testing and validation into their regular maintenance cycle.

Microsoft has indicated that it will release a service pack for Exchange 2013 in the future. When that occurs, it will be interesting to compare the contents of the service pack and assess its true value. In the interim, the story around quarterly cumulative updates continues to unfold and customers continue to cope with the accelerated cadence.

Testing cumulative updates

Make sure that you test cumulative updates before you deploy them into production because sometimes, in previous versions of Exchange, problems occurred during or after the installation of the update. Sometimes the fault is not exclusively with the Exchange developers and is exposed when customers deploy the new software, as when an Internet Explorer problem caused an issue for the Exchange 2010 Management Console. Predeployment testing is an absolute necessity to detect problems that might sneak past the large Microsoft array of automated tests. It's also important to install cumulative updates with accounts that have full administrator permission to ensure that all files are properly updated.

Version numbers

Exchange build numbers tell you the exact version of the software. The format used is:

<product version>.<service pack number>.<major build number>.<minor build number>

A version number such as 15.00.0516.032 means:

- **Product version 15** Exchange 2013 is a component of the Office 15 wave of products. Exchange 2007 was version 12, and Exchange 2010 was version 14. There was no version 13.
- **Service pack 00** No service packs have been applied, so you know that this is the original (RTM) version of the product.
- **Major build 516** Microsoft builds new versions of the product on an almost daily basis, and the build number increments sequentially, so this is build 516, the build used for the RTM version of Exchange 2013. Build numbers differ from version to version, so there is no sequential update from the builds used for Exchange 2010 to those for Exchange 2013.
- **Minor build 032** Thirty-two minor updates have been applied to major build 516. The number of minor updates is accounted by the way Microsoft tweaked the last major build of Exchange 2013 with a series of minor fixes as it drove to complete the product.

If you view server details through the Server node of EAC, you can see the version number of every server in the organization. This is broadly equivalent to executing the following EMS command:

```
Get-ExchangeServer | Select Name, ServerRole, AdminDisplayVersion
```

For administrative purposes, you might want to capture the software edition that's installed on each server and output this information to a comma-separated-value (CSV)–formatted file that you can later analyze with Microsoft Excel, Microsoft Access, or another program that is capable of reading CSV data:

```
Get-ExchangeServer | Select Name, Edition, AdminDisplayVersion, ServerRole | Export-
CSV C:\TEMP\Servers.CSV
```

The data exported should look similar to this:

```
#TYPE Selected.Microsoft.Exchange.Data.Directory.Management.ExchangeServer
"Name","Edition","AdminDisplayVersion","ServerRole"
"EXSERVER1","Enterprise","Version 15.0 (Build 516.32)","Mailbox, ClientAccess"
"EXSERVER2","StandardEvaluation","Version 15.0 (Build 516.32)","Mailbox,
ClientAccess"
```

Version 516.32 is the RTM release of Exchange 2013. The development team released this version to manufacturing on October 11, 2012. The first production customer deployments were already in progress before this time as part of the Microsoft Technology Adoption Program. General deployments in the wider customer base began in March 2013 after the release of Exchange 2013 RTM CU1 and Exchange 2010 SP3 and Exchange 2007 SP3 RU10, the versions needed to interoperate with Exchange 2013. The code build number for Exchange 2013 RTM CU1 is 620.29 while the value for Exchange 2013 RTM CU2 is 712.22.

Exchange also writes information about version numbers into the system registry (Figure 2-7). You can retrieve this information with some Windows PowerShell code to identify the major and minor builds that are installed on a server. In this instance, MsiBuildMajor is 516, and MsiBuildMinor is 32, so this server runs version 516.32, or the RTM release of Exchange 2013. The following example shows Windows PowerShell code you can use to fetch the Exchange version information from the system registry:

```
$RegExSetup = 'Software\\Microsoft\\ExchangeServer\\v15\\Setup'
$Server = (Get-Content env:ComputerName)
$Registry = [Microsoft.Win32.RegistryKey]::OpenRemoteBaseKey('LocalMachine', $Server)

$RegKey = $Registry.OpenSubKey($RegExSetup)
$V1 = "MsiBuildMajor"
$V2 = "MsiBuildMinor"
$BuildMajor = ($RegKey.GetValue($V1))
$BuildMinor = ($RegKey.GetValue($V2))
Write-Host $Server "runs version" $BuildMajor "." $BuildMinor
```

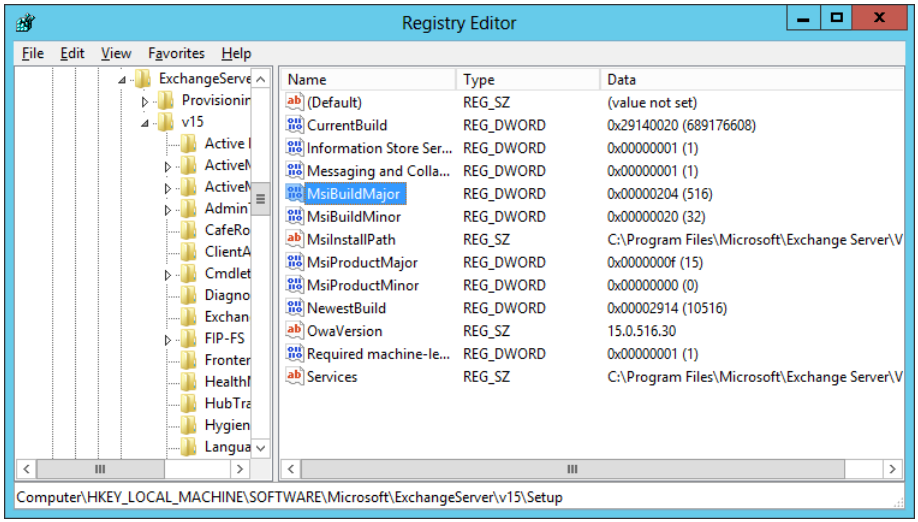


Figure 2-7 Installation information in the system registry

Exchange also stores information in the system registry about updates that are installed on a server. These data are held in a series of entries (one for each update) at HKLM \Software\Microsoft\Windows\CurrentVersion\Installer\UserData\S-1-5-18\Products \AE1D439464EB1B8488741FFA028E291C\Patches. The subkeys that provide information about each roll-up update are called DisplayName (the name of the update) and InstallDate (the date it was installed on a server).

Object versions

Every object belonging to Exchange is also assigned a version number, which is used to determine the minimum version of the management tools that can manage the object. For example, if you use the Get-Mailbox cmdlet through the Exchange Management Shell to view the properties of a mailbox object, you should see the version reported as shown here:

```
Get-Mailbox -Identity "Redmond, Tony" | Select ExchangeVersion
```

```
ExchangeVersion
0.20 (15.0.0.0)
```

You can see that version 15 is listed, so you know that this object should be managed using the Exchange 2013 management tools (EAC and EMS). You can also see that the minor version is 0, and you know that the major build number for the RTM release is higher (516), meaning any build of Exchange 2013 can manage mailbox objects. Mailboxes certainly exist in previous versions of Exchange, and the fact that the version number specifies

that Exchange 2013 must be used indicates that mailbox objects have been upgraded for Exchange 2013. This is true because new attributes have been added to mailbox objects for features such as in-place holds. If you look at the version numbers for other objects that are introduced in Exchange 2013, such as data loss protection policies, you'll see similar version numbers. By comparison, if you look at a send connector and do the same thing, you see a different version number:

```
Get-SendConnector | Select ExchangeVersion
```

```
ExchangeVersion
(8.0.535.0)
```

In this context, version 8 indicates Exchange 2007. (This is despite the fact that Exchange 2007 was codenamed Exchange 12, just as Exchange 15 became Exchange 2013.) As it happens, build 535 for Exchange 2007 was the original RTM release, so this version indicates that any management tool issued with Exchange 2007 onward can manage the object. But in any case, the situation can be boiled down to a simple rule: when in doubt, use the same version of the Exchange management tools as indicated by the ExchangeVersion reported by the object.

An excellent article on the Exchange development team blog (*http://msexchangeteam.com/archive/2010/01/08/453722.aspx*) gives guidance on how to determine which tools to use to manage different Exchange objects.

Reporting licenses

When you install a server, it is in an unlicensed or trial state. In other words, you've installed Exchange to kick the tires and see what the server can do. Servers in trial status are not eligible to receive support from Microsoft and are limited to the functionality available in the Standard edition of Exchange 2013. The major consequence of this is that although you can create a DAG, you can mount only five databases on a server. To move a server to full functionality, you have to install a valid license.

You can view the current licensing situation for a server by opening the Servers section of EAC and reviewing the list of servers for the organization. Unlike Exchange Management Console (EMC), EAC does not use different icons to mark licensed and unlicensed servers. Instead, it indicates the current license status for a server in the action pane (Figure 2-8).

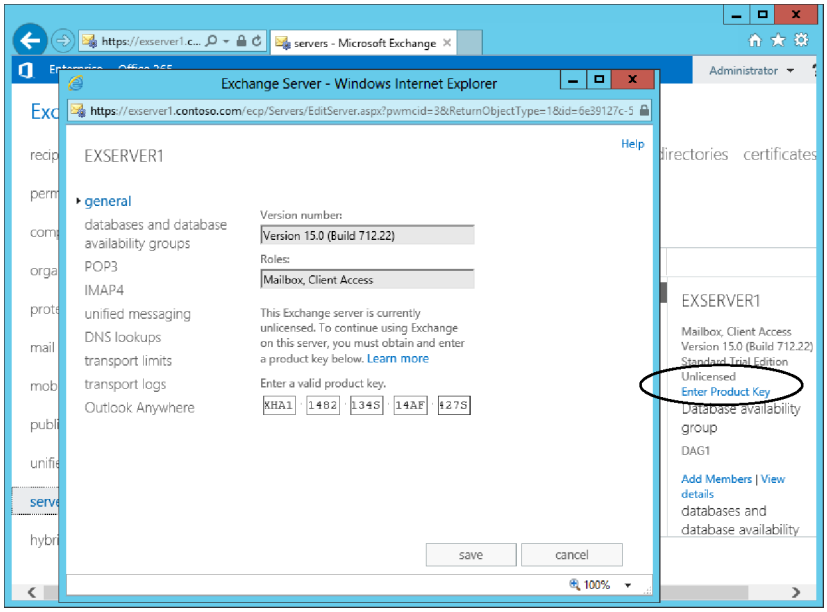


Figure 2-8 Viewing license status for a server

Scrolling through a list of servers is a reasonable way to proceed when you have only a small number of servers to manage. It is much more efficient to identify any unlicensed servers in the organization using EMS by running the following command:

```
Get-ExchangeServer | Where {$_.IsExchangeTrialEdition -eq $True} | Format-Table Name,
RemainingTrialPeriod – AutoSize
Name       RemainingTrialPeriod
----       --------------------
EXSERVER2 177.10:40:53.0863341
```

In this instance, you can see that one unlicensed server exists in the organization and that it was installed just a few days ago (177 days remain of its 180-day trial period). To resolve this problem, select the unlicensed server in EAC and then select the Enter Product Key option in the action pane. This displays the general properties of the server and where you can enter a valid product key. Alternatively, you can input the product key through EMS by using a command like this:

```
Set-ExchangeServer -Identity 'ExServer6' -ProductKey '25-Char-Product-Key-Value'
```

You do not need to be connected to the Internet to enable Exchange to validate the entered key. If the key is acceptable, Exchange generates a product identifier for the server. You can discover the product identifier for a server by looking at its properties with EAC or by running this command:

```
Get-ExchangeServer -Identity ExServer1 | Select Productid

ProductID
---------
02064-004-0073112-02145
```

The change in licensing status will not be active until the next time the Information Store service is restarted. You can enter a new product key for a licensed server if you want to upgrade it from Exchange 2013 Standard edition to Exchange 2013 Enterprise edition. However, you cannot downgrade a server licensed to run the Enterprise edition to run the Standard edition.

TIP

Exchange does not terminate abruptly after the warning period elapses. Its functionality continues as before, and you can even upgrade a server that has exceeded its trial period to Exchange 2013. However, you might be guilty of running unlicensed software, and Exchange will continue to inform you of this until you enter the required license. This is not a great situation because it exposes your company to large fines in most jurisdictions, so it's best not to go there unless the servers are used in labs or for other test purposes.

Apart from server licenses, you also need Client Access Licenses (CALs) for every client device that connects to Exchange. Two forms of CALs are available—standard and enterprise. The licenses are cumulative. The enterprise CAL adds to the functionality already licensed through the standard CAL. Table 2-2 lists the different areas of functionality that are licensed through the two types of CALs.

TABLE 2-2 Determining the requirements for enterprise CALs

Feature set	Standard CAL required	Enterprise CAL required
Standard email functionality with Outlook, Outlook Web App, or other clients, including calendar, journal, notes, and contacts and apps for Outlook Web App	X	
Public folders	X	
Advanced ActiveSync policies for mobile devices		X
Site mailboxes	X	
Journaling on a database basis	X	
Journaling on a selective basis (per-user or other criteria)		X

Feature set	Standard CAL required	Enterprise CAL required
Unified Messaging		X
Retention policies (if configured with a personal tag)		X
Personal archive		X
Data Loss Prevention		X
In-place Hold		X
Information Protection, including journal and transport rule decryption, Outlook Protection Rules, and search of protected content		X

INSIDE OUT **A free hybrid license with CU2 and later**

With Exchange 2013 RTM CU2 and onward, a special free hybrid license is available from Microsoft to support situations where an Exchange 2013 server has to be deployed to support hybrid connectivity between an on-premises organization and Office 365. You have to request the hybrid key from Microsoft Support.

Security groups and accounts Exchange creates

Alongside the schema changes made to support server functionality, the Exchange installation procedure adds a number of security groups and mail-enabled accounts to Active Directory that Exchange uses for different purposes. Table 2-3 lists the security groups Exchange creates in Active Directory and their purpose. Most are universal security groups (USGs), which are created in a new organizational unit (OU) called Microsoft Exchange Security Groups in the root domain of the forest. As indicated in Table 2-3, many are used to hold the security principals required to enable users to perform the tasks assigned to role groups defined in the Exchange role-based access control mechanism, though you shouldn't change the membership of these groups directly; use the role-based access control (RBAC) tools to do so.

TABLE 2-3 Exchange USGs created in Active Directory

Group	Purpose	RBAC
Compliance Management	Administrative role group for users who need to work with Exchange compliance features such as data loss prevention policies and retention policies.	Yes
Delegated Setup	Administrative role group for accounts that complete the setup for an Exchange server after it has been provisioned by an organization administrator.	Yes
Discovery Management	Administrative role group for accounts that perform discovery searches and retrieval of discovered information from user accounts.	Yes
Exchange Servers	Every Exchange server in the organization is a member of this group, which is used to allow the servers to authenticate against one another mutually.	No
Exchange Trusted Subsystem	A highly privileged group that has read-write access to every object in the Exchange organization. Unless the Active Directory split permissions model is used, this group is a member of the Administrators local group and the Exchange Windows Permissions group and can create and update Active Directory objects. Exchange system components such as the System Attendant and Information Store processes use the access granted through this group to perform tasks. Removal of this group from other groups or access control lists (ACLs) will invariably cause Exchange to malfunction. If you deploy Exchange 2007 SP3, you'll find that the Exchange Trusted Subsystem group is created in the Microsoft Exchange Security Groups OU to support interoperability among Exchange 2007, Exchange 2010, and Exchange 2013.	No
Exchange Windows Permissions	A privileged group Exchange uses to manipulate Windows permissions and Active Directory objects.	No
Exchange LegacyInterOp	A group Exchange uses to perform privileged operations with older servers in the same organization.	No
Help Desk	Administrative role group that enables users to perform common help desk tasks such as modifying options on behalf of a user.	Yes
Hygiene Management	Administrative role group to manage tasks to cleanse the email stream such as anti-spam and antivirus.	Yes

Organization Management	Administrative role group for accounts that have permission to manage objects at an organization level such as to create new connectors or transport rules. Members have full control over any object in the Exchange container in Active Directory. The account used to install the first Exchange server in an organization automatically belongs to this group.	Yes
Public Folder Management	Administrative role group for accounts that have permission to manage (add, modify, delete) public folders and their settings (quotas, expiration periods, and so on). Although these accounts can mail-enable a public folder, they cannot change mail-enabled attributes for a folder (such as an email address) because this requires recipient management status.	Yes
Recipient Management	Administrative role group for accounts that have permission to manage mailboxes, distribution groups, and other recipient types.	Yes
Records Management	Administrative role group for accounts that need to perform records management tasks such as the definition of a managed folder mailbox policy.	Yes
Server Management	Administrative role group for accounts that need to perform server-specific management tasks such as monitoring messaging queues, configuring virtual directories, and so on.	Yes
UM Management	Administrative role group for accounts that can manage Unified Messaging.	Yes
View-Only Organization Management	Administrative role group for accounts that have permission to view details of the Exchange configuration.	Yes

In addition to the groups listed in Table 2-3, a group called Exchange Install Domain Servers is created in each domain that supports an Exchange server. This group is created during the installation of the first Exchange server in a domain and is added to the membership of the Exchange Servers USG in the Microsoft Exchange Security Groups OU. The setup program uses it to ensure that it can complete even if Active Directory replication has not yet populated details of the USG in the local domain. It is therefore best to wait for replication to finish before you attempt to install a server.

Exchange creates a small number of mail-enabled accounts in the default Users OU of the root domain in the Active Directory forest. Microsoft could have created these accounts in a special OU but preferred to exploit the fact that you can usually guarantee that the Users OU is always available unless it has been removed for some reason by an administrator, so using it this way eliminated the need to create an additional OU. If you want, you can move these accounts to another OU. The accounts are named in such a way that it's obvious that they are intended for system rather than human use:

- ***SystemMailbox{e0dc1c29-89c3-4034-b678-e6c29d823ed9}*** Used to hold metadata about discovery searches. The display name for this account is Microsoft Exchange. This mailbox is also used to hold administrator audit log reports. See Chapter 11, "Compliance," for more information on how to configure the auditing of administrator actions. This mailbox is also used to hold UM metadata.

- ***SystemMailbox{1f05a927-xxxx-xxxx-xxxx-xxxxxxxxxxxx}*** Where x is a randomly assigned number, this account is for moderating messages, for storing details of move requests that are currently in progress, and for testing mail flow and Messaging Application Programming Interface (MAPI) connectivity. The display name for this account is Microsoft Exchange Approval Assistant.

- ***FederatedEmail.4c1f4d8b-8179-4148-93bf-00a95fa1e042*** Used for federation operations among different Exchange organizations and for Rights Management operations. This account also has a display name of Microsoft Exchange Approval Assistant.

- ***DiscoverySearchMailbox{D919BA05-46A6-415f-80AD-7E09334BB852}*** Used for storing the results of discovery searches so that administrators can retrieve these items. Exchange creates the default discovery mailbox with the name just shown. The mailbox is created in the mailbox database of the first Exchange 2010 server in the organization. You can create other discovery mailboxes to use with searches. For example, in a large, distributed organization, you might have separate discovery mailboxes for every country/region.

- ***Exchange Online-ApplicationAccount*** **(new for Exchange 2013)** Used when a hybrid connection is created between Exchange 2013 and Exchange Online.

- ***Migration.8f3e7716-2011-43e4-96b1-aba62d229136*** **(new for Exchange 2013)** Used by the Migration Service to hold details of the mailboxes that are being moved in migration batches.

- ***SystemMailbox{bb558c35-97f1-4cb9-8ff7-d53741dc928c}*** **(new for Exchange 2013)** The default organization mailbox Exchange uses to hold files for the Offline Address Book as they are generated.

CAUTION!

Apart from accessing the discovery mailbox to access the results of discovery searches (see Chapter 11), you should not attempt to log on to these accounts or use the mailboxes for anything other than management purposes. If you need to remove a server or database that hosts any of these mailboxes, you should move the mailboxes to another database first.

The Exchange Trusted Subsystem USG

The Exchange Trusted Subsystem USG is interesting from many perspectives. It is a highly permissioned USG that allows Exchange services read and write access to any object Exchange owns. Usually, you won't need to touch the Exchange Trusted Subsystem group, but there are some known circumstances in which you might need to add it explicitly to an object. The most common is when you configure the file share witness for a DAG to be on a server that does not have Exchange 2013 installed on it. In this scenario, you need to add the Exchange Trusted Subsystem group to the local Administrators group on the server that holds the file share witness to enable that server to participate within the DAG.

Namespace planning

Namespace planning describes the activity companies go through to determine the public connectivity points that will be offered to clients that wish to access email services through the Internet and how those connections will be secured. The most basic situation occurs when a single connectivity point such as mail.contoso.com is used to access all Exchange services, including ActiveSync for mobile clients, Outlook Anywhere, Outlook Web App and Exchange Administration Center for browser access, and Exchange Web Services. Because Exchange uses Secure Socket Layer (SSL)-based client–server connections across the Internet, you also need to install certificates to prove that the services are offered by the company and not by an imposter. *Exchange Server 2013 Inside Out: Connectivity, Clients, and UM* contains a more comprehensive description of how the Exchange 2013 CAS uses certificates.

Self-signed certificates

When you install Exchange 2013 on a server, Exchange automatically creates three self-signed certificates (Figure 2-9) to enable the server to communicate with other servers and clients within the organization. However, only other Exchange servers in the organization automatically trust the self-signed certificates presented by a server. Self-signed certificates that Exchange creates include the server name and its FDQN in the Subject Alternate Name field and last for five years. The biggest advantage of a self-signed certificate is that it is free, but aside from providing the basis for testing inside an organization, that is its only advantage.

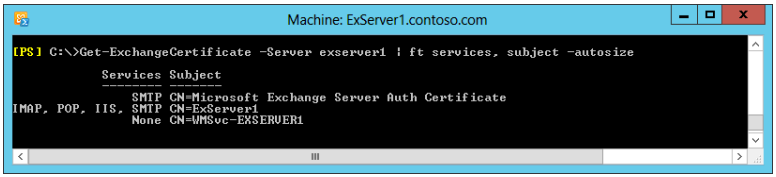


Figure 2-9 Viewing the self-signed certificates on an Exchange 2013 server

Some companies have deployed self-signed certificates into production because they want to avoid the cost of buying certificates signed by an external vendor. This is a regrettably short-sighted step because saving the relatively low cost of the certificate cannot justify the security exposure that a self-signed certificate can create. Indeed, an Australian researcher reported in July 2012 how Android and Apple iOS devices might fall victim to a man-in-the-middle attack by using ActiveSync when self-signed certificates were used. The attack could result in wiping out device contents. All Exchange 2013 deployments that offer services through the Internet should buy and install the certificates necessary to secure the services.

Security alert: Limitations with self-signed certificates

A self-signed certificate can be used with all Exchange communications within the firewall and supports some external connections (Outlook Web App and ActiveSync) after the certificate has been copied into the trusted root certificate store on the client computer. Some mobile devices don't permit this, so you can't use these devices with self-signed certificates. However, you cannot use these certificates to secure Outlook Anywhere connections. Microsoft Outlook 2007 accepts self-signed certificates used for internal communications quietly in that it doesn't display any errors when they connect to an Exchange 2010 server. Outlook 2010 and Outlook 2013 are more cautious and signal a potential problem to a user (Figure 2-10).

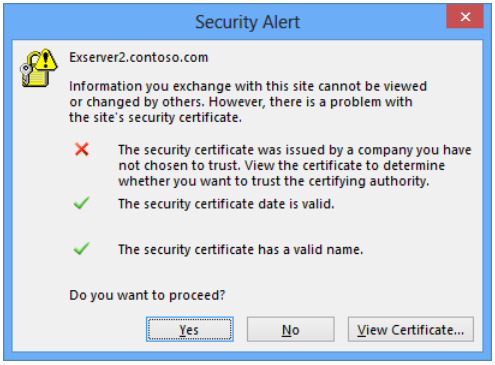


Figure 2-10 Outlook 2013 flagging a problem with a self-signed certificate

In most cases, because of the limitations imposed on the supported functionality and the difficulty of managing the certificates over time, self-signed certificates are used for test deployments or deployments that do not support external client connectivity only.

Using the Exchange 2013 CAS to handle connections

If you run Exchange 2007 or Exchange 2010 today and want to introduce Exchange 2013 at some point in the future, you must bring Exchange 2013 CAS servers into operation. The recommended approach is to have the Exchange 2013 CAS take over the namespace (for example, mail.contoso.com) for the organization so that it then works with its Exchange 2007 or Exchange 2010 counterparts to route incoming client connections to the correct mailbox servers. The Exchange 2007 and Exchange 2010 servers must first be updated to enable them to coexist with Exchange 2013 inside the organization.

In many companies, the usual approach to handle incoming connections is to deploy a load balancer in front of the CAS. (For the purposes of this section, "CAS" means either an individual CAS or a member of a CAS array in an Internet-facing site.) It doesn't matter whether the load balancer operates at layer 4 (now supported by Exchange 2013) or layer 7. What does matter is that the load balancer will pass incoming client connections to the CAS.

At this point, the CAS has to decide where to direct the connection. The location of the destination mailbox determines how processing proceeds. If the mailbox is on an Exchange 2013 Mailbox server, the CAS can proxy the connection direct to that Mailbox server without any problems. However, if the mailbox is on an Exchange 2007 or Exchange 2010 server, the CAS has to use an HTTP proxy or redirection to transfer the connection to an Exchange 2007 or Exchange 2010 CAS, which then takes responsibility for making the final connection to the mailbox.

The proxy from the Exchange 2013 CAS uses Kerberos to communicate with its earlier counterparts, and the destination is the RPCproxy or Outlook Anywhere endpoint (the /rpc virtual directory in Internet Information Services [IIS]), which requires secured connections. If you configure Basic Authentication for Outlook Anywhere, IIS enables only Basic Authentication on the /rpc virtual directory. For this reason, IIS has to be updated to support Integrated Windows Authentication (IWA—previously known as NTLM) connections; otherwise, Kerberos won't work. However, if you were to modify IIS to support IWA, Exchange would overwrite it and incur the side effect of changing the authentication mechanism that clients use to connect to the CAS, which is probably not what you want to happen.

The workaround is reasonably simple—you run the Set-OutlookAnywhere cmdlet in the Exchange Management Shell to update IIS on all your older CAS servers so that internal (between CAS) connections are authenticated with Kerberos, and external (client) connections continue to use BASIC. The command is as follows:

```
Set-OutlookAnywhere -Name ExCAS01 -ClientAuthenticationMethod Basic
-IISAuthenticationMethods Basic, NTLM
```

If you've already decided to enable NTLM for Outlook Anywhere, you don't need to make any changes to IIS. However, even if this is the case, you still need to consider how the Exchange 2013 CAS proxies connections to other CAS servers located in internal or non-Internet-facing sites (that is, without a direct connection to the Internet through a firewall and so on).

This requirement does not exist for previous versions of Exchange, so it's unlikely that Outlook Anywhere is enabled on CAS servers in non-Internet-facing sites because they do not expect to process incoming Outlook Anywhere connections. However, if the Exchange 2013 CAS has to proxy an incoming connection because the target mailbox is on a server in an internal site, the proxy is to the Outlook Anywhere endpoint. It therefore follows that the CAS servers in these sites need to be updated by running the Enable-OutlookAnywhere cmdlet so that they can accept the incoming proxy to their Outlook Anywhere endpoint with NTLM authentication enabled.

The case for protocol-specific namespaces

Perhaps the Exchange developers were unaware of the law of unintended consequences when they decided to change the Exchange load-balancing requirement from layer 7 to layer 4 of the network stack. In most situations, the change is good. For some, especially those responsible for deployments that have to cater to large numbers of incoming client connections, it creates an interesting question about protocol handling that deserves some attention during the planning for the implementation of Exchange 2013.

Anyone dealing with high-end deployments based on Exchange 2007 and Exchange 2010 is very aware of the need to manage incoming connections carefully. Typically, the solution involved a hardware-based load balancer (running on physical or virtual hardware) that terminated the incoming SSL connection and then sent it on to one of an array of CAS servers that then processed the connection and directed it to the correct mailbox server. Usually, this worked, albeit at the expense of having a load balancer and some complexity in ensuring proper client connection affinity.

Exchange 2013 greatly simplifies matters. You still need to deploy hardware-based load balancers when high availability for client connections is required, but Exchange 2013 supports solutions such as Windows NLB and round-robin balancing to handle lower-end deployments. All this is because the Exchange 2013 CAS doesn't do the rendering and protocol handling that its predecessors did. Instead, the Exchange 2013 CAS simply proxies connections to the appropriate mailbox server, which does all the real work. The idea is to break the version linkage that previously existed between CAS and mailbox insofar as you couldn't upgrade one without the other. Version independence is a big theme for future versions of Exchange, and if all goes well, you'll be able to upgrade different parts of the infrastructure in the knowledge that the new components won't break anything running on the old bits.

Simplification is always good in computer technology because complexity invariably leads to additional cost, confusion, and potentially poorer results. However, any change has consequences, and one consequence that flows from the move to L4 is the loss of protocol awareness. When a load balancer terminates an incoming SSL connection at L7, it can sniff the packets and discover the protocol to which the connection is directed. Exchange offers a rich set of protocols and services, including Exchange Web Services (EWS), Outlook Web App, ActiveSync (EAS), the Offline Address Book (OAB), and Exchange Administration (ECP), each of whose endpoint is represented as an IIS virtual directory. However, when an L4 load balancer handles a connection, it sees it going to TCP port 443 and the IP address for the external connectivity point (such as mail.contoso.com). Later, the CAS will sort out the connection and get it to the right place, but that's too late to have any notion of protocol.

Contemplating management

Now that you've installed Exchange 2013 after making sure that your Active Directory was ready to support its deployment, you are ready to contemplate the challenge of understanding the management of Exchange. Exchange 2013 is firmly anchored in Windows PowerShell, so that's the next logical step in this odyssey.

CHAPTER 3

The Exchange Management Shell

How Exchange uses Windows PowerShell 83
Using remote Windows PowerShell. 86
EMS basics 92
Active Directory for PowerShell 120
Exploring useful EMS examples 123
Verbose PowerShell 129
Controlling access to Exchange 129

Windows PowerShell is an extensible automation engine consisting of a command-line shell and a scripting language. Exchange Server 2007 was the first major Microsoft application to support Windows PowerShell in a comprehensive manner. Although not every administrator welcomed the opportunity to learn a new scripting language, the overall impact was extremely positive. The role of Windows PowerShell continues to expand across Microsoft products, and it now extends into the newest Microsoft offerings, including the deployment and management of applications on the Azure cloud computing platform.

Windows PowerShell is built on top of the Microsoft .NET Framework and is implemented in the form of cmdlets, specialized .NET classes that contain the code to implement a particular operation such as the creation of a new mailbox or the enumeration of the processes that are currently active on a server. Applications implement Windows PowerShell support by providing sets of application-specific cmdlets that collectively represent the functionality required to support the application, or they can be used to access different data stores such as the file system or system registry. Cmdlets can be run separately or combined by piping the output generated by one cmdlet to become the input of the next. Cmdlets can also be combined into scripts (with a .ps1 file extension) to provide more comprehensive processing and logic or included in executables when the need exists to launch a standalone application. Many scripts are available on different Internet sites to assist with Exchange management.

How Exchange uses Windows PowerShell

From an Exchange perspective, Windows PowerShell provides a way to perform tasks quickly and simply in a variety of manners, from one-off interventions to process one or more Exchange objects to complex scripts to perform tasks such as mailbox provisioning. Most administrators cut their teeth on PowerShell by using the Exchange Management Shell (EMS) to do simple things, such as using Get-Mailbox to report on a mailbox's

properties and Set-Mailbox or Set-CASMailbox to set a property, before moving on to the more esoteric commands to manipulate connectors or control the ability of devices to connect through ActiveSync and so on. The saying is that almost anything is possible with Windows PowerShell, and this is certainly true when you dedicate enough energy and time to mastering the language, not to mention the time necessary to scan the Internet for useful examples of scripts that can be adapted to meet your needs.

Prior to Exchange Server 2007, business logic was scattered in components throughout the product. The management console did things—even simple things like setting a property on a server—by using different code and logic than in the setup program, and the application programming interfaces (APIs) included in the product usually provided a third way to approach a problem. The result was a total lack of consistency, duplication of code, and a tremendous opportunity to create bugs in multiple places. In addition, administrators could not automate common tasks to meet the needs of their organization; essentially, if an Exchange engineer didn't code something into the product, it couldn't be done.

Figure 3-1 illustrates the central role Windows PowerShell now plays in the Exchange architecture and shows how it provides a central place to encapsulate business logic that underpins the Exchange setup program, the Exchange Administration Center (EAC), the mailbox options that users can update through Outlook Web App, and the Exchange Management Shell (EMS).

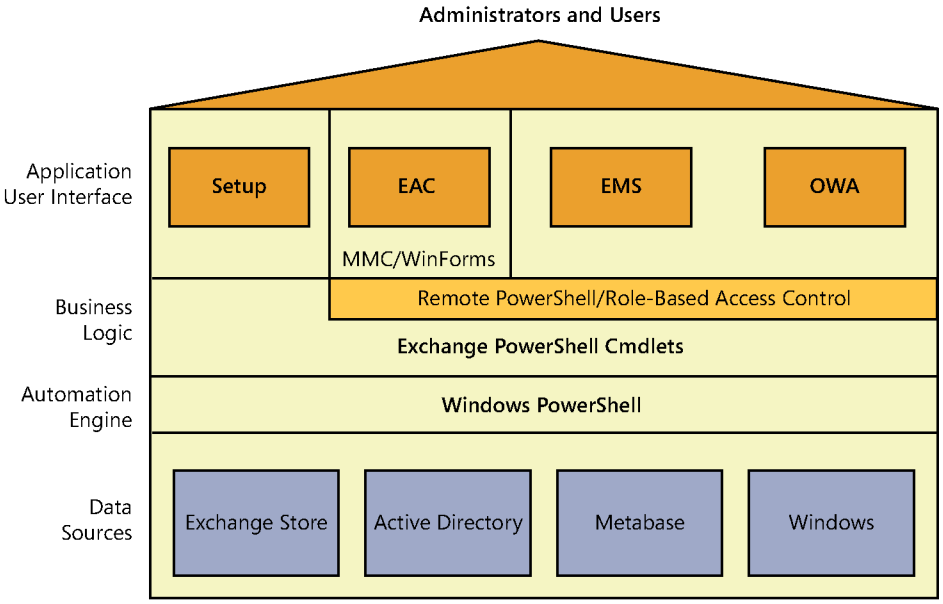


Figure 3-1 Windows PowerShell at the heart of Exchange

The way Exchange uses Windows PowerShell to implement business functionality is probably the most extensive of any Microsoft application. As explored throughout this book, the options presented by EAC to work with mailboxes, connectors, servers, and other objects invariably result in a call to one or more PowerShell cmdlets that actually do the work. The functionality presented to administrators, specialist users (those who perform a subset of administrative tasks such as maintaining user details), and normal users is all based on PowerShell.

The exact scope and range of the functionality presented to any individual user is determined by the permissions granted to him through role-based access control (RBAC). RBAC is designed to function across a range of environments, from a single-server organization to an organization composed of a mixture of on-premises and hosted servers. The need to accommodate such a wide range of environments is also why Microsoft has moved from local PowerShell (by which all commands are executed on a local server) to remote PowerShell (by which commands are redirected through Internet Information Services [IIS] for execution on a target server). The details of just how remote PowerShell and RBAC work together in EMS are covered shortly.

Simplifying the implementation of new functionality

The administrative interfaces in Exchange all lead to the same place and execute the same business logic. Apart from removing redundant and overlapping code, having a single place to implement business logic enables the Exchange engineers to concentrate on implementing new functionality rather than re-implementing features specifically for use by EAC, EMS, or the setup program. The approach enables Exchange to deliver a more consistent administrative environment and a comprehensive method to automate tasks to deal with mailboxes, databases, connectors, and all the other components that collectively make up an Exchange organization.

At the time of writing, Exchange 2013 RTM CU2 includes 965 cmdlets that are added to the standard set of Windows PowerShell cmdlets, including cmdlets to work with the system registry, file system, variables (including environmental variables), and so on that are available in an EMS session. Depending on the RBAC role groups of which your account is a member, the number of cmdlets available to you might vary.

Collectively, the set of EMS cmdlets manages the objects and the properties of the objects that form Exchange. Objects include mailboxes, servers, transport rules, connectors, and so on. You can determine the exact number of cmdlets Exchange owns by using the following command (this command doesn't work with Exchange Online):

```
Get-ExCommand | Measure-Object | Select Count
```

INSIDE OUT **Finding the cmdlets available to you**

As you learn in the discussion about RBAC in Chapter 4, "Role-based access control," an EMS session allows you access only to the cmdlets and parameters that are defined in the roles included in the role groups of which your account is a member. Accounts that are highly permissioned, such as those belonging to the Organization Management role group, can use many more cmdlets than those that belong to a less-permissioned role group, such as Help Desk or Recipient Management. You can use this command to generate a full list of all the Exchange 2013 cmdlets your account can access:

```
Get-ExCommand > C:\Temp\ExCommands.txt
```

By comparison, Exchange 2007 includes 394 cmdlets, Exchange 2010, 584; and the RTM version of Exchange 2013, 958. The hundreds of new cmdlets included in Exchange 2013 and subsequently augmented through cumulative updates reflect the new functionality in the product such as the introduction of site mailboxes and data loss protection policies, along with the expansion of existing functionality such as the changes to compliance.

PowerShell use and syntax are fundamental skills for Exchange administrators to master. In fact, many Exchange administrators prefer EMS to EAC because of the additional flexibility that EMS provides. This chapter lays out the basics of Windows PowerShell and sets the stage for the examples of PowerShell found in other chapters. To begin, review how the Exchange management tools actually connect to PowerShell.

Using remote Windows PowerShell

Exchange 2010 began the necessary transformation from a model that assumed an administrator would always have some form of physical access to a server to the point at which remote management has become the norm. Remote PowerShell provides the fundamental building block for connectivity to remote systems. The combination of remote PowerShell and RBAC enables administrators to manage objects residing on a server in a remote datacenter as easily as managing objects on a local server.

Note

You can think of Windows PowerShell as implemented in Exchange Server 2007 as "local PowerShell" because cmdlets are executed in a local process. The only element of remote access in Exchange 2007 is when you pass the –Server parameter to identify a server against which to execute a command. Even so, if data are needed from a remote server, such as fetching a set of mailbox objects, it is retrieved across the network and processed locally.

Exchange 2010 and Exchange 2013 support the remote execution of commands in a secure manner, using HTTPS and a Kerberos-based encryption mechanism that is easily manageable through firewalls (assuming that port 80 is open). Remote PowerShell is now used for all EMS sessions. Even if you are logged on to an Exchange server and want to use EMS to change a property of that server, EMS still creates a remote session on the local server to do the work. The same applies for EAC because Exchange creates a remote session when you log on to connect to a server in the local Active Directory site to retrieve information about the organization and then display it in the console. In effect, remote PowerShell has replaced local PowerShell for all server roles except edge servers. The sole exception is for commands used during setup, which continue to execute locally. Remote PowerShell separates business logic into code that runs on the client and code that runs on the Exchange server. The logic for replacing local PowerShell with the remote model is simple. Just as the change in Exchange 2007 forced all messages to flow through the transport system so that a common place existed to apply features such as transport rules, remote PowerShell forces all aspects of Exchange administration to flow through RBAC so that tight control can be achieved over the actions an administrator—or, indeed, a user—is allowed to perform.

An RBAC role group defines the set of administrative actions a user is allowed to perform inside Exchange and can be resolved into a set of PowerShell cmdlets the user is allowed to use within her PowerShell session. After it is created, a PowerShell session is populated with cmdlets by reference to the RBAC role groups of which the user is a member so that it will only ever include the cmdlets that have been assigned to an administrator through her membership in role groups. Thus, the fact that an administrator is not a member of a particular role group is reflected in that any PowerShell session she initiates will never be able to call the cmdlets associated with the management group. Consider the case of an administrator who is a member of the Organization Management role group, typically believed to be the all-powerful role for Exchange management. Even though membership in the Organization Management role group grants a user access to the vast majority of Exchange cmdlets, out of the box, it does not grant access to the set that controls movement of data into and out of mailboxes. To protect user data, the Mailbox Import-Export role has to be explicitly assigned to any user who wants to perform these tasks, even those who are already members of the Organization Management role group.

The need to support massively scalable, multitenant platforms such as Office 365 was a major influence on the Exchange move to remote PowerShell. Providing a secure and controllable mechanism to permit administrators to execute privileged commands to control the subset of objects they own inside an infrastructure that is controlled and managed by someone else is always a difficult task, especially when all the data have to pass across the Internet. When you use PowerShell to create a session with Exchange Online, the session is by default remote because you cannot connect to the servers running in Microsoft datacenters in any other way. After it accepts the credentials you provide in the connection request, PowerShell loads in the cmdlets you are allowed to use through membership in

RBAC role groups. The fact that Exchange Online restricts the number of cmdlets available to administrators when compared to on-premises deployments is immaterial. Exactly the same mechanism populates your PowerShell session with cmdlets whether you connect to an on-premises deployment or to a cloud-based service.

CAUTION!

Until Microsoft removes the functionality, it is possible to use local PowerShell with the Exchange 2013 snap-in to perform management operations on a server. However, Microsoft is not testing local PowerShell with Exchange anymore, and it is possible that problems will appear in local PowerShell that will never be resolved. In addition, running local PowerShell means that you bypass the controls otherwise imposed by RBAC. Given the engineering and strategic focus on remote PowerShell, it makes sense for everyone to make the transition now and embrace this platform as the future of command-line Exchange management.

Connecting to remote PowerShell

To understand how remote PowerShell and RBAC work together, examine how an administrator might create a new mailbox on a remote server. In this example, the administrator works on a help desk and has been assigned a role that enables him to create new mailboxes and update the properties of existing mailboxes. Further, assume that the user's account is enabled to use remote PowerShell. In many cases, people in specialist roles such as help desk personnel use EAC to perform tasks, but an experienced Exchange administrator might prefer to use a command-line interface because of its power and flexibility, especially when processing multiple objects, when compared to EAC.

Figure 3-2 lays out the various components remote PowerShell uses from the local PowerShell host on a workstation or server across the network to IIS and the PowerShell application running there. The other components are the PowerShell engine and the complete set of cmdlets available to Exchange 2013, the Exchange authorization library that handles the interpretation of roles in terms of the cmdlets that each RBAC role can use, and the Active Directory driver that reads data from Active Directory. For the purpose of this discussion, assume that the account has been assigned a role such as Recipient Management and is enabled for remote PowerShell. If you are unsure about the account's status, you can enable it to use remote PowerShell as follows:

```
Set-User –Identity AccountName –RemotePowerShellEnabled $True
```

All PowerShell sessions flow through IIS because even a local connection goes through localhost. All Exchange 2013 servers support IIS and the PowerShell virtual directory, or

vdir; all are members of the Exchange Trusted Subsystem security group and therefore can manipulate any object in the organization.

If you run EMS on a workstation or server on which the Exchange management components are installed, EMS creates a remote session automatically as part of its initialization process. If you run PowerShell on a workstation that doesn't have the Exchange management components installed, you must specify the name of the server with which you want to work. This is done by using the New-PSSession cmdlet, passing the name of the server to which to connect in the *https://fqdn/PowerShell/* form. This cmdlet creates a secure, authenticated connection to IIS running on the target server and begins a session there by checking the role held by the account that initiates the connection.

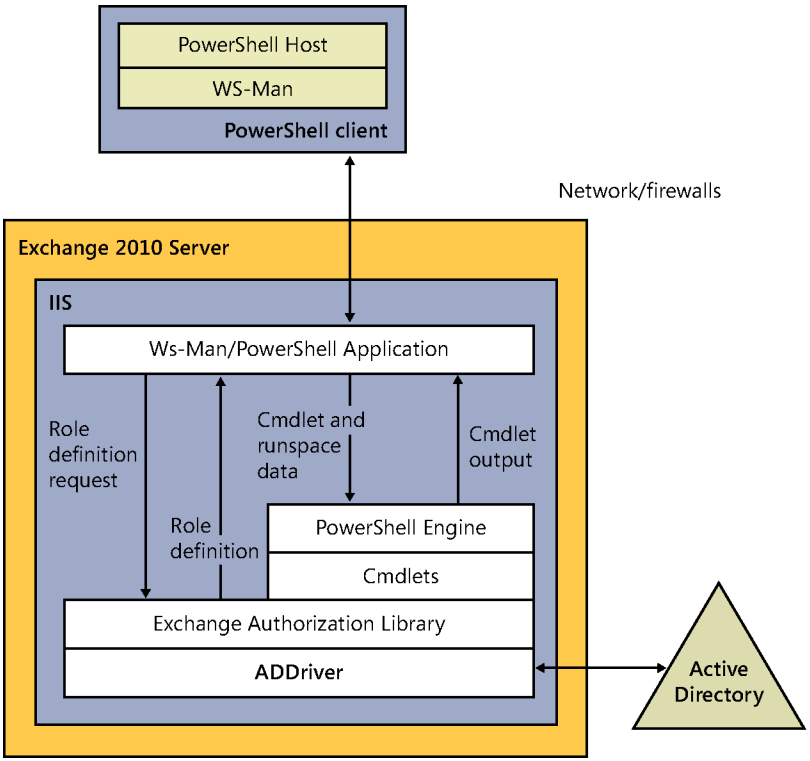


Figure 3-2 Remote PowerShell architecture

When you run EMS on a server on which the Exchange 2013 management components are installed, the EMS initialization script creates an environment by executing the code in the RemoteExchange.ps1 script (in the Exchange binaries folder), which first attempts to create a remote session with the local host. If successful, it then identifies your account to Exchange, uses RBAC to determine the cmdlet set you are allowed to use, and so on.

The RemoteExchange.ps1 script is quite complex because it has to handle the initiation of sessions in different circumstances, so it's easier to look at the code that connects a new session with Exchange Online to see how the basic requirements to create a remote PowerShell session are met. The following code defines the Connect-ExchangeOnline function, which is suitable for inclusion in your PowerShell profile (a document that initializes a session with common commands and settings that you might want to use during the session). In fact, this is how I connect to Exchange Online when I want to manage my tenant domain. Three commands are executed:

- Get-Credential gathers the name of the account with which to connect and its password.
- New-PSSession establishes the remote PowerShell session with Exchange Online.
- Import-PSSession imports the Exchange commands from the Exchange Online session into your current session.


```
function Connect-ExchangeOnline
{
  $LiveCred = Get-Credential
  $global:Session365 = New-PSSession -ConfigurationName Microsoft.Exchange
-ConnectionUri https://ps.outlook.com/powershell/ -Credential $LiveCred
-Authentication Basic -AllowRedirection
   Import-PSSession $global:Session365
}
```

How the initialization script finds a server

When a user creates a remote PowerShell session on an Exchange server, the initialization script attempts to connect him to the same server. If the attempt to establish a connection with the local server fails, the initialization script then enumerates the full set of Exchange servers in the local site and attempts to make a connection to one of the servers chosen at random. If this attempt fails, the script moves on to the next server and continues until a successful connection is established or all available servers have been attempted and have failed. The initialization script works in the same way when executed on a workstation that has the Exchange management components installed on it, except that the initialization begins by randomly selecting one of the servers from the site.

IIS uses the RBAC mechanism to check the user's role and associated permissions through the Exchange Authorization Library. The Exchange Authorization Library (or ADDriver) connects to Active Directory to use it as the definitive source of information about accounts and supplements these data with its knowledge about the Exchange-specific roles that

administrators have assigned to users. During a PowerShell session, ADDriver connects to a domain controller in the local site to fetch data from Active Directory and keeps this connection throughout the session (referred to as DC affinity). Many PowerShell cmdlets support the DomainController parameter to enable you to connect to a specific domain controller (specifying the fully qualified domain name [FQDN]) if the need arises.

Because this user works with mailboxes as defined by the Recipient Management role group, the set of cmdlets he can use includes commands with easily identified purposes such as New-Mailbox, Set-Mailbox, Get-Mailbox, and so on.

Tip

Permissions granted through RBAC are evaluated during session initialization. If you are assigned a new role, you have to create a new session with EMS or EAC before you can access the cmdlets made available through the newly assigned role.

Users are not normally aware that they are restricted in terms of available cmdlets unless they attempt to use one to which they do not have access. The point is that they shouldn't care that they can't use hundreds of cmdlets, many of which do obscure things such as setting properties on messaging connectors or performing one-off operations such as creating a new Database Availability Group (DAG) or mailbox database. Instead, RBAC makes sure that users can access only the cmdlets they need to perform their jobs.

Limiting user functionality

When a new PowerShell session is created, you see no evidence that your role has forced RBAC to restrict the cmdlet set or the parameters you can use with cmdlets because the initialization of a session progresses just as it would for a fully privileged user. However, after you start to execute cmdlets, you quickly realize that you can't do as much as you'd like. For instance, if you log on with a restricted user account and attempt to use the Get-Mailbox cmdlet to fetch a list of mailboxes, all you'll see is your own mailbox. This is logical because your role allows you to see details of your own mailbox but not others'. In the same way, if you then attempt to use the Set-Mailbox cmdlet to update a property that only administrators can access, you won't be able to use even tab completion to reveal a restricted property. However, unless the default role assignment policy has been amended to block access to basic user options, you can use the Set-Mailbox cmdlet to update properties that are generally exposed for user update through Outlook Web App options, so (assuming JSmith is the alias for your mailbox) you'll be able to do things such as this:

```
Set-Mailbox -Identity JSmith –MailTip 'Hello World'
```

or this:

```
Set-Mailbox -Identity JSmith -Languages 'EN-US', 'EN-IE'
```

INSIDE OUT **You can do some things; you can't do others**

Somewhat strangely, you'll also be able to execute Get-MailboxStatistics to report the number of items in your mailbox but not Get-MailboxFolderStatistics to report on the folders and the items that each contains. This is all controlled by RBAC, the roles that your account holds, and the scope for the roles in terms of the cmdlets and parameters defined in each role. From this discussion, you should now understand how critical RBAC is to remote PowerShell and, by extension, to every aspect of the Exchange 2013 management toolset.

EMS basics

Exchange 2013 RTM CU2 includes 965 cmdlets, but you're not likely to use the vast majority of these simply because many are designed for one-time use. For example, after you configure a receive connector, you probably will not revisit the Set-ReceiveConnector cmdlet very often after the connector is working. However, you'll use cmdlets such as Get-Mailbox daily. Some examples (in no particular order) of frequently used Exchange cmdlets are the following:

- **Get-ExchangeServer** Return a list of Exchange servers in the organization.
- **Disable-Mailbox** Disable a user's mailbox.
- **Add-DistributionGroupMember** Add a new member to a distribution group.
- **Set-Mailbox** Set a property of a user's mailbox.
- **Get-MailboxDatabase** Retrieve properties of a mailbox database.
- **Get-MailboxStatistics** Return statistics about user mailboxes such as the total item count, quota used, and so on.

Note the consistent syntax of verb (Get, Set, Move, Remove, or Disable) and noun (Mailbox, User, and so on). Along with commands that operate on objects, you find commands that help you work with data, such as Where-Object, Sort-Object, and Group-Object. Where-Object, Sort-Object, and Group-Object are commonly shortened by using their aliases of

Where, Sort, and Group. You can type **Help** followed by a cmdlet name at any time to get help on the syntax of the command.

> **Tip**
> **When you start to write scripts, consider spelling out cmdlet names completely and avoiding the use of aliases. This is important because you can never know in what environment a script will be run and therefore cannot assume that an alias will be defined and available for use in your code.**

The Exchange developers have provided very accessible help for the EMS cmdlets. Apart from using the Help cmdlet, there are other ways of seeking help. RBAC controls limit help content so that a user sees help only for the set of cmdlets available to the roles that user holds. You can do the following:

- Use the Get-Command cmdlet to list the cmdlets you can use with different objects. The set of cmdlets will be limited to whatever is permitted by the RBAC roles held by your account. For example, Get-Command *contact* lists all the cmdlets available to work with contacts (shown in the following example). You can also use the shortened alias of gcm for Get-Command.

```
CommandType             Name                              Definition
-----------             ----                              ----------
Function                Disable-MailContact               ...
Function                Enable-MailContact                ...
Function                Get-Contact                       ...
Function                Get-MailContact                   ...
Function                New-MailContact                   ...
Function                Remove-MailContact                ...
Function                Set-Contact                       ...
Function                Set-MailContact                   ...
```

- Use the /detailed switch to get more detailed help about a cmdlet. For example: Get-Help Get-CASMailbox –Detailed.
- Use the /full switch to have EMS return every bit of information it knows about a cmdlet. For example, Get-Help Get-DistributionGroup –Full.
- Use the /examples switch to see whatever examples of a cmdlet in use EMS help includes. For example, Get-Help Get-MailboxServer –Examples.

- Use the /parameter switch to get information about a selected parameter for a cmdlet. For example, Get-Help Get-Mailbox –Parameter Server. This switch supports wildcards, so you can do something like Get-Help Set-Mailbox –Parameter *Quota*.

INSIDE OUT **Getting to know the cmdlets**

You will probably begin by using the /full switch to retrieve all available help for a cmdlet to get to know what each cmdlet does. After you learn more about the cmdlet, you can move on to the default view as you become more accustomed to working with EMS. Remember that the Exchange help file contains information about all the EMS cmdlets. The advantage of using the help file (which is always present on a server) is that you can use the help file's index to search for specific entries.

Most of the time, you will probably work with commands by invoking EMS interactively and then typing whatever individual commands or scripts are necessary to perform a task. The user interface of EMS is based on the Win32 console with the addition of features such as customizable tab completion for commands. After you become accustomed to working with EMS, things flow smoothly, and work is easy. It is then usually faster to start EMS and issue the necessary code to change a property on a mailbox or a server than to start EAC and navigate to the right place to make the change through the graphical user interface (GUI).

Tip

Working through EMS is especially valuable if you have to perform management operations across an extended network link when waiting for the GUI to display can be painful. If you have a programmatic mind, you can also call EMS cmdlets through C# code, which is how Microsoft invokes them in EAC and other places throughout Exchange, such as to set up servers and databases in the setup program. (The blog Glen Scales writes at *http://gsexdev.blogspot.com/* provides many good examples of how to call EMS cmdlets from code.) In the past, the different groups that contributed to Exchange had to build their own programming interfaces, whereas now everyone uses PowerShell.

You can see that EMS focuses on performing tasks rather than taking the more object-focused approach implemented in the GUI, something that reflects a desire to accommodate administrators who think about how to do things rather than how to work with objects. After all, it is human nature to think in terms of the task of moving a mailbox to a different server rather than thinking about how to manipulate the properties of a mailbox object to reflect its new location.

Cmdlets accept structured pipelined input from one another in a common manner to allow them to process data in a consistent manner, no matter which cmdlet provides the data. Programmers therefore do not have to reformat data for input to specific cmdlets, so the task of assembling different cmdlets into a script to do a job is much easier. Microsoft built PowerShell around the concept of objects, so objects are accepted as input, and the output is in the form of objects that you can then pipe to other cmdlets. Even if the output from a cmdlet looks like plaintext, what you see is one or more objects that you can manipulate in a much more powerful manner than you can ever work with text output. The implementation is elegant.

Command editing

It should be apparent that you could do a lot of typing to enter commands into PowerShell, make the inevitable mistakes, correct them, and try again. To make the task a little easier, PowerShell supports the same kind of command-line editing as the Win32 console (CMD) does. Some of the more important keys you can use are described in Table 3-1.

TABLE 3-1 Command editing keystrokes for PowerShell

Keyboard command	Effect
F2	Creates a new command based on your last command. A pop-up screen appears in which to enter a character. PowerShell then creates a new command, using the last entered command up to the character you specify. For example, if the last command is Get-MailboxStatistics –Identity TRedmond, and you enter F2 followed by c, PowerShell inserts "Get-MailboxStatistics". You can then complete the command as you like.
F4	Deletes characters in the current command up to a specified position. For example, if the cursor is located at the "M" of Get-MailboxStatistics, and you enter F4 followed by x, PowerShell deletes "Mailbo" and the result is "Get-xStatistics". Although this example wouldn't result in a useful command, F4 is useful when you need to edit many parameters in a complex command.
F7	Opens a list of the last 50 commands used in the current session to enable you to select a command for reuse.
F8	Moves backward through the command history.
Tab	Requests PowerShell to complete a command based on what you've typed.
Left/Right arrows	Moves the cursor left and right through the current command line.
Up/Down arrows	Moves up and down through the history of previous commands.
Delete	Deletes the character under the cursor.
Insert	Toggles between character insert and character overwrite mode.
Backspace	Deletes the character before the cursor.

Most of these keys are straightforward. The two most interesting keys are F7 and Tab. F7 opens a list of the last 50 commands you have run in the current session (Figure 3-3) so that you can both see what you've done in the immediate past and select one of the commands to re-execute. You can type a couple of characters into the F7 list, and EMS will look for the first matching command, or you can use the Up and Down arrows to navigate through the command history. At times, it's more convenient to use Up and Down arrows because you can retrieve more commands and edit a command before executing it. (F7 selects the command and executes it immediately.)

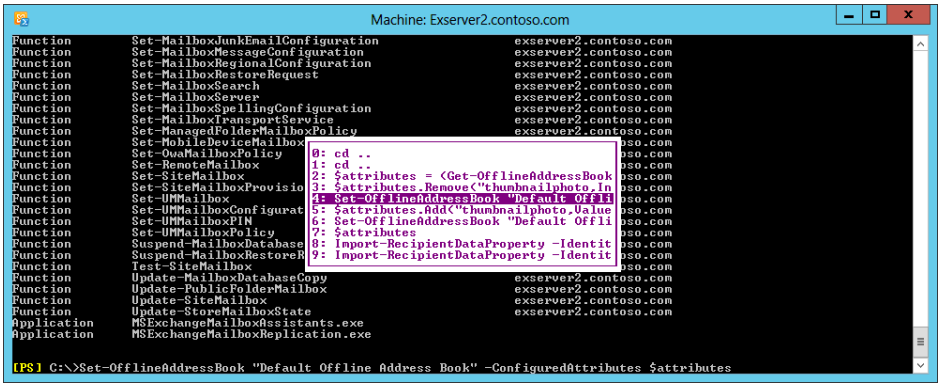


Figure 3-3 Using F7 to recall EMS commands

INSIDE OUT An easy way to type a command

Tab completion is a wonderful feature that Windows PowerShell inherited from CMD. You can partially enter a command and then press Tab to have PowerShell fill in the rest of the cmdlet name followed by its parameters. For example, type:

```
Get-Dist
```

This isn't the name of a valid cmdlet, but it is the root of several cmdlets, so when you press Tab, PowerShell completes the first valid cmdlet that matches and inserts:

```
Get-DistributionGroup
```

If you press Tab again, PowerShell moves to the next cmdlet that matches and inserts:

```
Get-DistributionGroupMember
```

If you press Tab again, PowerShell returns to Get-DistributionGroup because there are only two valid matches. PowerShell also supports completion for parameters. If you insert a dash to indicate a parameter value after Get-DistributionGroup and press Tab,

PowerShell starts with the first parameter and continues through all valid parameters. If you press Tab too many times and pass the parameter you want to use, you can press Shift+Tab to go back through the parameter list. If you add some characters to help PowerShell identify the parameter, it attempts to complete using that value. For example:

```
PowerShell completes Get-DistributionGroup –Ma into the command
Get-DistributionGroup –ManagedBy.
```

Even better, tab completion is context-sensitive, so it understands the structure of the object you are navigating. For example, if you want to move through the system registry, tab completion understands the hive structure, so you can type a location in the registry and then use the Tab key to move through the available choices from that point. For example, type:

```
CD HKLM:\Software\Microsoft\Exchange
```

Now press Tab, and PowerShell leads you through all the registry locations Exchange uses.

Windows PowerShell supports both named and positional parameters. Identifiers are a good example of a positional parameter. For example, if you enter Get-Mailbox Tony, PowerShell assumes that *Tony* is the value for the –Identity parameter.

Finally, PowerShell completes variables and even the properties of variables (such as their length) in a way similar to how the Microsoft Visual Studio IntelliSense feature works. If you type the incomplete name of a variable and press Tab, PowerShell completes it from the list of known variables. For example, if you fill a variable with details of a mailbox as in the following:

```
$Mailbox = Get-Mailbox –Identity Redmond
```

and then type **$Ma** and press Tab, PowerShell completes it and returns $Mailbox. This is a useful feature if you forget the names of variables you've defined. To see how properties are completed, type:

```
$Mailbox.Di
```

Pressing Tab now will request PowerShell to go through the list of properties beginning with Di. For a mailbox, the list is DistinguishedName and DisplayName.

Handling information EMS returns

Any cmdlet such as Get-EventLog that retrieves some information about an object will output a default set of properties about the object (or references to an object). Sometimes those properties are not exactly the ones you want to examine, so you will inevitably use the Format-List and Format-Table cmdlets to expand the set of properties a command returns. For example, if you use the Get-Mailbox cmdlet to view the properties of a mailbox, the information returned isn't interesting:

```
Get-Mailbox –Identity TRedmond
```

```
Name                 Alias            ServerName          ProhibitSendQuota
----                 -----            ----------          -----------------
Tony Redmond         TRedmond         ExServer1           unlimited
```

However, if you pipe the output to Format-List, you see much more information—far too much to review comfortably on screen—so it's better to pipe the output to a text file and compare it at your leisure.

The Get-Mailbox cmdlet does not return every property you can set on a user object because EMS differentiates between general Active Directory properties for a user object and those that are specific to Exchange. For example, Get-Mailbox does not list the Office property for a user because every user object in Active Directory has this property regardless of whether it is mail-enabled. Thus, if you want to retrieve or update the Office property, you have to use the Get-User and Set-User cmdlets, respectively. The same differentiation exists for groups and contacts when the Get-Group/Set-Group and Get-Contact /Set-Contact cmdlets are available.

Selective output

It is easy to list every property, but when you have limited screen space, you need to be more selective about the properties you want to output, and that's why it's often a good idea to use the Select-Object cmdlet to select the data you need before you pipe to Format-Table. In this case, you use the Select alias for Select-Object just because this cmdlet is used so often and it is nice to use shorthand.

```
Get-Mailbox –Identity Pelton | Select Name, PrimarySmtpAddress, Database
```

```
Name                 PrimarySmtpAddress                  Database
----                 ------------------                  --------
David Pelton         David.Pelton@contoso.com            ExServe1\DB1
```

PowerShell output can obscure data because it contains too many spaces. For example:

```
Get-ExchangeServer
```

```
Name        Site                  ServerRole   Edition      AdminDisplayVersion
----        ----                  ----------   -------      -------------------
EXSERVER1   contoso.com/Conf....  Mailbox,...  Enterprise   Version 15.0 (Bu...
EXSERVER2   contoso.com/Conf....  Mailbox      Enterprise   Version 15.0 (Bu...
```

To force PowerShell to remove spaces and display more useful data, pipe the output to the Format-Table cmdlet and use the –AutoSize parameter to fit the output columns into the available space:

```
Get-ExchangeServer | Format-Table -AutoSize
```

```
Name       Site      ServerRole              Edition        AdminDisplayVersion
----       ----      ----------              -------        -------------------
EXSERVER1  contoso.com/Configuration/Sites/Default-First-Site-Name
                     Mailbox, ClientAccess   Enterprise     Version 1...
EXSERVER2  contoso.com/Configuration/Sites/Default-First-Site-Name
                     Mailbox, ClientAccess   Enterprise     Version 1...
```

Another way of extracting and then working with data is to direct the output of a command into a variable, in which case you have a complete picture of the object's properties in the variable. For example, this command loads all the available information about the ExServer2 server into the $Server variable:

```
$Server = Get-ExchangeServer –Identity 'ExServer2' -Status
```

You can extract additional information about the server to use by including the name of the property in which you're interested. (Specifying the –Status parameter requests Get-ExchangeServer to provide some additional information about the current domain controller and global catalog the server is using.) You can also use a variable as an array and populate the array with a call to a command.

In this example, you populate a $Mailboxes array with a call to Get-Mailbox, using a filter to extract details of all the mailboxes stored in a particular database. This output is a good example of how cmdlets can generate individual objects or an array of objects with each object being individually accessible within the array.

```
$Mailboxes = Get-Mailbox –Database DB2
```

When it is populated, you can then navigate through the array as follows:

```
$Mailboxes[0]
$Mailboxes[1]
$Mailboxes[2] etc etc etc.
```

You can reference specific properties of the objects by using the "." operator.

```
$Mailbox[2].Name
$Mailbox[53].PrimarySmtpAddress
```

INSIDE OUT Finding what you want when there's a lot of output

The output from a cmdlet such as Get-Mailbox can easily result in a lot of data that are hard to read to find the piece of information in which you are really interested. One technique that helps is to pipe the output to the Out-String cmdlet and then use the FindStr cmdlet to search the output for a particular term. For example, here's how to use the two cmdlets to search the output from Get-Mailbox to find a particular term. In this instance, EMS lists any occurrence of the word "Tony" if it exists in the list of mailbox names Get-Mailbox returns:

```
Get-Mailbox | Out-String | FindStr "Tony"
```

By default, EMS truncates the output of multivalue properties after 16 values. For example:

```
Get-Mailbox –Identity 'Pelton, David' | Format-List Name, EmailAddresses


Name           : Pelton, David
EmailAddresses : {smtp:dp12@contoso.com, smtp:dp11@contoso.com, smtp:dp10@
contoso.com, smtp:dp9@contoso.com, smtp:dp8@contoso.com, smtp:dp7@contoso.com,
smtp:dp6@contoso.com, smtp:dp4@contoso.com, smtp:dp5@contoso.com, smtp:dp3@
contoso.com, smtp:dp2@contoso.com, smtp:dp1@contoso.com...}
```

Truncation can hide some valuable data. In the preceding example, many of the email addresses are defined for a mailbox, but the default Simple Mail Transfer Protocol (SMTP) address is not shown. If this limitation becomes a concern, you can force EMS to output more values for a property by amending a *$FormatEnumerationLimit* variable. This variable is defined in the EMS initialization script (\bin\Exchange.ps1), and the default value of 16 is usually more than sufficient for normal purposes. If you want to see more variables, you can set the variable to a different limit or set it to -1 to instruct EMS that it can enumerate as many values as are available for any property. For example:

```
$FormatEnumerationLimit = -1
Get-Mailbox –Identity 'Pelton, David' | Format-List Name, EmailAddresses
```

```
Name           : Pelton, David
EmailAddresses : {smtp:dp12@contoso.com, smtp:dp11@contoso.com, smtp:dp10@
contoso.com, smtp:dp9@contoso.com, smtp:dp8@contoso.com, smtp:dp7@contoso.com,
smtp:dp6@contoso.com, smtp:dp4@contoso.com, smtp:dp5@contoso.com, smtp:dp3@
contoso.com, smtp:dp2@contoso.com, smtp:dp1@contoso.com, smtp:dp@contoso.com,
smtp:dp16@contoso.com, smtp:dp15@contoso.com, smtp:dp14@contoso.com, smtp:dp13@
contoso.com, smtp:Pelton@contoso.com, SMTP:David.Pelton@contoso.com}
```

Using common and user-defined variables

PowerShell includes a number of variables you will use a lot. *$True* and *$False* are variables you can pass to shell commands and scripts to check for true and false conditions. Usually, *$True* is equivalent to setting a check box for an option in EMC, and *$False* is equivalent to clearing a check box. If you prefer numeric values, you can replace *$True* and *$False* with 1 (one) and 0 (zero), respectively. Other global variables you commonly meet as you work with PowerShell include *$Null* (no value), *$home*, which returns the user's home folder, and *$pwd*, which returns the current working folder. Important Exchange variables include the following:

- ***$ExBin*** Points to the directory in which Exchange binaries and other important files are kept. On an Exchange 2013 server, this variable normally resolves to disk: \Program Files\Microsoft\Exchange Server\V15\bin.
- ***$ExScripts*** Points to the directory in which important Exchange .ps1 scripts are kept. On an Exchange 2013 server, this variable resolves to disk: \Program Files \Microsoft\Exchange Server\V15\Scripts.
- ***$ExInstall*** Points to the root directory for Exchange. On an Exchange 2013 server, this variable resolves to disk: \Program Files\Microsoft\Exchange Server\V15.

You can use these variables to access files in these directories. For example, to see a list of scripts Exchange provides, type **Dir $ExScripts**.

Checking that a value is *$True* or *$False* is a common occurrence. For positive conditions, you can shorten the check by just passing the property against which to check, and PowerShell will assume that you want to check whether it is true. For example, assume that you want to find out which mailboxes are enabled to use Outlook Web App. You can use this command and, as you can see, there is no mention of *$True*, but it works:

```
Get-CASMailbox | Where-Object {$_.OWAEnabled} | Select Name
```

Note the use of *$_* in the last command. *$_* is a very important variable because it points to the current object in the pipeline. Scripting languages on other platforms such as UNIX and Linux also support pipelines, which compose complex commands by allowing the output of

one command to be passed as the input to another. The | operator indicates that a pipeline is in place. Data are passed as fully formed objects rather than as a text stream. This enables PowerShell to operate on the full structure of data that are pipelined, including the attributes and types that define the objects piped from one cmdlet to another.

For example, if you create a filter to look for people in a certain department because you want to update the name of the department, you might do this:

```
Get-User | Where-Object {$_.Department –eq 'Legal'} | Set-User –Department 'Law'
```

The Department property is prefixed with $_ to indicate that you want to check this property for every object the call to Get-User passes through the pipeline. You actually use $_. as the prefix because it includes the "." operator to specify that you want to access a property. If you just passed $_ the comparison would not work because PowerShell would compare "Legal" against the complete object.

User-defined variables can be integer, decimal, or string—you decide by passing a value to the variable you want to use. For example:

```
$Tony = 'Tony Redmond'
$Figure = 15.16
```

This creates a string variable, and the second variable holds a decimal value. Variables are case-insensitive and case-preserving. Using the preceding example, you can refer to $Tony as $TONY or $tony or even $ToNY, and PowerShell will refer to the same variable. Variables are local unless you declare them to be global by prefixing them with Global, as in:

```
$Global:Tony = 'Tony Redmond'
```

When a variable is global, you can reference it interactively and in scripts you can call from anywhere.

A word of caution about PowerShell and quotation marks

Be careful how you use quotation marks in PowerShell because although it might appear that double and single quotation marks are interchangeable, there is a subtle difference that might catch you out. Single quotation marks represent a literal string, one that PowerShell will use exactly as you provide it. Double quotation marks mean that PowerShell should examine the string and resolve any variable it finds inside through a process called variable expansion. Consider this example:

```
$n = Date
$n1 = 'Right now, it is $n'

Right now it is $n
```

```
$n2 = "Right now, it is $n"
$n2

Right now, it is Tue Jan 16 17:59:54 2013
```

Can you see the difference a little quotation mark makes? Best practice is to use single quotation marks whenever you are sure that you want a string variable to stay exactly as you have typed it and to use double quotation marks elsewhere. Be careful about using editors that insert smart quotation marks because PowerShell cannot deal with them; it is best to use a simple text editor whenever you create or edit a script. You cannot mix and match the different types of quotation marks to enclose a variable because PowerShell will refuse to accept the command. You will not do any great harm if you use double quotation marks instead of single quotation marks, but it is best to use single quotation marks as the default.

Tip

Do not include hyphens when you name variables because PowerShell interprets the hyphens as parameters. In other words, $ServerName is a good name for a variable, but $Server-Name is not.

Like any good scripting language, PowerShell supports conditional checking with IF and ELSEIF that you will mostly use in scripts. It's easy to generate code that goes through a certain number of iterations with constructs such as `1..100 | ForEach-Object <command...>`. You will see examples of these constructs as you see more sophisticated PowerShell code in later chapters.

Using PowerShell ISE with Exchange

If you don't like the bare-bones nature of EMS, you might prefer to use ISE, the PowerShell Integrated Scripting Environment. ISE is installed on Windows 2008 R2 SP1 and Windows 2012 servers to provide a GUI for PowerShell that allows users to write, test, and debug scripts. PowerShell ISE is also installed by default on Windows 7 and Windows 8 workstations.

ISE supports multiline editing, tab completion, syntax coloring (or highlighting of different parts of commands), context-sensitive help, and keyboard shortcuts. Because of its debug features, ISE is a good way to write complex scripts for use with Exchange 2013. All the code included in this book can be worked on through ISE.

When you start ISE, it has no knowledge of Exchange or how to create the kind of remote session with an Exchange server in the way EMS does when it starts. Some work is therefore necessary to integrate ISE with Exchange. The easiest way to do this is to insert some code in the PowerShell profile so that ISE learns enough about Exchange when it initializes to access Exchange when you need it to.

The code you need to use with ISE is very similar to the code you met earlier when discussing the basics of creating a remote PowerShell session. Start ISE and type **Notepad $Profile** to edit your PowerShell profile, and then insert the following code (amending the reference to contoso.com to reflect your own environment):

```
$PSISE.CurrentPowerShellTab.AddOnsMenu.SubMenus.Add(
 "Connect to Exchange", {
 $user = Get-Credential
 $Server = Read-Host "Connect to what Exchange server "
 $connectpoint = $Server + ".contoso.com/PowerShell/"
 $ExSession= New-PSSession -ConfigurationName Microsoft.Exchange -ConnectionUri
$connectpoint -Credential $user
 Import-PSSession $ExSession
 },
 "Control+Alt+1"
)
```

The code defines a new menu choice called Connect to Exchange that appears on the ISE Add-ins menu. The option can also be invoked with the Control/Alt/1 key combination. In either case, when invoked, the code prompts for user credentials and the server to which to connect and then initiates a new remote PowerShell session with the selected Exchange server. After the connection is established, you can work as with EMS except that extra information and facilities are available to you, such as a context-sensitive list of cmdlets that appears when you start typing a cmdlet name (Figure 3-4).

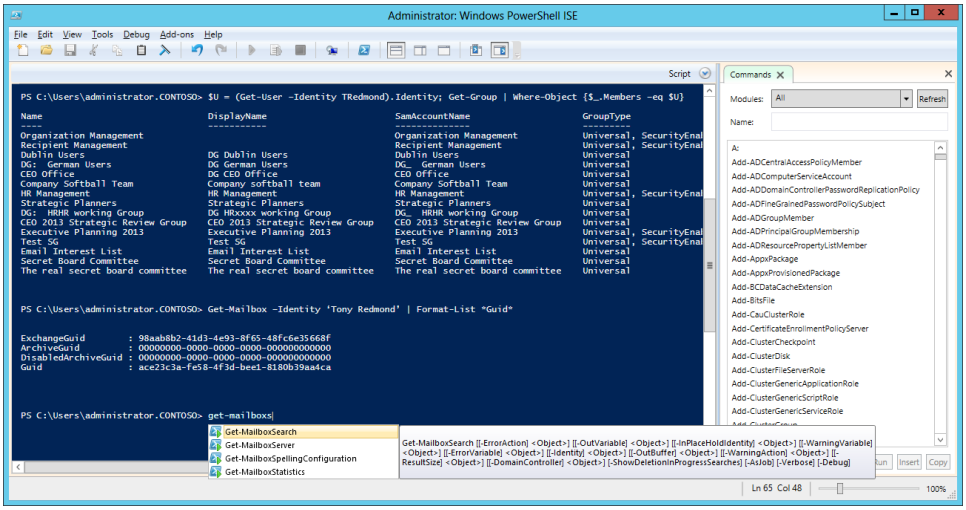


Figure 3-4 Working with Exchange 2013 through the PowerShell ISE

Identities

You might have noticed the –Identity parameter in some of the cmdlets you have explored so far. In many cases, a call to an Exchange cmdlet results in a set of objects being returned (for example, all the mailboxes on a server). In these instances, you might need to identify a specific object within the chosen set with which to work. (Think of a pointer to an item in an array.) For example, if you issue the Get-ExchangeServer cmdlet, you retrieve a list of all the Exchange servers in the organization. If you want to work with one server, you have to tell EMS which server you want to select by passing its identity. For example, to work with just the server named ExServer1:

```
Get-ExchangeServer –Identity 'ExServer1'
```

Apart from its obvious use to identify the object with which you want to work, –Identity has a special meaning within PowerShell because it is a positional parameter. You can specify the parameter's value without specifying the parameter's name, so the example previously used is just as valid if you use:

```
Get-ExchangeServer 'ExServer1'
```

INSIDE OUT **Best practice to include the –Identity parameter**

Although you might find it faster to omit the –Identity parameter when you're working interactively with EMS, it is best practice always to include the –Identity parameter when you write code for reusable scripts because this ensures that there is no possibility that another administrator or programmer will mistake the value passed for the identity for anything else.

If you want, you can retrieve a list of objects and store them in a variable and retrieve the values as you wish. The variable holds the objects as an array. For example, to populate a variable with a set of mailboxes hosted by a server:

```
$Mbx= Get-Mailbox –Server 'ExServer1'
```

To retrieve the different objects in the array, pass the number of the object with which you want to work, starting from zero. For example, to fetch the first mailbox in the array:

```
$Mbx[0]
```

You can be more specific and ask for one of the object's properties. For example, to get the identity of the first mailbox in the array:

```
$Mbx[0].Identity
```

```
IsDeleted           : False
Rdn                 : CN=Eoin P. Redmond
Parent              : contoso.com/Exchange Mailboxes
Depth               : 3
DistinguishedName   : CN=Eoin P. Redmond,OU=Exchange Mailboxes,DC=contoso,DC=com
IsRelativeDn        : False
DomainId            : contoso.com
ObjectGuid          : 0bcd15b3-c418-43be-b678-2658614f732b
Name                : Eoin P. Redmond
```

You might be surprised by the amount of information returned here for the mailbox's identity (it's all defined in the schema), but it contains all the ways you can navigate to this object through its relative distinguished name (shown here as the rdn property), distinguished name, globally unique identifier (GUID), and name. Normally, you'll just use the name of a mailbox to find it, but you can use the other methods, and Exchange will find the mailbox. There is no requirement to parse out a specific piece of the identity you want to

use or to trim values; PowerShell does it all for you. For example, you can use an identity to discover the groups to which a user belongs. Here's the code:

```
$U = (Get-User -Identity TRedmond).Identity; Get-Group | Where-Object {$_.Members -eq
$U}
```

The Get-User cmdlet loads the user's identity into a variable, and then the Get-Group and the Where-Object cmdlets scan all groups to discover any that include the user in their membership. Scanning the membership list of groups to discover string matches is never going to be as quick (and will get slower as the number of groups in the forest grows) because a string compare will never get close to the backward pointers that consoles such as Active Directory Users and Computers or EMC use to display group membership in terms of speed of access, so don't be surprised. Scanning for group membership in this way takes some time to complete.

If you don't like user-friendly forms such as email addresses or mailbox names, Exchange also allows you to use GUIDs as identifiers. Because they are obscure and long, GUIDs are difficult to type, but you can still use them. One slightly complicating factor is that you must know which GUID to use where. You might want the GUID that points to a user's mailbox, the GUID pointing to her Active Directory account, or even the one pointing to her archive mailbox. For example, this command displays all GUIDs registered for a mailbox:

```
Get-Mailbox -Identity 'Tony Redmond' | Format-List *Guid*
```

```
ExchangeGuid        : c2c4a3b5-c1a6-5a17-971d-8549123a78d0
ArchiveGuid         : 00000000-0000-0000-0000-000000000000
DisabledArchiveGuid : 00000000-0000-0000-0000-000000000000
Guid                : 288617d1-4592-4211-bb20-26ab755458c8
```

The ExchangeGuid property points to the user's mailbox. This is a tremendously important property because the GUID pointing to a mailbox can be guaranteed to be unique across an Exchange organization, which is why the Store uses this value to locate a user's mailbox. It's also why Outlook users see the ExchangeGuid of their mailbox instead of the server name when viewing the server name property shown when viewing the server settings of an Exchange 2013 mailbox (Figure 3-5).

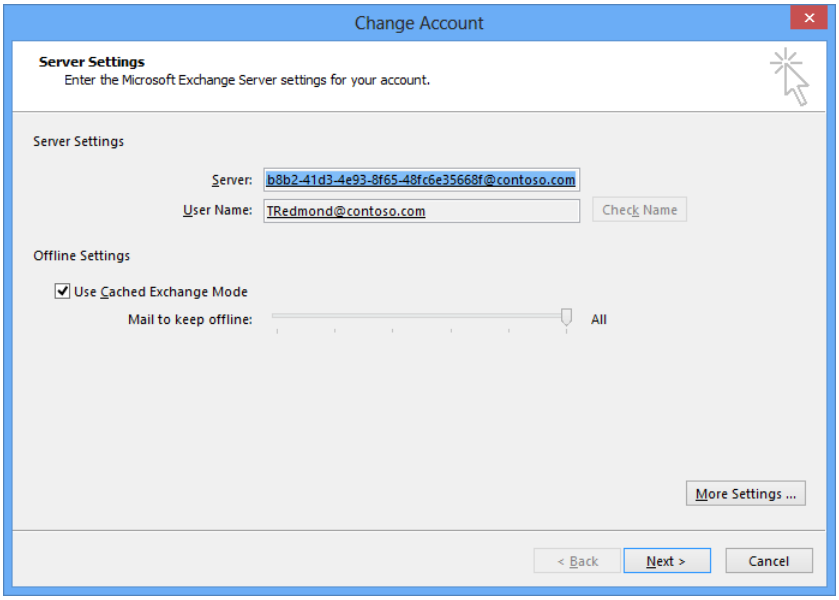


Figure 3-5 How Outlook displays the ExchangeGuid

It is confusing, but if you run Get-MailboxStatistics to retrieve summary details of the contents of a mailbox, EMS returns a MailboxGuid property. This is the same value as the ExchangeGuid when reported by Get-Mailbox. Why Microsoft felt that two names were required for the same GUID is beyond me.

The Guid property identifies the user's Active Directory account and thus provides the essential link between a mailbox and an account. In this case, the ArchiveGuid is shown as all zeros, so no archive mailbox is associated with this mailbox. The DisabledArchiveGuid value is also all zeros. This GUID is used only when a user has been assigned an archive mailbox that was subsequently disabled for some reason. Exchange maintains the GUID so the archive can be reconnected to the mailbox up to the point at which it is permanently removed from a database after the expiry of the deleted mailboxes' retention period.

Now that you know what the GUIDs are, you could use them to reference a mailbox. For example:

```
$GUID = (Get-Mailbox -Identity 'Tony Redmond').Guid
Get-User | Where {$_.Guid -eq $GUID} | Format-Table Name
```

The great thing about identities is that you sometimes don't need to use them. This situation occurs when you pipe information from one cmdlet for processing by another because the shell understands that it needs to operate on the current object that has been fetched

through the pipe. For example, this command pipes a list of mailbox identities passed in strings to the Set-Mailbox cmdlet:

```
"TRedmond", "JSmith", "JDoe" | Set-Mailbox -Office "Dublin"
```

Piping

You'll pipe output from one cmdlet to another frequently as you work with Exchange data. The important thing to remember is that PowerShell outputs fully formed objects that can be manipulated when fed as input to other cmdlets through the pipeline. This wouldn't be possible if PowerShell output text strings. For example, assume that you want to change the value of the Office property for a set of users who have moved to a new building. It would be tedious if you had to fetch the identity of each user individually, determine each identity, and then pass the value to make the change to each user's properties. A simple pipe works because PowerShell knows that it can use the stream of data from one command to identify the objects it has to process with another. Here's how you might update the Office property for a complete set of users without any mention of an identity. You'll see that the two cmdlets that do the work are separated by the pipe character, "|". This is the character that tells PowerShell to pipe the output from the first cmdlet to become the input to the second.

```
Get-User -Filter {Office -eq 'Building A'} | Set-User -Office "Building B"
```


Too many objects

By default, EMS returns up to 1,000 objects in response to cmdlets. (The value in Exchange 2007 is 5,000.) Therefore, if you run a cmdlet such as Get-Mailbox, Exchange will return up to 1,000 mailboxes if they are available. If you work in a small Exchange organization that supports fewer than 1,000 mailboxes, you don't need to worry too much about the number of objects you have to deal with because PowerShell will likely return relatively few objects, and things usually progress quickly. However, it's a different situation in large organizations, in which you have to pay attention to the filters you specify to retrieve data or override the default limit for returned objects by specifying the ResultSize parameter for cmdlets. For example, to let EMS return as many mailboxes as it can find, you could use a command like this:

```
Get-Mailbox -ResultSize Unlimited
```

This command will work, but it will be very slow because EMS has to read every mailbox in the organization. Think about how long this might take to execute in an organization that supports more than 300,000 mailboxes. In these situations it's always better to specify a filter to restrict the number of objects EMS looks for and returns.

OPATH filters

OPATH is the basic syntax used for PowerShell queries. It is similar in concept to but uses different syntax from Lightweight Directory Access Protocol (LDAP) queries. Dynamic distribution groups (see Chapter 6, "Groups and other objects") also use OPATH queries to locate objects in Active Directory when the transport system builds addressee lists to deliver message addresses to these groups.

Some base guidelines about the syntax OPATH queries are as follows:

- OPATH requires a hyphen before –and, –or, and –not operators.
- Comparison operators include –eq (equal), –ne (not equal), –lt (less than), –gt (greater than), –like (like), –ilike, and –notlike. –Like and –notlike are wildcard string compares. –iLike and –inotlike are case-insensitive.
- Filters should be expressed within braces; for example, {Office –eq 'London'}.

You'll see many more examples of OPATH queries in the remainder of this book.

Server-side and client-side filters

Windows PowerShell supports server-side and client-side filters. There's a big difference in performance between the two types of filters, especially when you have to process more than a hundred objects. Client-side filters are the default. Any code that uses the Where cmdlet executes a client-side filter. Client-side filters request data from a server and then perform the filtering on the client. This is an effective approach if you only have 10 or 15 objects to process, but it obviously doesn't scale too well as the number of objects increases.

Server-side filters have better scalability because the request for data forces the server to return a filtered data set to the client. Because Exchange servers often have to deal with tens of thousands of objects, a number of the Exchange cmdlets support server-side filters. If a cmdlet supports the –Filter parameter, it supports server-side filters. Usually, these are cmdlets that deal with objects that output large numbers, such as mail-enabled recipients or message queues. All the precanned filters generated for dynamic distribution groups, address lists, and email address policies use server-side filters.

As an example of server-side and client-side filtering in action, two methods are available to find all the mailboxes with "James" in their name, as demonstrated in these commands:

```
Get-Mailbox –Filter {Name –like '*James*'} –ResultSize 5000
Get-Mailbox –ResultSize 5000 | Where {$_.Name –like '*James*'}
```

On the surface, these two pieces of code seem reasonably similar, but they are very different in reality. The first difference is that the first code example uses a server-side filter, and the second uses a client-side filter. The second difference is that the two filter types can generate very different results because of the way the filters operate. If you omit the –ResultSize parameter, the same query is generated: Find all the mailboxes with a name that contains "James." (The ResultSize parameter in the first example limits the total number of objects returned to 5,000.) However, if you time both queries, the server-side filter invariably executes faster than the client-side filter, largely because fewer data are transferred between server and client. To understand why the filters generate different results, you have to appreciate how the filters work:

- The server-side filter returns the first 5,000 mailboxes it finds that include "James" in the mailbox name.
- The client-side filter fetches data for the first 5,000 mailboxes and then applies the filter to find the mailboxes that include "James" in the mailbox name. However, the filter applies only to the set the client fetched and might not find all the mailboxes you actually want to discover.

Even though you ask the server-side filter to do more work (working with any reasonably sized set of mailboxes, the server-side filter will have to process significantly more data to find the first 5,000 mailboxes that match), it still executes faster. For example, when I executed similar commands within a very large Exchange organization (170,000 mailboxes), the server-side filter completed processing in 43 seconds, whereas the client-side filter completed in 81 seconds. The rule here is that the effect of server-side filtering gets better as the number of objects increases.

INSIDE OUT PowerShell and memory limits

Another aspect to consider is that PowerShell cannot fetch and cache data on disk temporarily the way a database might. This is not an issue if you want to process only a few objects, but it can lead to memory issues if you attempt to process tens of thousands of mailboxes at one time, especially if you use client-side filters and want to pipe the output to another command. In this case, you ask PowerShell to find all the objects that match the specified filter, store the data in memory, process the data, and pipe the matching objects to the second command. Experience shows that these operations can cause PowerShell to complain that it is running out of memory. This is likely to be one of the growing pains through which all software goes and, apart from using loops to process data, no good solution to the memory exhaustion problem is available today.

Sometimes people make the mistake of assuming that client-side filters are faster because server-side filters provide the data in one motion after the server processes all the data. You therefore wait for a while without seeing anything and then see all the filtered records at one time. By comparison, client-side filters fetch and filter data continuously, so you see output as the command finds each matching record. However, the important indicator of performance is how long each type of filter takes to complete, and server-side filters are always faster.

The commands you are most likely to use with server-side filters are as follows:

- **Get-User** Retrieve basic Active Directory properties for any user account, including mail-enabled accounts.
- **Get-Mailbox** Retrieve Exchange-specific properties for mailboxes.
- **Get-DistributionGroup** Retrieve Exchange-specific properties for mail-enabled groups.

Each of the commands you can use to work with user accounts, groups, and mailboxes supports a different set of filterable properties. To discover which properties are available for filtering, you can use PowerShell to query the properties of a returned object. For example:

```
Get-Mailbox -Identity Redmond | Get-Member | Where-Object {$_.MemberType -eq
'Property'} | Sort-Object Name | Format-Table Name
```

This set of commands calls a command to return some information about an object. It then pipes the information returned by the first command to the Get-Member cmdlet, which extracts information about the properties. You sort the properties by name and output them in table format. Here's an excerpt from the output:

```
Name
----
AcceptMessagesOnlyFrom
AcceptMessagesOnlyFromDLMembers
AddressListMembership
Alias
AntispamBypassEnabled
CustomAttribute1
CustomAttribute10
...
WindowsEmailAddress
```

This method works for the Get-Mailbox, Get-CASMailbox, Get-User, Get-Recipient, Get-DistributionGroup, and Get-DynamicDistributionGroup cmdlets. You can use any of the values reported in a –Filter statement. For instance, the call you just made to Get-Mailbox

reports that the custom attributes are available, so to find all mailboxes that have a value in the CustomAttribute10 property, you can generate a command like this:

```
Get-Mailbox -Filter {CustomAttribute10 -ne $Null}
```

If you look at the filterable properties reported by the Get-DynamicDistributionGroup cmdlet, you can see that the ManagedBy property is available for this dynamic distribution group, whereas it is not for mailboxes. Hence, you can execute a filter like this:

```
Get-DynamicDistributionGroup -Filter {ManagedBy -ne $Null}
```

When you create a filter, it is best to be as specific as possible. You can state several conditions within a filter. An example of a server-side filter that returns all the mailboxes in the Dublin office where the user name contains "Tony" is shown next. The Get-User cmdlet also works with this filter, but Get-Mailbox executes a tad faster because the server does not have to process accounts that are not mail-enabled.

```
Get-Mailbox -Filter {Office -eq 'Dublin' -and Name -like '*Tony*'}
```

After you have mastered server-side filtering, you will use it all the time to work with sets of users. For example, assume that you want to give a new mailbox quota to members of a certain department but no one else.

```
Get-User -Filter {Department -Eq 'Advanced Technology'} | Set-Mailbox
-UseDatabaseQuotaDefaults:$False
-IssueWarningQuota 5000MB -ProhibitSendQuota 5050MB -ProhibitSendReceiveQuota 5075MB
```


INSIDE OUT WhatIf and Confirm

Before you execute any command to perform a bulk update of objects, you can run the command with the /whatIf switch added to force EMS to show you which objects will be altered. After you are sure that the correct set of objects will be updated, you can run the command without /whatIf, and EMS will perform the changes. The /confirm switch is also useful in terms of stopping administrators before they do something they should not. If you include the Confirm parameter, EMS prompts the administrator with "Are you sure that you want to perform this action" and waits for a "Y" or "Yes" response (or "A" for "all" if multiple objects are involved) before continuing. Act in haste, repent in leisure.

Transcripts

If you encounter a problem executing some EMS commands and need to produce some debug information to give to your support team or Microsoft, you can do this by generating a transcript. A transcript captures details of all commands executed in a session and is useful in terms of capturing the steps necessary to solve a problem or documenting steps to expose an issue that you want to report to Microsoft. You can combine this by adding the –Verbose parameter to most commands to gather a lot of information about what you've tried to do and what happened when you tried it. Use the Start-Transcript cmdlet to force EMS to capture debug information. For example:

```
Start-Transcript c:\Temp\Transcript.txt
```

All commands and output will be captured until you stop the transcript by using the Stop-Transcript cmdlet. At this point, you can examine the output with any text editor, and you'll see something like the output shown in the following example.


```
**********************
Windows PowerShell Transcript Start
Start time: 20130313093116
Username  : CONTOSO\Administrator
Machine   : ExServer1 (Microsoft Windows NT 6.2.9200.0)
**********************
PS C:\temp> $env:path
C:\Windows\system32\WindowsPowerShell\v1.0\;C:\Windows\system32;C:\Windows;C:
\Windows\System32\
Wbem;C:\Windows\System32
\WindowsPowerShell\v1.0\;C:\Windows\idmu\common;C:\Program Files\System Center
Operations Manager 2007\;C:\Program Files\Microsoft\Exchange Server\V14
\bin;c:\temp
```

Bulk updates

Those faced with the task of bulk updates (either to create a lot of new mailboxes or other objects or to modify many existing objects) before the advent of PowerShell support for Exchange had quite a lot of work ahead of them because Exchange offered no good way to perform the work. You could create comma-separated value (CSV) or other load files and use utilities such as CSVDE or LDIFDE to process data in the files against Active Directory, or you could write your own code to use CDOEXM or ADSI to update Active Directory. Either approach involved a lot of detailed work and made it quite easy to make a mistake. Using a console to make the necessary changes was boring and an invitation to make a mistake. The cause of Exchange's problems with bulk changes was the lack of a programmable way to automate common management operations, a situation that changed with the arrival of EMS.

You can combine the Get-User and Set-Mailbox cmdlets effectively to solve many problems. Here is an example in which you need to update the send quota property on every mailbox for a set of users whose business group has decided to fund additional storage. You can identify these users by their department, which always starts with "Advanced Tech" but sometimes varies into spellings such as "Advanced Technology" and "Advanced Technology Group." Conceptually, the problem is easy to solve:

1. Look for all users who have a department name beginning with "Advanced Tech."
2. Update the send quota property for each user.

You could use the Find option in Active Directory Users and Computers to build a suitable filter to establish the set of users, but then you have to open each user's mailbox that Active Directory Users and Computers locates to update his quota through the GUI, which could become boring after several accounts. You could also export a CSV-formatted list of users to a text file, manipulate the file to find the desired users, and then process that list through CSVDE to make the changes, but you have to search for all matching users across the complete directory first. That is a lot of work to do.

The process is easier in EMS. First, you use the Get-User cmdlet with a suitable filter to establish the collection of mailboxes you want to change. The following command returns all users who have a department name that begins with "Advanced Tech" and then updates the ProhibitSendQuota property to the desired amount (say, 20 GB). Because you have a collection of user objects established, you can use the Set-Mailbox cmdlet to perform the update. Note that some of these users might not be mail-enabled, but error handling is another day's work.

```
Get-User | Where {$_.Department -like '*Advanced Tech*'} | Set-Mailbox
-ProhibitSendQuota 20GB -UseDatabaseQuotaDefaults $False
```

Mergers, acquisitions, and internal reorganizations pose all sorts of problems for email administrators. EMS will not solve the big problems, but it can automate many of the mundane tasks that are necessary. For example, department names tend to change during these events. EMS makes it easy to find all users who belong to a specific department and update their properties to reflect the new organizational naming conventions. If only executing organizational change were as easy as this one-line command, which transfers everyone who works for the Old Designs department over to the Cutting Edge Design department, things would be much easier:

```
Get-User | Where {$_.Department -eq 'Old Designs'} | Set-User -Department 'Cutting
Edge Design'
```

Note the use of $_.Department; this indicates a value fetched from the current pipeline object. In this case, it is the department property of the current user object that Get-User

fetched. To verify that you have updated all the users you wanted to (and maybe provide a report to human resources or management), you can use code like this:

```
Get-User | Where {$_.Department -eq 'Cutting Edge Design'} | Select Name,
Department | Sort-Object Name | Format-Table > c:\temp\Cutting-Edge.tmp
```

A variation on this theme is to output the data to a CSV file to make the data easier to work with in Microsoft Excel, Microsoft Access, or another tool that can read CSV data.

```
Get-User | Where {$_.Department -eq 'Cutting Edge Design'} | Select Name, Department
| Sort Name | Export-CSV c:\temp\Cutting-Edge.CSV
```

Things are even easier if you just need to change everyone's company name after your company is acquired.

```
Get-User | Set-User -Company 'New Company'
```

You can even do such things as alter only the users whose mailbox belongs to a particular database:

```
Get-Mailbox -Database 'VIP Mailboxes' | Set-User -company 'Big Bucks'
-Department 'Executives'
```


Tip

All the examples discussed so far depend on you being able to identify some property you can use as the basis for a filter. But what about when you do not have a common property value to check for? In this case, you can build a simple list of mailbox names (or any other format the –Identity parameter will accept, such as a Universal Principal Name [UPN]), use the Get-Content cmdlet to read the names one by one, and pipe these values to whatever other command you need to use. For example, here is how you can use that trick to enable ActiveSync access for a set of users. In this example, the Get-Content cmdlet reads lines containing the identities of the mailboxes you want to change from a text file and pipes them as input to the Set-CASMailbox cmdlet:

```
Get-Content c:\temp\Users.txt | Set-CASMailbox -ActiveSyncEnabled $True
```

Another example of when EMS excels is when you want to apply a common setting across all servers in your organization. For example, assume that you want to apply a new deleted item retention limit of 150 days (perhaps mandated by the legal department) to all servers:

```
Get-MailboxDatabase | Set-MailboxDatabase -DeletedItemRetention 150.00:00:00
```

These simple examples demonstrate the value of having a scripting language that supports automation of common management tasks.

Calling scripts

After you have written a script, you have to decide where to keep it. You could put the new script in the directory that stores the Exchange binaries, but this is a bad idea for many reasons, not least because your script could be overwritten by the installation of a future Exchange service pack, a roll-up update, or even a completely new version.

INSIDE OUT **A wise practice**

It is wise to maintain a clear separation between the code for which you are responsible and the code Microsoft distributes with Exchange. Therefore, you should create a directory to hold all the scripts you use to work with Exchange. You can then call your scripts safely in the knowledge that they will be available.

The basic rule of calling a script is that if the script is in the working directory (the directory you are currently in), you prefix the name with ".\"

```
C:>.\Get-All-Users.ps1
```

If you're not in the right directory, you can move to where you want to be by using the cd command:

```
C:> cd c:\Scripts\
```

Alternatively, you can supply the full path to where the script is located:

```
C:>c:\Scripts\Get-All-Users.ps1
```

If there are spaces in the directory names, then you need to enclose the path in single or double quotation marks:

```
C: '\Program Files\Microsoft\Exchange Server\V15\Scripts\CollectOverMetrics.ps1'
```

Even better, you can amend the path PowerShell uses by looking for scripts and adding your directory to it. For example, running this command adds the C:\MyScripts directory to the path:

```
$env:path = $env:path + ";c:\MyScripts'
```

After a script is in a directory that's included in the path, you can invoke it by just typing its name.

Execution policies

EMS is powerful, and just a few cmdlets can have a tremendous effect on many objects throughout Exchange. You might have thought about how to control the ability of users to execute EMS commands.

RBAC provides the first line of protection. As you recall, users are permitted access only to the set of cmdlets and parameters available to the roles each user holds. Even though trusted users are assigned the roles they need to do their work, you still don't want them to execute scripts they download from the Internet or obtain elsewhere.

A second line of defense is therefore provided by Execution Policies, which define the conditions under which Windows PowerShell loads files for execution. There are four policies: Restricted, AllSigned, RemoteSigned, and Unrestricted. You configure the execution policy used for a server by using the Set-ExecutionPolicy cmdlet. The default is RemoteSigned, which you can verify by using the Get-ExecutionPolicy cmdlet. In this mode, EMS permits the execution of any script created locally and any script downloaded from the Internet, provided the script includes a digital signature. All the scripts that come with Exchange are signed for this purpose (see Table 3-2). The caveat is that any script you attempt to run can contain only Exchange cmdlets that are supported by the role the user holds who invokes the script. Table 3-2 lists the alternate modes together with the potential trade-off in security that you might have to make for each mode.

TABLE 3-2 Windows PowerShell execution policies

Execution Policy mode	Meaning
Restricted	No scripts can be run, even if they are signed by a trusted publisher.
AllSigned	Scripts must be digitally signed by a trusted partner before EMS will run them.
RemoteSigned	EMS will run any script created locally. Scripts that originate outside the system (such as those downloaded from the Internet) cannot be run.
Unrestricted	EMS will run any script. This mode should be used for test environments only.

If you attempt to run an unsigned script that doesn't comply with policy, Windows PowerShell signals that it cannot load the script. Scripts are signed with the Set-AuthenticodeSignature cmdlet, but you need to get a valid certificate first. The certificate can be one you generate yourself or one you buy from a commercial vendor such as VeriSign.

See *http://technet.microsoft.com/en-us/library/bb125017.aspx* for further details of how to generate and apply certificates to sign scripts.

CAUTION!

Obviously, running an Exchange server with an unrestricted execution policy is a bad idea. In fact, you should avoid any deviation from the default policy unless you have an excellent reason to change. For example, you might decide that you want to run scripts you find on the Internet. This might be acceptable if you run the scripts on a test system only, but it's a much better idea to take the time to go through the code to understand exactly what it does before you think of deploying to a production system. Remember that if you edit a script to create a new version on your computer, that version of the script is now considered local and can be run without changing the execution policy. Opening a downloaded script and saving it can lead to unintended consequences, so be sure that you only save a script that you didn't write when you absolutely intend to create a new version.

If you deem it necessary to change the policy, use the Set-ExecutionPolicy command to update the default execution policy on an Exchange 2013 server. For example:

```
Set-ExecutionPolicy –ExecutionPolicy Unrestricted
```

The change to the execution policy is effective immediately. Be sure to test any change you want to make before you enable the change in production because it might break scripts on which you or applications depend. Execution policy is a server-specific setting. However, its setting is recorded in the system registry, and it is possible to use Group Policy to apply the same setting to every server within the organization. To do this, configure Group Policy to set the value of ExecutionPolicy to the desired execution mode. The key is located under:

```
HKLM\Software\Microsoft\PowerShell\1\ShellIds\Microsoft\PowerShell
```

Note that because the setting for the execution policy is held in the system registry, Windows will deny any attempt to update the value unless your account has the privilege to change the system registry.

Profiles

When you start EMS, PowerShell runs a script called Bin\RemoteExchange.ps1 to initialize EMS by loading the Exchange snap-in and defining a set of variables that EMS uses, such as the default scope for Active Directory queries. The script also prints some welcome information for EMS.

If you use EMS frequently, consider creating a profile EMS can load when it initializes a new session. If it finds a profile, PowerShell executes the commands in it before it runs

Exchange ps1 to create the EMS session. This order ensures that you can't interfere with the creation of the EMS session.

I like profiles because they remind me of the convoluted logon command procedures I used to create for OpenVMS. Typical examples of commands included in profiles are the following:

- Define some aliases (shorthand for commands). For example, you could use Set-Alias gmbx Get-Mailbox to use gmbx any time you want to run the Get-Mailbox cmdlet.
- Add one or more directories containing scripts to the path, as discussed earlier.
- Position your session in a specific directory in which you prefer to work.

PowerShell defines a global variable called $Profile to hold the location of your profile. The exact location varies across different versions of Windows. The profile doesn't exist by default, and you might have to create it before you can edit it to add some commands. First, see whether a profile is available for the account you use:

```
Test-Path $Profile
```

If the response is $True, you know that a profile exists. If not, you have to create it with:

```
New-Item –Path $Profile –Type File –Force
```

After you have a profile, you can edit it as follows:

```
Notepad $Profile
```

Here's a simple profile that you could begin with:

```
$env:path = $env:path + ";c:\Scripts"
'You are now entering PowerShell: ' + $env:Username
$StartTime = (Get-Date)
Write-Host "Session starting at $StartTime"
Set-Location c:\temp
```

After you finish updating the profile, save the file and restart EMS to see whether your changes are effective. There are endless possibilities for inventive code to run within a profile.

Active Directory for PowerShell

Active Directory is a huge dependency for Exchange, and it makes a lot of sense to be able to manage Active Directory through PowerShell. This was not always possible, but on Windows Server 2008 R2 SP1 servers, all you need to do is load the Active Directory Module for PowerShell that's installed under Administrative Tools. Assuming that the Active

Directory module is available on a server or client, you can load it into any PowerShell session by using the following command:

```
PS C:\> Import-Module ActiveDirectory
```

The Active Directory module is loaded automatically into EMS on Windows 2012 servers, so you can execute commands against Active Directory data immediately. To get a list of the Active Directory cmdlets, type:

```
PS C:\> Get-Help *-AD*
```

In terms of navigation through the directory structure, Active Directory is represented to PowerShell like files on a hard drive that is referenced as the AD: drive. If your system is joined to a domain, you can then navigate Active Directory. For example, here's how to create a new organizational unit (OU) called Marketing after navigating to the desired location in Active Directory. You can see the same in Figure 3-6.

```
PS C:\> CD AD:
PS AD:\> CD "DC=contoso,DC=com"
PS AD:\DC=contoso, DC=com> MD "OU=Marketing"
```

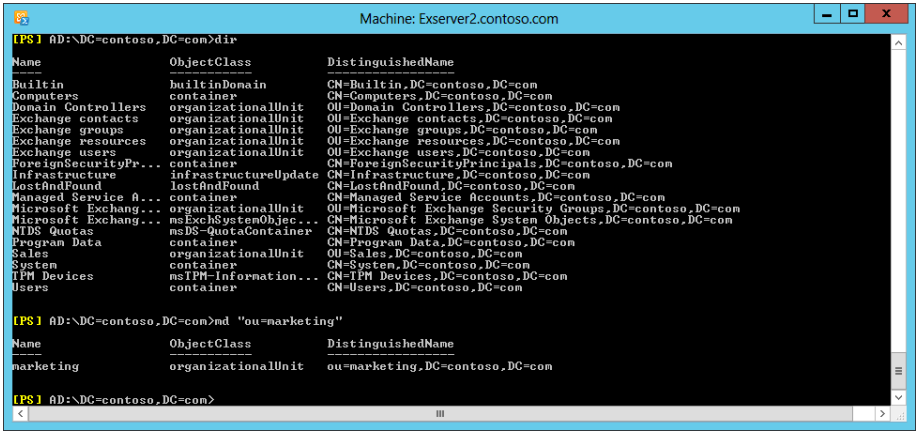


Figure 3-6 Creating a new OU in Active Directory

To compare how much easier it is to access Active Directory data by using the new module, the command to retrieve a list of domain controllers is:

```
PS C:\> Get-ADDomainController  | Format-Table Name, OperatingSystem
```

```
Name                        Operatingsystem
----                        ---------------
CONTOSO-DC07                Windows Server 2012 Standard
```

```
CONTOSO-DC01                Windows Server 2012 Standard
CONTOSO-DC02                Windows Server 2008 R2 Enterprise
```

Another useful example is when you want to scan for inactive Active Directory accounts so that you can clean up the directory. In this command, you scan for any account that has not been logged on to in the past 120 days and report the account name and the date the user last logged on.

```
Search-ADAccount –UsersOnly –AccountInActive –TimeSpan 120 | Format-Table Name,
LastLogonDate
```

You could then disable these accounts by piping the discovered list to the Disable-ADAccount cmdlet. However, this is a dangerous thing to do in an Exchange environment because so many accounts are never logged on to because they are used for purposes such as room and discovery mailboxes.

Another one-liner that is extremely useful on test systems searches for all Active Directory accounts that have an email address and sets the accounts so that the passwords never expire. This gets rid of a lot of annoying prompts you might otherwise encounter because passwords expire!

```
Get-ADUser –Filter {EmailAddress –Like "*@contoso.com"}  | Set-ADUser
–PasswordNeverExpires $True
```

See *http://technet.microsoft.com/en-us/library/dd378937(v=ws.10).aspx* for information about how to perform Active Directory management by using PowerShell for Windows 2008 R2.

Setting the right scope for objects in a multi-domain forest

When you start EMS, Exchange sets the default scope for queries performed against Active Directory to the domain to which the server belongs. This is fine if you operate a single-domain forest, but it is definitely not if you have to manage objects in a multi-domain forest because it means that any query you perform will return only objects from the local domain. To control the scope for Active Directory objects, use the Set-ADServerSettings cmdlet. Set the ViewEntireForest parameter to be $True (to see the entire forest) or $False (to see just the objects owned by the default domain). The logical place to do this is in your personal PowerShell profile. For example:

```
Set-ADServerSettings -ViewEntireForest $True
```

You can also use this command to point to a particular domain controller to retrieve Active Directory data. For example:

```
Set-ADServerSettings –PreferredServer 'DC1.contoso.com'
```

If you do not want to set your scope to the entire forest, a partial workaround is to specify a global catalog server in the remote domain to use for the query. Another way of forcing EMS to operate on a forest-wide basis is to specify the –IgnoreDefaultScope parameter for cmdlets such as Get-Mailbox. This parameter tells EMS to ignore the default recipient scope setting for EAC (typically the domain into which a server is installed) and use the entire forest instead. For example, if you wanted to set up a batch of mailboxes to move from an Exchange 2007 server to Exchange 2013 that used accounts in multiple domains, you could use a command like this:

```
Get-Mailbox –Server 'Exchange2007' –ResultSize Unlimited –IgnoreDefaultScope |
New-MoveRequest -TargetDatabase 'Mailbox Database 1002' –BatchName 'Move Group from
Exchange 2007'
```

The natural question at this point is whether changing the scope for Active Directory queries will affect how you work with EMS. The answer is yes because when you set a forest-wide scope, EMS fetches data from across the forest rather than from the local domain. Unless you use parameters to focus on particular groups of objects, such as specifying that you want to work with the mailboxes from one server, you will probably have to wait longer for a response. This is because you will ask EMS to process cmdlets that deal with servers, mailboxes, databases, or other objects across a complete forest rather than with just one domain, but in most cases, the wait is worthwhile because you see the complete picture and do not run the risk of missing something.

Exploring useful EMS examples

A scan of the Internet results in many interesting EMS code snippets that can be usefully employed by an Exchange administrator. This section discusses some good examples. The idea is not to present complete solutions. Rather, I hope to inspire you to experiment with EMS to see just how much value you can get from a few lines of reasonably straightforward code. After all, if you can do a lot of work in a couple of lines that take just a few minutes to type in and get running, think of how much you can do if you really set your mind to exploiting EMS!

Before reviewing the examples of EMS in use, I have two specific pieces of advice for the aspiring EMS aficionado. Because this book is emphatically not designed to be a reference guide for EMS, if you think that you will become heavily involved with EMS, purchase a copy of *Microsoft Exchange 2013 PowerShell Cookbook*, Second edition (Packt Publishing, 2013). The book is packed full of guidance, tips, and programming examples that are extremely useful for both on-premises and Exchange Online administrators.

Second, many of the Exchange MVPs provide an extremely valuable service to the Exchange community by publishing what become de facto standards for how to write a script to solve certain problems. You should download these scripts and use them as a

starting point for understanding just how to approach writing industrial-strength EMS code. I would also bookmark their websites and keep up to date with their activities so that you can learn from their future work. At the risk of offending others, among my favorite sites are:

- **Pat Richard (*http://www.ehloworld.com/*)** Features a great script (Send-NewUserWelcome.ps1) that shows how to build a welcome message to new Exchange users on a scheduled basis.
- **Andy Grogan (*http://www.telnetport25.com*)** Look at his script for automating the setup of an Exchange lab environment.
- **Mike Crowley (*http://mikecrowley.wordpress.com/*)** Contains a nice script to report on the proxy addresses assigned to email users.
- **Steve Goodman (*http://www.stevieg.org/*)** Shows an extremely useful Exchange environment report, a comprehensive overview of lots of information about your Exchange organization; output in HTML format.
- **Paul Cunningham (*http://exchangeserverpro.com/*)** Offers the best mailbox report script around (Get-MailboxReport.ps1). Paul also maintains a nice server health monitoring script that generates and sends an HTML format message to administrators on a regular basis.

These scripts can be downloaded from these sites; the code is fully revealed and is easily adapted to meet any particular needs that exist in your environment. New sites that feature great tips appear all the time, and I'm sure you will accumulate your own list of go-to people you consult when you meet a problem. In the meantime, look at some examples to get started with EMS.

Looking for large folders

The first example shows how to discover users who might be suffering from performance problems because they have very large folders in their mailboxes. The number of items that is considered bad has grown over time in line with the updates Microsoft has made to tune the database schema. With Exchange 2000 or Exchange 2003, the danger mark is around 5,000 items. The threshold increases to 20,000 with Exchange 2007 and leaps to 100,000 for Exchange 2010 onward. The client used is also important because Outlook 2010 and Outlook 2013 are better at dealing with large folders than Outlook 2007 is. Having more than 20,000 items in a folder is evidence of solid pack-rat behavior by anyone, and it marks a folder that probably will never be cleaned out simply because it takes too much effort to explore the contents and decide what should be kept and what should be deleted. Assume

that you want to flag potential issues to users who have more than 5,000 items in a folder. You can use code like this:

```
Get-Mailbox -Server ExServer2 | Get-MailboxFolderStatistics | Where {$_.ItemsInFolder
-GT 5000} | Sort ItemsInFolder -Descending | Format-Table Identity, ItemsInFolder
-AutoSize
```

```
Identity                                                    ItemsInFolder
---------                                                   -----------------
contoso.com/Exchange Users/Redmond, Eoin\I                  5271
contoso.com/Exchange Users/Ruth, Andy\Inbox                 5265
contoso.com/Exchange Users/Andrews, Ben\Inbox               5263
contoso.com/Exchange Users/Pelton, David\Inbox              5230
contoso.com/Exchange Users/Simpson, David\Inbox             5218
contoso.com/Exchange Users/Redmond, Tony\Sent Items         5215
```

Of course, it would be impolite to send a note to these users to remind them that good filing practices lead to clean mailboxes, but you can still think about it!

This code does the following:

- Calls Get-Mailbox to generate a list of all mailboxes located on databases hosted by a server. It is possible to process all mailboxes in an organization by changing the code to Get-Mailbox –ResultSize Unlimited, but such a command will take a long time to process in any organization with more than a couple of thousand mailboxes (though you could use a server-side filter when appropriate).
- Calls Get-MailboxFolderStatistics to extract a count of items in each folder.
- Filters any folder with more than 5,000 items.
- Sorts the filtered folders by descending order.
- Outputs the information.

If you run this command against an Exchange 2010 or Exchange 2013 server, even details of the folders in the dumpster (for example, Deletions) will be shown that are not reported by an Exchange 2007 server.

Outputting a CSV file

Many examples of outputting CSV files from Exchange data use the Export-CSV cmdlet. For instance, here's a two-line script that looks for any mailbox that has an ActiveSync partnership created, which indicates that the user has connected a mobile device to the mailbox by using ActiveSync. An expression is included to force a call to the Get-ActiveSyncDevice

cmdlet to retrieve the count of devices associated with each user. This kind of information is useful when understanding how many people actually connect mobile devices to Exchange!

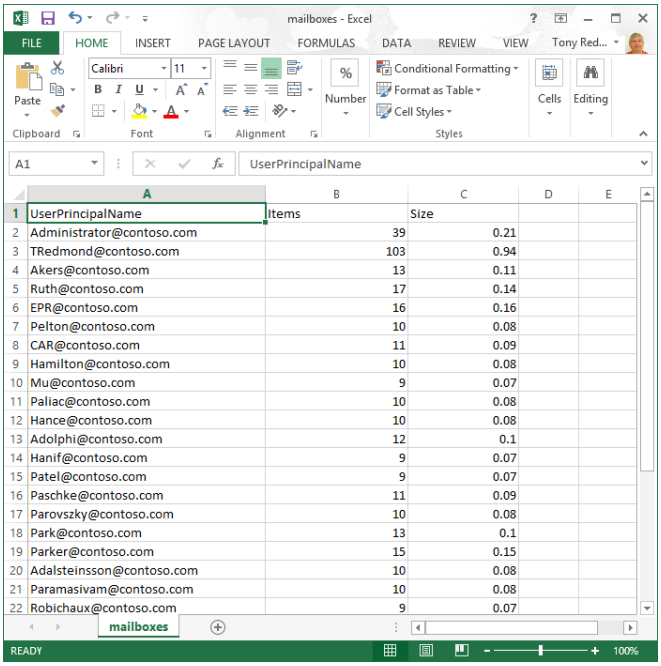
```
$Mbx = Get-CASMailbox -Filter {HasActiveSyncDevicePartnership -eq $True} |
Get-Mailbox
$Mbx | Select DisplayName, UserPrincipalName, @{Name="Devices";Expression=
{(Get-ActiveSyncDevice -Mailbox $_.Identity).Count)} | Export-CSV
"c:\temp\ActiveSync.csv" -NoTypeInformation
```

Export-CSV is great because it takes care of all the formatting issues required to create a valid CSV file that will be recognized by applications such as Excel. However, there are other ways to generate CSV data. This script creates a CSV file you can use to analyze mailbox usage (Figure 3-7). A check in the code limits processing to the mailboxes found in a specific database and ignores anything but user mailboxes. (Objects such as room or arbitration mailboxes are ignored.) This script could take quite a while to finish if there are more than a few hundred mailboxes in the selected database, so be sure to test it on perhaps a smaller group before you launch it to process larger collections.


```
$Outputfile = "C:\temp\Mailboxes.csv"
Out-File -FilePath $OutputFile -InputObject "UserPrincipalName, Items, Mailbox Size"
-Encoding UTF8
$mbx = Get-Mailbox -Database DB2
Foreach ($M in $Mbx)
   {

   if ($M.RecipientTypeDetails -eq "UserMailbox")
        {

        # Fetch information about the mailbox
        $UserMailbox = Get-Mailboxstatistics -Identity $($M.Identity)

       $UserPrincipalName = $M.UserPrincipalName
       $ItemSizeString = $UserMailbox.TotalItemSize.ToString()
       $MailboxSize = "{0:N2}" -f ($ItemSizeString.SubString(($ItemSizeString.
IndexOf("(") + 1),($itemSizeString.IndexOf(" bytes") - ($ItemSizeString.IndexOf("(")
+ 1))).Replace(",","")/1024/1024)
       $ItemCount = $UserMailbox.ItemCount

    #Prepare the user details in CSV format for writing to file and append line
      $UserDetails = $UserPrincipalName + "," + $ItemCount + "," + $MailboxSize
      Out-File -FilePath $OutputFile -InputObject $UserDetails -Encoding UTF8 -append
        }
   }
```

This script generates fairly basic data about mailboxes, and if you scan the Internet, you can find many other approaches to the problem of mailbox reporting, some of which are much better than others. With anything to do with mailboxes, the key is speed because code that is quite good at processing one or two mailboxes might not be smart when confronted with

a few thousand. It's also a good idea to consider what information needs to be output and make sure that the data reported is formatted in a way that is most useful to the reader.



	A	B	C
1	UserPrincipalName	Items	Size
2	Administrator@contoso.com	39	0.21
3	TRedmond@contoso.com	103	0.94
4	Akers@contoso.com	13	0.11
5	Ruth@contoso.com	17	0.14
6	EPR@contoso.com	16	0.16
7	Pelton@contoso.com	10	0.08
8	CAR@contoso.com	11	0.09
9	Hamilton@contoso.com	10	0.08
10	Mu@contoso.com	9	0.07
11	Paliac@contoso.com	10	0.08
12	Hance@contoso.com	10	0.08
13	Adolphi@contoso.com	12	0.1
14	Hanif@contoso.com	9	0.07
15	Patel@contoso.com	9	0.07
16	Paschke@contoso.com	11	0.09
17	Parovszky@contoso.com	10	0.08
18	Park@contoso.com	13	0.1
19	Parker@contoso.com	15	0.15
20	Adalsteinsson@contoso.com	10	0.08
21	Paramasivam@contoso.com	10	0.08
22	Robichaux@contoso.com	9	0.07

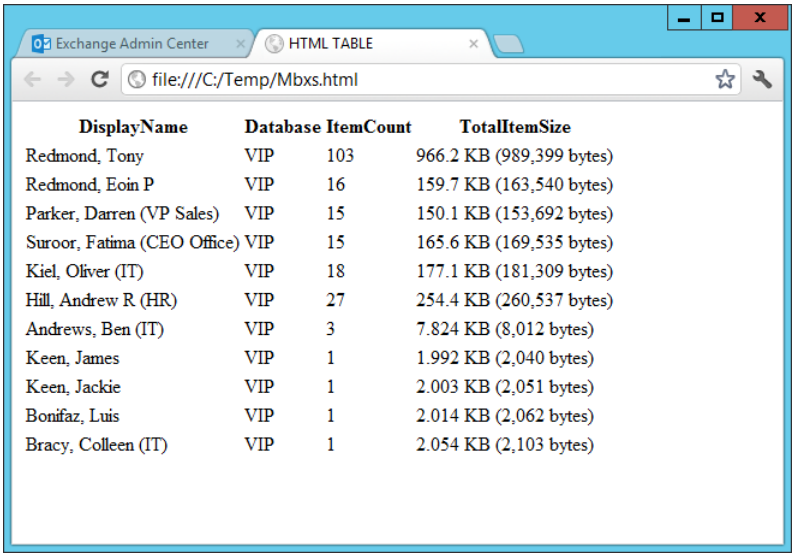
Figure 3-7 User mailbox CSV data

Creating a report in HTML

PowerShell is flexible in terms of processing output. Generated reports can show management and others the kind of work that servers do. The typical reports EMS generates are plaintext. You can also generate HTML reports by piping objects through the ConvertTo-HTML cmdlet. (The Out-HTML cmdlet at *http://poshcode.org/1612* is also useful for generating HTML content.) This example explores how to generate a useful report that shows mailboxes that have exceeded their storage quota. You could use a report like this to check proactively for users who are experiencing problems with their quota and perhaps allocate them some additional quota to enable them to resume working. The output is shown in Figure 3-8.

```
Get-Mailbox -Database VIP | Get-MailboxStatistics | Sort TotalItemSize -Descending |
ConvertTo-HTML DisplayName, Database, ItemCount, TotalItemSize > C:\Temp\Mbxs.html
```

You can enhance the output further by formatting the HTML with a style sheet or adding other information such as the date and time of the report. I leave that as an exercise for the reader.



DisplayName	Database	ItemCount	TotalItemSize
Redmond, Tony	VIP	103	966.2 KB (989,399 bytes)
Redmond, Eoin P	VIP	16	159.7 KB (163,540 bytes)
Parker, Darren (VP Sales)	VIP	15	150.1 KB (153,692 bytes)
Suroor, Fatima (CEO Office)	VIP	15	165.6 KB (169,535 bytes)
Kiel, Oliver (IT)	VIP	18	177.1 KB (181,309 bytes)
Hill, Andrew R (HR)	VIP	27	254.4 KB (260,537 bytes)
Andrews, Ben (IT)	VIP	3	7.824 KB (8,012 bytes)
Keen, James	VIP	1	1.992 KB (2,040 bytes)
Keen, Jackie	VIP	1	2.003 KB (2,051 bytes)
Bonifaz, Luis	VIP	1	2.014 KB (2,062 bytes)
Bracy, Colleen (IT)	VIP	1	2.054 KB (2,103 bytes)

Figure 3-8 Viewing the HTML version of the mailbox report

It's worth noting that when you run the Get-MailboxStatistics cmdlet, you force EMS to make a remote procedure call (RPC) to the Information Store to retrieve the latest data for the mailboxes (individual, database, or server). The information is completely up to date and reflects the exact state of the mailbox rather than cached data that could be a couple of hours old. The Store caches information about mailbox quotas and updates the cache every two hours to avoid the overhead of the I/O that it would otherwise need to generate to check quotas every time a user attempts to send a message or to check that a mailbox can accept a new message.

TROUBLESHOOTING

Users report that they've deleted messages but still exceed quota

Given the dynamic flow of messages in and out of mailboxes, it's likely that a small difference exists between the cached data and the actual state. This sometimes causes confusion when a user reports that she has exceeded quota and can't send mail even though she has deleted many messages, and she has to wait until the Store refreshes its cache to determine the new mailbox size and respect the fact that she has reduced the size under quota. If this becomes a problem and users complain that Exchange takes too long before it allows them to resume email activity, you can amend the system registry to force Exchange to refresh the cache more often with the caveat that more frequent refreshes impose an extra overhead on the server. See *http://technet.microsoft.com/en-us/library/aa996988(EXCHG.80).aspx* for details.

Verbose PowerShell

Usually, EMS gets on with whatever you ask it to do and doesn't give any indication of the processing it performs in the background. You ask for a new mailbox to be created, and it's created, or some problem occurs that stops the command from executing. If the problem originates with an error introduced by the user, such as an error in syntax or attempting to do something that doesn't make sense, such as creating a mailbox in a database that doesn't exist, you can just fix the problem and try again.

Sometimes you need to know exactly what EMS does to help track down a problem, perhaps to provide information to Microsoft support to help them figure out what's going on in your Exchange deployment. You might just want to know what's happening when you execute a command. In either case, you can add the /verbose switch to a command to have PowerShell generate details of exactly what it does as it proceeds. Figure 3-9 shows some of the output when the New-MailboxDatabase cmdlet is used to create a new mailbox database. You can see how EMS validates the context within which it is executing, including checks to locate a global catalog server, validate RBAC authorization, and confirm that the mailbox database doesn't already exist.

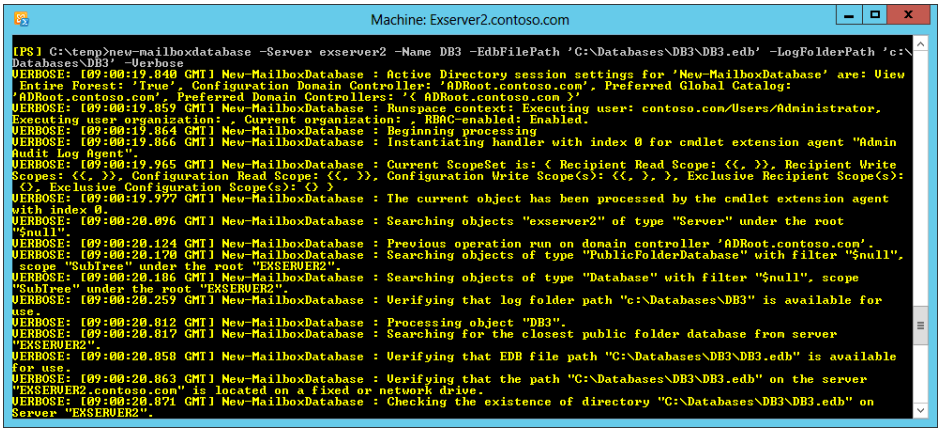


Figure 3-9 Examining some verbose PowerShell output

Controlling access to Exchange

EMS is a great way to get work done with Exchange as long as you don't mind grappling with the command-line interface. If no control were exerted, you could do massive damage to an Exchange organization with EMS, such as selecting all the mailboxes in a database and removing them with a single line of code. Only the people who need to control the full scope of the organization should be able to take such drastic action. Traditionally, control is

given through permissions and privileges. Exchange takes a different approach and adopts the RBAC model. All administrators need a solid grounding in RBAC and its implementation in Exchange, and that's the next subject of discussion.

CHAPTER 4

Role-based access control

RBAC basics 136
Roles 140
Role groups 143
Scopes 145
Working with RBAC 147
Role group management 155
Role assignment 156
Assignment policies 167
Implementing a split permissions model 170
Figuring out RBAC 171
On to management 172

There's no doubt that the introduction of the Database Availability Group (DAG) was the most technologically interesting and useful advance in Microsoft Exchange Server 2010; when everything is stripped away, the implementation of a brand-new permissions model based on the concept of role-based access control (RBAC) exerted the most pervasive influence across all parts of the product. Simply put, without RBAC, Exchange 2013 could not function the way it does in on-premises, cloud, and hybrid environments. RBAC is therefore the most important element for administrators to master.

RBAC is an approach to restricting access to systems and data based on the need of administrators and users to do their jobs. In other words, you should have access only to information you need rather than access to a lot of data you don't need. RBAC is not a new concept, and it has been implemented in other operating systems and applications, including HP-UX, Linux, Oracle DBMS, SAP R/3, and others. The big difference between role-based access control systems and more traditional permission-based systems is that access is granted according to a person's job role rather than according to his user account. In other words, if your current job requires the ability to reset the password for other user accounts, you can be given membership in the appropriate role group so you can perform that function. When you no longer need that ability, you can leave the role group. Joining and leaving role groups enables you to perform functions specific to a job without affecting the rights assigned to you as an individual user.

Earlier versions up to and including Exchange 2007 use Windows access control lists (ACLs) to grant permissions to users and administrators to enable them to do their work. ACLs are an effective way to manage permissions for environments that are well structured and do not tend to change much over time. Microsoft Windows itself and many of the applications that run on Windows fall into this category. Exchange is a little more dynamic and spans hundreds of thousands of objects in its largest deployments. Tracking and updating

ACLs across such a dynamic environment for so many objects is a challenge, and real-world experience has revealed that ACLs are often updated incorrectly, can be difficult to understand and therefore debug when things go wrong, and are prone to disruption through changes applied by software upgrades or updates. Another aspect is that the conventional model of creating, updating, and setting permissions on objects often doesn't translate well into the task-driven environment that exists within an application such as Exchange, in which permissions have to support scenarios such as allowing help desk personnel to reset a user's password without being able to affect any other setting of the user's account. The granularity available in RBAC addresses this need. The value of that granularity is now well proven in Exchange 2010, Exchange 2013, and Office 365.

Even if some problems are encountered as administrators learn about RBAC, the way Microsoft has implemented RBAC is relatively easy to understand. Over time, RBAC will avoid complexity and ease management, especially now that Exchange Administration Center (EAC) includes the necessary user interface that enables administrators to work with the fundamentals of RBAC, such as role definitions, role groups, and role assignments, without resorting to Windows PowerShell. In addition, RBAC should simplify matters for administrators enormously by removing the need to grapple with ACLs on a daily basis. Instead, administrators should be able to assign the ability to perform tasks by adding users to role groups, leaving RBAC to do all the necessary ACL maintenance in the background.

It is impossible to overstate the influence of RBAC throughout Exchange. It would require a 300-page book to cover every aspect of RBAC in depth. This chapter does not attempt to present such comprehensive coverage. Instead, the basics are set out to get you started and encourage you to investigate further to determine how best to apply RBAC to help you manage your organization.

RBAC basics

RBAC in Exchange is often described in the form of a triangle (Figure 4-1) to show how roles, role groups, scopes, and assignments fit together.

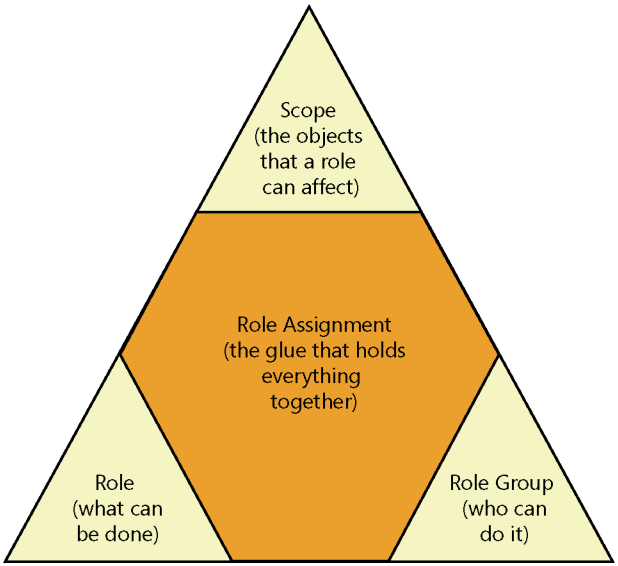


Figure 4-1 The RBAC triangle

These are the major elements of RBAC as implemented in Exchange:

- **Management role** A collection of role entries that define the set of cmdlets and parameters a user can run. For example, the Mailbox Import Export role permits users to import or export mailbox data to and from PSTs.

- **Management role group** A container for a group of management role entries that collectively enable a user to function in a role such as recipient management. Exchange includes a default set of management role groups, and you can define new management role groups to meet specific needs that are not served by a default role group. For example, the Discovery Management role group allows members to execute discovery searches. The EAC refers to management role groups as admin roles.

- **Management role assignment** The ability to assign a management role to an individual user or to the members of a role group (universal security group). You make a management role assignment through the Permissions section of the EAC or by running the New-ManagementRoleAssignment command.

- **Management role assignment policies** Management role groups predominantly intended for use by administrators to enable them to perform administrative tasks such as recipient management. The ability of users to work with personal data is controlled by management role assignment policies. Out of the box, Exchange provides a default role assignment policy that defines how users can update the information in their profile (contact phone numbers, display name, and so on) and distribution

group membership. The default role assignment policy is automatically assigned to users when their mailbox is created or moved to an Exchange server unless another role assignment policy is explicitly assigned. A mailbox can only be assigned a single role assignment policy at a time.

- **Management role scope** The definition of the scope or the collection of objects with which a management role can work. A role such as Organization Management has a scope of the complete organization because the users who hold this role have to be able to manage any object in the entire organization. Other roles might be restricted to a particular scope such as an organizational unit (OU) in Active Directory to enable a fine granularity of management operations, such as the ability to manage mailboxes that belong to a certain region. Exchange 2013 also supports scopes based on servers and databases to enable specific administrators to manage certain objects.

- **Management role entries** Permit access to one or more cmdlets to enable a user to perform a certain task. For example, access to the New-Mailbox cmdlet enables a user to create a new mailbox. It is possible to restrict role entries to selected parameters for a cmdlet. A management role is composed of one or more management role entries.

Another way of understanding RBAC is to look at it from the perspective of the work someone does with Exchange. This will be as an administrator or as an end user. The methods RBAC uses to associate the rights the two groups need to do their work are as follows:

- Administrators and other specialist users who have to perform operational tasks with an Exchange server gain the rights to do their work through membership in appropriate role groups. Each role group consists of a number of roles. To give an administrator permission to do something, just assign him the correct management role by putting him in the appropriate role group.

> **Note**
> **The Organization Management role group is the most powerful because it includes nearly every role available to Exchange (with some exceptions).**

- Users don't need to be granted membership in role groups to be able to interact with Exchange because control of their data (mailbox and mailbox settings) is granted through the default management role assignment policy. That's a long and complicated term to explain default settings. You can access the default management role assignment policy through the user roles section under Permissions in EAC.

Before we plunge into the details of what roles, assignments, and policies mean, Table 4-1 helps establish a context for the discussion by associating various tasks different individuals perform in an Exchange organization with the role group that provides access to the permissions required to execute each task.

TABLE 4-1 Linking role groups to tasks

Task	Role Group required	Notes
I want to be the manager of the complete Exchange organization	Organization Management	Some roles have to be explicitly delegated before even a member of the Organization Management group can perform a task. The need to assign the Mailbox Import Export role to an account to gain access to the cmdlets to import or export mailbox data is the best example.
I want to be able to see the objects in the Exchange organization, but I don't need to edit anything.	View-Only Organization Management	This role enables its holders to view details of configuration objects (servers, connectors, and so on) and recipients anywhere in the organization.
I want to be able to manage mailboxes and distribution groups.	Recipient Management	Members of this role group can create, edit, and delete any mail-enabled object except public folders.
I want to be able to help users maintain the settings for their mailboxes.	Help Desk	The Help Desk role group includes the User Options and View-Only Recipients roles. This set of roles might limit the effectiveness of the Help Desk role group in some companies, which is why you can modify role groups to add new roles to expand what the role group members are allowed to do.
I want to be able to manage Exchange server configuration settings.	Server Management	Members of this role group are not able to manage recipient objects unless they are also members of the Recipient Management role group. Customizations are possible to restrict the ability to manage specific servers or databases.
I need to be able to perform discovery searches and respond to legal actions.	Discovery Management	This role group also enables its members to manage the process of putting mailboxes on an in-place hold.
I need to be able to manage public folders.	Public Folder Management	Members of this group can use the public folder management options in EAC to manage public folders.

Task	Role Group required	Notes
I need to manage different aspects of compliance across the organization.	Records Management	This role group allows its members to manage administrative auditing, message tracking, journaling, retention policies and tags, message classifications, and transport rules.
I need to manage the Unified Messaging servers and set up objects such as dial plans.	UM Management	This role group enables administrators to manage the Exchange Unified Messaging application (if deployed within the organization).

Now that you have some idea of how RBAC might affect the work administrators do and are familiar with the formal definitions of the terms you'll meet, consider what these entities mean in practical terms.

Roles

Exchange 2013 includes 85 built-in roles that are designed to cover the majority of administrative and user tasks performed in Exchange organizations. This is an increase over the 69 built-in roles included in Exchange 2010 SP1, with the new roles being introduced to support the management of new objects, such as team mailboxes, or to accommodate greater flexibility in management, such as enabling users to manage the apps they want to add to Outlook Web App. The majority of the roles cover administrative roles, from taking care of address lists to managing public folders, and there is a small set of roles (all prefixed with My) that are used to assign rights to users to maintain their mailbox settings and other options. You can see the complete collection of roles with the following command:

```
Get-ManagementRole
```

A role can be broken down into a set of cmdlets and parameters that Exchange makes available to anyone who holds the role. In RBAC terms, the components of a management role are called management role entries, each of which identifies a single cmdlet and its parameters, a script, or a permission. The parameters for a cmdlet can be restricted so that only certain functionality is made available through a management role entry. If a role doesn't include a role entry for a cmdlet, that cmdlet is unavailable to the user who holds the role.

Some of the roles included with Exchange are very simple. For instance, if you look at the Reset Password role by running the Get-ManagementRoleEntry command to return its details, you see that it contains just one role entry:

```
Get-ManagementRoleEntry 'Reset Password\*'
```

```
Name                  Role                 Parameters
--------              ----                 ----------
Set-Mailbox           Reset Password       {Password, ResetPasswordOnNextLogon}
```

You might be surprised to see that the role is quite so simple, but in fact when you think about what permissions you need to be able to set someone's Active Directory password, all you need is the ability to run the Set-Mailbox command with two parameters: the new password and a flag to say whether the user has to reset that password when she next logs on. If you look at the management role definition for the Reset Password role, that's exactly what Exchange allows the people who are assigned this role to do.

Because roles have to deal with all aspects of Exchange administration, they can be much more complex. For example, the Message Tracking role specifies a set of 19 role entries describing the cmdlets necessary to search through message tracking logs:

```
Get-ManagementRoleEntry 'Message Tracking\*'
```

```
Name                                  Role               Parameters
----                                  ------             ----------
Write-AdminAuditLog                   Message Tracking   {Comment, Confirm,
Debug, DomainController, ErrorAction, ErrorV...
Set-OrganizationRelationship          Message Tracking   {DeliveryReportEn-
abled, Identity}
Set-AvailabilityConfig                Message Tracking   {Confirm, Debug,
DomainController, ErrorAction, ErrorVariable, ...
Set-ADServerSettings                  Message Tracking   {ConfigurationDomain-
Controller, Confirm, Debug, ErrorAction, Er...
Search-MessageTrackingReport          Message Tracking   {BypassDelegateCheck-
ing, Confirm, Debug, DomainController, DoNo...
Resume-MailboxExportRequest           Message Tracking   {Confirm, Debug,
DomainController, ErrorAction, ErrorVariable, ...
Remove-AvailabilityAddressSpace       Message Tracking   {Confirm, Debug,
DomainController, ErrorAction, ErrorVariable, ...
New-OrganizationRelationship          Message Tracking   {DeliveryReportEn-
abled, DomainNames, Name}
Get-Recipient                         Message Tracking   {Anr, BookmarkDisplay-
Name, Credential, Database, Debug, DomainC...
Get-MessageTrackingReport             Message Tracking   {BypassDelegateCheck-
ing, Debug, DetailLevel, DomainController, ...
Get-MessageTrackingLog                Message Tracking   {Debug, DomainCon-
troller, End, ErrorAction, ErrorVariable, Even...
Get-Mailbox                           Message Tracking   {Anr, Arbitration,
Archive, Credential, Database, Debug, Domain...
Get-ExchangeServerAccessLicenseUser   Message Tracking   {Debug, ErrorAction,
ErrorVariable, LicenseName, OutBuffer, Out...
```

```
Get-ExchangeServerAccessLicense       Message Tracking   {Debug, ErrorAction,
ErrorVariable, OutBuffer, OutVariable, Sta...
Get-ExchangeServer                    Message Tracking   {Debug, Domain,
DomainController, ErrorAction, ErrorVariable, I...
Get-DomainController                  Message Tracking   {Credential, Debug,
DomainName, ErrorAction, ErrorVariable, For...
Get-AvailabilityAddressSpace          Message Tracking   {Debug, DomainCon-
troller, ErrorAction, ErrorVariable, Identity,...
Get-ADServerSettings                  Message Tracking   {Debug, ErrorAction,
ErrorVariable, OutBuffer, OutVariable, Ver...
Add-AvailabilityAddressSpace          Message Tracking   {AccessMethod, Con-
firm, Credentials, Debug, DomainController, E...
```

Some of the cmdlets included in the Message Tracking role don't appear to have very much to do with interrogating the message tracking logs to determine the routing of a message within an organization. This is because it is convenient to use some roles in different role groups to enable administrators to perform tasks without having to create a profusion of relatively similar roles.

If you recall the discussion about the Exchange Management Shell (EMS) and its relationship with RBAC in Chapter 3, "The Exchange Management Shell," it should now be clear how cmdlets are made available to users and loaded into an EMS session during initialization. In this case, if a user holds the Message Tracking role through membership in a role group, EMS loads the 19 cmdlets and their available parameters set as listed earlier to make them available during the session. RBAC definitions differ when you manage an Exchange Online tenant. You don't need access to many cmdlets because Microsoft does a lot of the work with servers and databases that usually has to be done by an on-premises administrator. For instance, you won't see any trace of a New-DatabaseAvailabilityGroup cmdlet when you connect to Exchange Online.

Individual role entries are named through a combination of the role group name and the cmdlet, so including the asterisk wildcard character in the command instructs Exchange to return all the cmdlets assigned to the role group. To return details of a specific cmdlet, including the parameters that can be used, include the cmdlet name:

```
Get-ManagementRoleEntry 'Move Mailboxes\Get-Recipient' | Format-List
```

You can also find the list of roles to which a specific cmdlet is assigned by using the Get-ManagementRole cmdlet. For example, to list all roles that can update mailbox information with the Set-Mailbox cmdlet, use:

```
Get-ManagementRole -Cmdlet 'Set-Mailbox'
```

INSIDE OUT **No permissions**

How do you know whether your assigned roles allow you to perform a task with Exchange? If you attempt to use a cmdlet in EMS and EMS does not respond, it's a good indication that the cmdlet is not included in any of the roles that you possess. The same is true of EAC: if you don't see functionality showing up, it's because EAC has assessed the sum of your rules and displayed all of the functionality that you're allowed to see. Finally, if you run a script and see the error shown below, it's simply because the script attempted to use a command that isn't available to you.

```
The term 'xxxx' is not recognized as the name of a cmdlet, function, script
file, or operable program. Check the spelling of the name, or if a path was
included, verify that the path is correct and try again.
```

Role groups

Roles can be assigned on an individual basis or on a group basis. Although roles provide the granularity necessary to break down all the tasks a typical Exchange administrator performs, it would be far too complex to assign tasks through individual roles. Role groups provide a convenient method to gather the roles necessary to perform higher-level tasks such as Mailbox Search and avoid the need to assign the 11 roles that would otherwise be required. It's much easier to manage the assignment of a single role group than it is to manage individual role assignments, and it's less likely that administrators will make mistakes and create security problems when they manage RBAC through role groups.

Users are assigned roles by making them members of role groups. In effect, a role group describes a high-level set of tasks that you expect a certain type of administrator to perform. For example, technicians working on a help desk need to be able to view and update details of recipients, but you probably don't want them to mess with a send connector or transport rule. The role group defined for the help desk contains all the roles (and therefore access to all the cmdlets) that are necessary to do the work required by this role and no more.

Role groups provide much of the foundation of the RBAC implementation in Exchange. You can see the built-in role groups (and any that you have subsequently created) with:

```
Get-RoleGroup
```

Behind the scenes, every role group is represented by a universal security group (USG) held in the Microsoft Exchange Security Groups OU in Active Directory (Figure 4-2). The USGs are flagged to Exchange so that it knows that RBAC uses these groups. When necessary, the

existing Exchange 2007 ACLs are copied to a role group when the first Exchange 2010 or Exchange 2013 server is installed in an organization that contains Exchange 2007 servers to enable the role group to perform its management function.

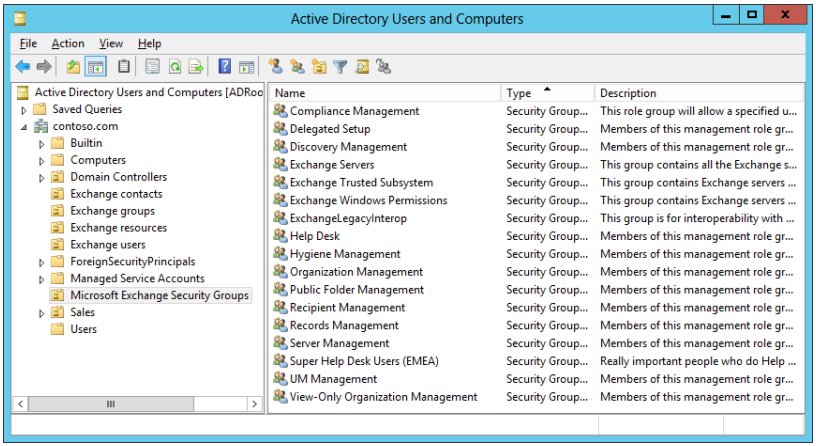


Figure 4-2 The set of USGs RBAC uses

A key difference between the USGs that instantiate role groups and other USGs is that you can manage role groups (and, by default, their underlying USGs) from EAC and EMS. The Super Help Desk Users (EMEA) USG shown in Figure 4-2 is not one of the standard USGs created during the installation of Exchange for RBAC. You won't see it in your Exchange deployment because it's a USG Exchange created when I created a new role group for my organization. This underlines the point that there is a one-to-one mapping between role groups and USGs.

INSIDE OUT Delegating roles that are not assigned by default

Apart from having access to the vast majority of roles defined in an organization, users in the Organization Management role group can delegate the roles they don't possess by default. The Mailbox Import Export role is the best example of a role that is not held by users who are part of the Organization Management role group but which a member of that role group can delegate to himself if required. The reason Mailbox Import Export is not held by members of the Organization Management role group is simple: user mailbox contents should be protected against unnecessary access, so you have to take a deliberate step to grant access to mailboxes before you are allowed to run the cmdlets to export or import mailbox data.

Despite the fact that USGs underpin roles, it is a mistake to assume that you could just use the Active Directory Users and Computers console to add user accounts to the USGs to assign roles. Behind the scenes, Exchange notes the role assignments, and adding a user to a USG is not sufficient; it will cause unpredictable results in the future. The Organization Management and Delegated Setup roles are also unique in that they are assigned Active Directory ACLs in addition to Exchange permissions because of the need to have these ACLs to perform tasks that affect non-Exchange parts of Active Directory such as installing servers. The vast majority of the work done by users holding roles to manage the various aspects of Exchange is facilitated by RBAC, so they don't need to be assigned ACLs. See Chapter 3 for more information about how role group membership determines the cmdlets available to a user when she initializes an EMS session.

You can find information about the set of default role groups and role assignments included in Exchange 2013 at *http://technet.microsoft.com/en-us/library/dd638077.aspx*.

Role groups and assignments can change over time as Microsoft tweaks RBAC through updates to Exchange. Each role group spans a number of administrative roles that provide granularity for task assignment. The names of the role groups are reasonably descriptive of the tasks you could expect someone assigned to the role group to undertake. The Microsoft goal is to provide a set of role groups that meet the needs of the majority of customers, but you can customize a role group (for example, to remove or add a task) or create a new role group if the default set doesn't meet your requirements. Again using the default Help Desk role as an example, you might decide that you want these users to see message queues and perform message tracking. In this case, you could customize the Help Desk role group to add the Transport Queues and Message Tracking roles.

It is also possible to assign a specific role to a user or group without placing that user or group in a role group. However, as mentioned earlier, Microsoft doesn't recommend taking this approach because you are likely to accumulate a proliferation of role assignments that become difficult to monitor and manage.

Scopes

All RBAC roles have scopes to tell Exchange which objects can be accessed and updated by holders of the role. The implicit (preset) scopes Exchange supports are:

- **Recipient read scope** Determines the Active Directory recipient objects (users, groups, contacts) the holder can read. When Organization is shown in the recipient read or write scope, the holder has access to every object in the organization. The default role assignment policy for users has Self as its scope, meaning that the holder can update properties of his mailbox.

- **Recipient write scope** Determines the Active Directory recipient objects the holder can update, create, or delete.
- **Configuration read scope** Determines the Active Directory configuration objects (servers, databases, connectors, and so on) the holder can read. When OrganizationConfig is shown in the configuration read or write scope, the holder can access any object in the Exchange configuration data held in Active Directory.
- **Configuration write scope** Determines the Active Directory configuration objects the holder can create, update, or delete.

Logically, the implicit write scope assigned to a role is always equal to or less than the read scope to ensure that the holder of a role can never modify objects she cannot see.

In this example, you look at the scope of the Move Mailboxes role, which is required by anyone who wants to move a mailbox between databases:

```
Get-ManagementRole 'Move Mailboxes' | Format-List *Scope*
```

```
ImplicitRecipientReadScope   : Organization
ImplicitRecipientWriteScope  : Organization
ImplicitConfigReadScope      : OrganizationConfig
ImplicitConfigWriteScope     : OrganizationConfig
```

The most important scopes are the implicit write scopes because these define the objects the cmdlets covered by the role can update. In this case, to move mailboxes, you need the ability to update the mailbox object afterward, so the recipient write scope is organization-wide. You also see that the role can read information from the organization's configuration so that you can select any database in the organization as a target for the mailbox move. Note that you cannot change the implicit scopes for a management role because these scopes always apply and cannot be overridden.

For more information on how Exchange uses scopes, see *http://technet.microsoft.com/en-us/library/dd335146.aspx*.

If you create a new RBAC role, it has to be the child of an existing RBAC role and automatically inherits the scope of the parent role unless you define a new scope. You see how this happens in Chapter 6, "Groups and other objects," when that chapter discusses how to allow users to update distribution groups they own without allowing them to create new groups. You can create new scopes as you create new roles, but a better technique is to define a scope once so that it is available to multiple roles. Exchange provides the New-ManagementScope cmdlet for this purpose. These custom scopes are called explicit because part of their definition is a filter setting out exactly which objects fall under the scope.

Scopes can be created so that roles are restricted to operate against specific servers, an OU, or a recipient filter such as the members of a distribution group. For example, this command creates a new scope based on a distribution group called Company Officers. You can even use a dynamic distribution group for this purpose.

```
New-ManagementScope -Name 'Company Officers' -RecipientRestrictionFilter {MemberOf-
Group -eq "Company Officers"}
Get-ManagementScope 'Company Officers'
```

```
RecipientFilter                      : MemberOfGroup -eq 'DC=Company Officers'
ServerFilter                         :
DatabaseFilter                       :
ScopeRestrictionType                 : RecipientScope
Exclusive                            : False
ExchangeVersion                      : 1.10 (14.1.90.0)
Name                                 : Company Officers
DistinguishedName                    : CN=Company Officers,CN=Scopes,CN=RBAC,
CN=contoso,CN=Microsoft Exchange,CN=Services,CN=Configuration,DC=contoso,DC=com
Identity                             : Company Officers
Guid                                 : 906b4ad8-2e7d-4590-bb01-973f32043207
```

After they are created, scopes can be assigned to role groups by using the –CustomConfigWriteScope (for server and database scopes) and –CustomRecipientWriteScope parameters (for recipient-based scopes).

Working with RBAC

After its initial introduction in Exchange 2010, RBAC was quickly tagged as being overly complex, perhaps because RBAC had to be managed through EMS. In fact, RBAC is more powerful and easier to use than messing around with ACLs on objects, but it is difficult for a component to lose a reputation after it's been acquired. Exchange 2013 doesn't use the RBAC phrase. Instead, EAC contains a Permissions section, which is a much better and more user-friendly term. The section is broken down into admin roles, user roles, and Outlook Web App policies. You see the first two in this chapter. Admin roles refer to the management role groups discussed so far, whereas user roles refer to the role assignment policies that govern what users are allowed to do with Exchange. Out-of-the-box Exchange 2013 comes with just the default role assignment policy.

Figure 4-3 shows how EAC shows admin roles (or management role groups). On the right side of the screen, you see a description of the selected role group and the management roles that are assigned to the role group—the set of management roles that represent the permissions a user can use after he is added to the management role group (or admin role—it is confusing when terminology changes between versions).

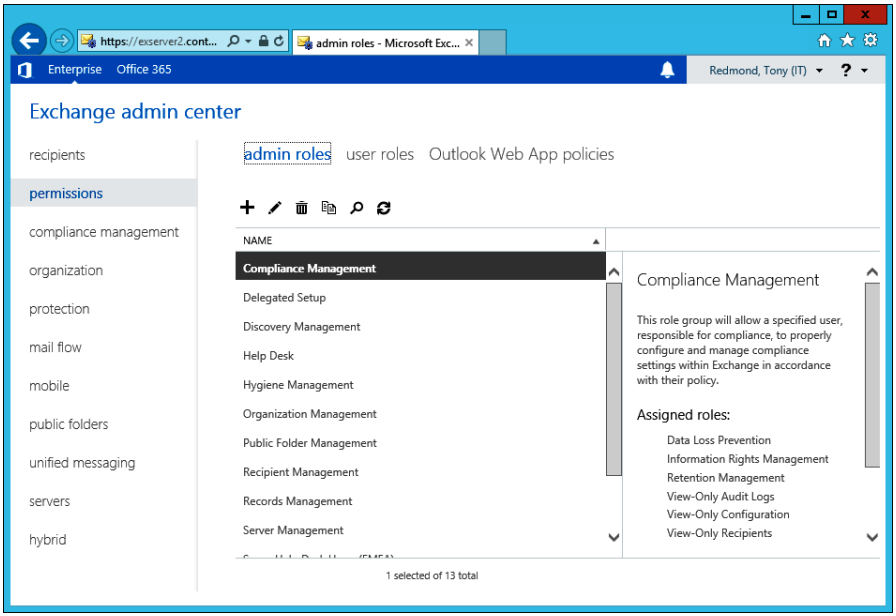


Figure 4-3 The Permissions section of EAC

If you click Edit for the selected management role group, EAC reveals some more information about the role group, including the users who are members of the group. Figure 4-4 shows the properties of the Organization Management role group, the most capable in terms of the number of roles that are included in the group and therefore the most powerful in terms of the number of cmdlets the members of the group are allowed to execute. On the left side of the screen, you see some descriptive information about the group and the information that the scope for the group is Default, meaning the entire organization. A list box then contains all the roles that are assigned to the group, starting with Active Directory Permissions and ending with Workload Management (not shown here). An equivalent EMS command to fetch this information is:

```
Get-RoleGroup -Identity 'Organization Management'
```

On the bottom of the rightmost screen, you can see the users who are members of the Organization Management role group. Essentially, these are the enterprise administrators because the roles assigned through membership of this group enable them to work with almost anything in an Exchange organization. To retrieve the membership of the Organization Management role group through EMS, use the command:

```
Get-RoleGroupMember -Identity 'Organization Management'
```

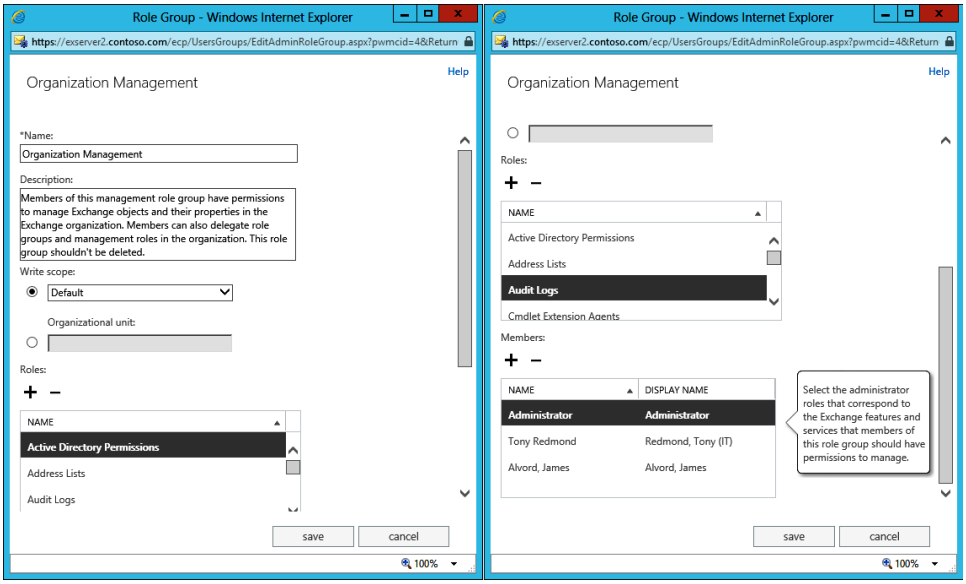


Figure 4-4 Properties of a management role group

Creating a new role group

Now that you understand the connections among roles, role groups, and assignments, consider how to create a new role group and see what happens to instantiate it. First, decide whether you actually need a new role group. The general approach to creating a new role group is as follows:

1. Write down the need for the new role group and why none of the out-of-the-box role groups is appropriate. It's always preferable to use one of the default role groups because creation of new role groups increases the level of complexity within the organization.

2. Determine the list of management roles that should be assigned to the new role group and why the members of the role group need to be able to use these roles.

3. Create the new role group with EAC or by running the New-RoleGroup command and assign the roles incorporated in the role group.

4. Assign users to the new role group.

Most companies have help desks that need access to certain functionality to do their work. Exchange 2013 includes a standard Help Desk role group that includes two management roles: the User Options role, which allows access to the Outlook Web App options for other

users, and the View-Only Recipients role, which allows view-only access to a number of sections within EAC, including recipients. In some respects, you might consider these roles insufficient for help desk agents to help users simply because they are so limited, and you might wonder why Exchange includes such a relatively insipid role. However, if more capabilities had been assigned to the Help Desk role, help desk personnel might have been given more permissions than necessary. This discussion proves that although the default RBAC roles cover a wide range of scenarios, it is impossible to satisfy everyone unless the ability exists to tailor roles and role groups to meet the specific needs of organizations.

Assume that the Help Desk role group is insufficient for the kind of help desk used by your company. You could edit the standard Help Desk role group and assign additional management roles to the group. You can do this easily with EAC by selecting the Help Desk role group and editing its properties. You can then click Add in the Roles section of the properties to reveal the Select a Role screen illustrated in Figure 4-5. All known management roles are listed and can be added to the role group. In this instance, the Help Desk personnel should be able to update details for mail recipients (mailboxes, groups, contacts), so you have selected the Mail Recipients management role. After you save the updated role information, the members of the Help Desk role group can run commands such as Set-Mailbox to update properties for a mailbox or Enable-Mailbox to enable an archive through either EAC or EMS. If you run the Get-ManagementRoleEntry command to see the cmdlets that are available to role group members who hold the Mail Recipients role, you'll note that Mail Recipients is actually quite a powerful role in terms of the number of cmdlets. However, the Mail Recipients role is also restricted in that the vast majority of permissions are for commands that update objects rather than create objects. For instance, you can't create a new mailbox or contact because the Mail Recipient Creation management role group must be assigned to gain that capability.

For performance reasons, Exchange caches RBAC assignments, so the update for a role group might not be effective for up to 15 minutes after it is made.

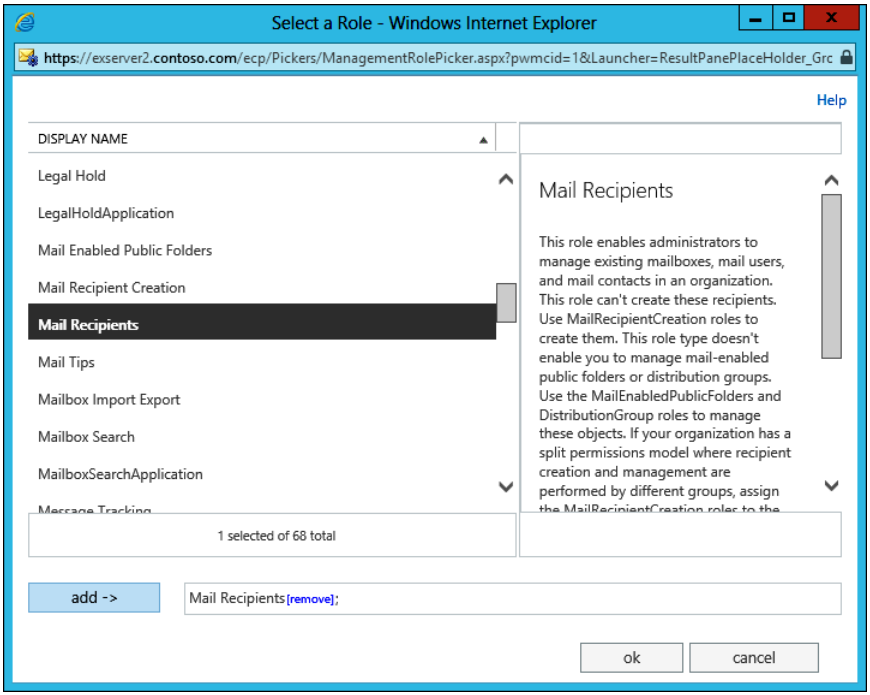


Figure 4-5 Selecting additional roles to add to a role group

Many companies organize help desks into different levels of support. Your adjusted Help Desk role group is probably adequate for the needs of entry-level help desk personnel. Assume now that you want to create a new role group that you assign to level 2 personnel who handle escalations and tasks that require a greater degree of knowledge. One way of achieving your goal is to use another standard role group. For example, the Recipient Management role group contains the following management roles:

- **Distribution Groups** Enables holders to create and manage groups, including dynamic distribution groups
- **Mail Recipient Creation** Enables holders to create new mail-enabled recipients such as mailboxes
- **Mail Recipients** Enables holders to set properties of mail-enabled recipients, including the ability to enable or disable an archive
- **Message Tracking** Enables holders to track the route messages take through an organization, including those to external recipients up to the point that they are processed by a connector and leave the organization

- **Migration** Enables holders to create and manage migration batches
- **Move Mailboxes** Enables holders to move mailboxes by using the New-MoveRequest and associated commands
- **Recipient Policies** Enables holders to create and amend throttling, Outlook Web App, and ActiveSync policies
- **Team Mailboxes** Enables holders to manage team mailboxes, including creating new team mailboxes and using commands such as New-SiteMailbox and Get-SiteMailboxDiagnostics

To demonstrate what's possible, create a new role group instead of just editing the Recipient Management role group. The new role group will include most of the management roles included in Recipient Management and add the Mailbox Import Export role group so that the members of the new role group can import and export mailbox data from and to PSTs.

EAC makes it easy to create a new custom role group based on an existing group by providing a copy option. Select the role group that you want to use as the base and click Copy. EAC then copies the properties of the base group and displays them so that you can make whatever changes are necessary to create the new group. Figure 4-6 shows the new role group being edited. The new group name is set to Help Desk Level 2, and some descriptive text is provided to inform administrators and other interested parties about why the new role group is needed. To complete the new group, edit the assigned roles as explained earlier, add the names of the users and groups you want to include in the group membership, and then click Save. In addition to creating the new management role group, Exchange also creates a new universal security group with the same name in the Microsoft Exchange Security Groups OU. The new USG is necessary to enable Exchange to assign security principals to the group.

The equivalent EMS command is New-RoleGroup. Here's an example that creates a new role group and defines the management roles covered by the group and the users who form the group membership:

```
New-RoleGroup 'Help Admins' -Roles 'Message Tracking', 'Mail Recipients', 'Move
Mailboxes' -Members 'Eoin.Redmond@contoso.com', 'Paul.Chen@contoso.com' -ManagedBy
'Nancy.Anderson@contoso.com', 'Mark.Jones@contoso.com' -Description 'This group is
used by Help Admins'
```

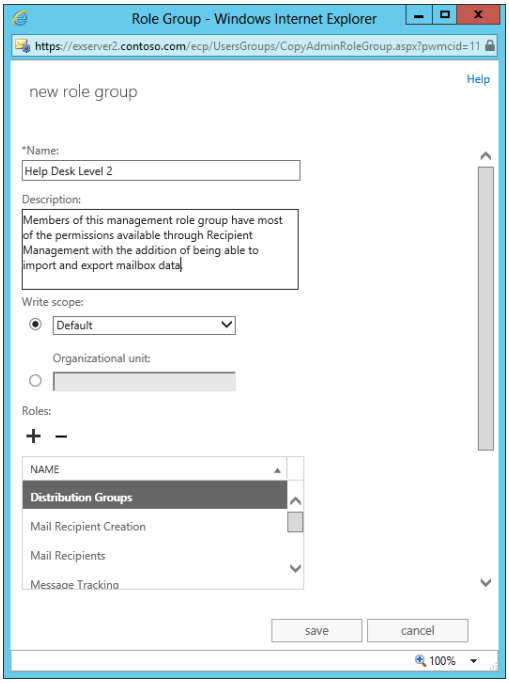


Figure 4-6 Creating a new role group from an existing group

In this example, specific users are named who will manage the new group. The users specified in the ManagedBy parameter are allowed to manage the group, but they are not members of the group and don't possess the role unless they are explicitly included in the group membership. Why this property is important is discussed in the "Role group management" section later in this chapter.

At least one role must be assigned to a new role group when it is created. If you do not assign a scope to the role group, it takes the default scope of the roles included in the group, which is usually organization-wide.

Maintaining role group membership

Users not included when the role group is created can be added afterward by editing the role group's properties by using EAC. Any mailbox or distribution group can be added to a role group's membership, but you cannot add a dynamic distribution group. Figure 4-7 shows how role group membership is edited through EAC.

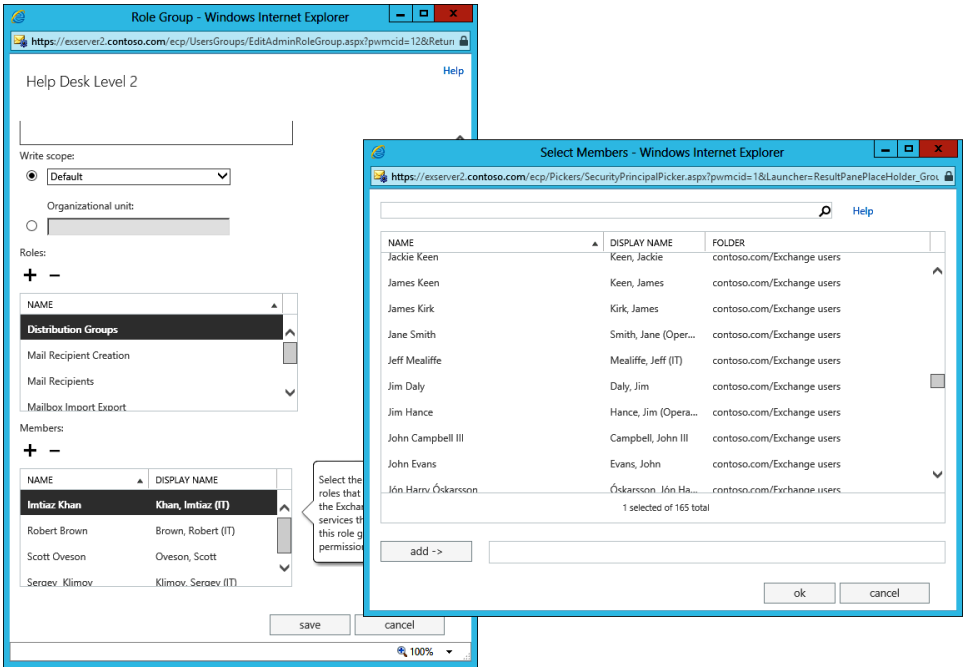


Figure 4-7 Editing role group membership

Another way to maintain role group membership is by running the Add-RoleGroupMember command to add new members and the Remove-RoleGroupMember command to remove group members. For example, these commands add my account to the selected role group and then remove my account from the membership:

```
Add-RoleGroupMember -Identity 'Help Desk Level 2' -Member 'Tony Redmond'
Remove-RoleGroupMember -Identity 'Help Desk Level 2' -Member 'Tony Redmond'
```

Adding individual members to a role group can be boring, but Exchange provides the Update-RoleGroupMember command to update the complete membership of a role group with a single command. For example, this command sets the membership of the role group to the three specified users:

```
Update-RoleGroupMember -Identity "Help Desk Level 2" -Members 'Rob.Young@Contoso
.com', 'John.Smith@Contoso.com', 'Nancy.Anderson@contoso.com'
```

You can always pipe a set of objects fetched using Get-Mailbox to add many members to a role group by using one command. For example, you can use Get-Mailbox to fetch details of every mailbox that is marked as belonging to an Exchange administrator by having ExAdmin in custom attribute 14 and piping the set of mailboxes to the Add-RoleGroupMember command:

```
Get-Mailbox -Filter {CustomAttribute14 -eq 'ExAdmin'} | Add-RoleGroupMember -Identity
'Help Desk Level 2'
```

Role group management

When EAC is used to create a role group, the Organization Management security group is added to the list of group managers along with the user who creates the group. Including Organization Management in the list of group managers ensures that anyone who works with the role group through EAC can update its properties or, when it is no longer required, delete the group. You can see this by running the Get-RoleGroup command. For instance:

```
Get-RoleGroup -Identity 'Help Desk Level 2' | Format-Table ManagedBy
ManagedBy
---------
{contoso.com/Microsoft Exchange Security Groups/Organization Management, contoso.com/
Exchange users/Tony Redmond}
```

The entries listed here are known as role group managers. The Organization Management universal security group is listed here rather than the Organization Management role group because normal Windows permissions provide the foundation for RBAC, and the permission to change or delete a group can only be assigned to an object that can hold Windows security principals such as security groups or user accounts.

If a user who is not listed in the ManagedBy property tries to change the membership of a role group or delete the group, EAC signals the error shown in Figure 4-8. The solution to the problem is to run the Set-RoleGroup command to update the ManagedBy property to include the Organization Management security group. For example, if John Smith is currently listed as the only manager of a role group, you can update the property as follows:

```
Set-RoleGroup -Identity 'My Role Group' -ManagedBy 'Organization Management', 'John
Smith'
```

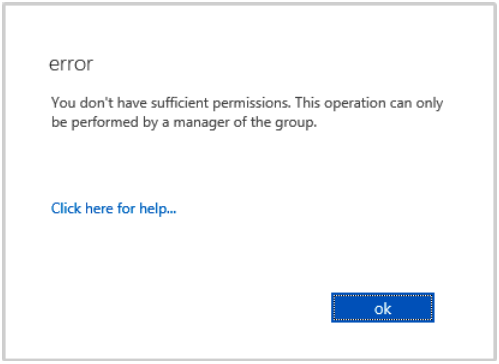


Figure 4-8 EAC signaling a permissions error when working with a role group

After running this command, any member of the Organization Management role group plus the user John Smith can manage the group.

If one of these members deletes the role group with EAC or by running the Remove-RoleGroup command, Exchange also removes the associated universal security group from Active Directory.

Role assignment

To summarize, you have defined scopes to say which objects a role can access and update, a role to say which cmdlets and parameters are made available to the holders of the role, and role groups to help manage who can do what. Role assignments provide what Microsoft refers to as the glue for RBAC in that they link roles and their scope to users or groups. To see the complete set of role assignments in the organization, type:

```
Get-ManagementRoleAssignment
```

Exchange 2013 includes some hundreds of role assignments ordered by assignee name. All the roles assigned to the Organization Management role group are found together, as are all the roles assigned to the other role groups such as Hygiene Management, Discovery Management, Move Mailboxes, and so on. Because it is the most powerful and functional role group, Organization Management has the most role assignments. Toward the bottom of the list, you'll find the roles assigned to the default role assignment, which is what Exchange assigns automatically to every user when the user's mailbox is created. These assignments include MyBaseOptions and MyContactInformation, which are the assignments necessary to enable a user to edit her own personal information through Outlook Web App options. Custom role assignments are found at the end of the list.

Role assignments are named using a convention of Role-Role Group, which results in names such as Message Tracking-Records Management and Transport Queues-Organization Management. Assume that the Help Desk Level 2 role group you created earlier includes seven roles, so you would then expect to see these represented by seven role assignments when you run Get-ManagementRoleAssignment, which is exactly what happens:

```
Get-ManagementRoleAssignment –RoleAssignee 'Help Desk Level 2'
```

```
Name                                                Role
-------                                             ----
Distribution Groups-Help Desk Level 2               Distribution Groups
Mail Recipient Creation-Help Desk Level 2           Mail Recipient Creation
Mail Recipients-Help Desk Level 2                   Mail Recipients
Message Tracking-Help Desk Level 2                  Message Tracking
Move Mailboxes-Help Desk Level 2                    Move Mailboxes
Recipient Policies-Help Desk Level 2                Recipient Policies
Mailbox Import Export-Help Desk Level 2             Mailbox Import Export
```

To examine a specific role assignment, pass the name of the assignment to the Get-ManagementRoleAssignment cmdlet:

```
Get-ManagementRoleAssignment –Identity 'Mail Recipient Creation-Help Desk Level 2' |
Format-List
```

RBAC does not function in the same way as regular security permissions, by which the operating system normally applies the most restrictive permission to secure access to an object. Instead, RBAC operates on the principle that it provides users with the combination of all the roles that have been assigned to them. This enables them to accomplish any of the tasks covered by those roles. Consider when you make an account a domain administrator. Immediately, the account gains a great deal of power within the domain, and there is a huge difference between the capabilities of that account and those of another standard user account. Now consider what happens when you add an account to a role group. It gains the ability to run any of the cmdlets covered by the role. Gradually, as you add the account to other role groups, it gains access to more cmdlets until you arrive at the Organization Management level, which is an accumulation of nearly every role available to Exchange. You begin to realize how roles build on roles to create a highly customizable framework to enable all kinds of users to do their work. Unity through role accumulation delivers a more granular security assignment mechanism than any of the previous Exchange ACL-based authorization schemes, even if there is no explicit deny mechanism that positively blocks someone from doing something.

INSIDE OUT **Controlling role assignments**

Each role assigned to a user provides cmdlets and parameters the user is authorized to run. The total set of roles a user holds creates the complete set of cmdlets and parameters the user can run. Because users gain the ability to run an increasing set of cmdlets through role assignment, you should assign only the precise roles to users that you want them to have. Otherwise, you run the risk of users gaining the ability to run a cmdlet or use a parameter because they are assigned a role that they don't really need. You can tailor the roles by adding or removing cmdlets. You can also tailor a role group by removing or adding a role assignment to reduce or increase the functionality available to the members of the role group.

Using role assignment policy to limit access

Role assignments can limit access down to the parameter level for a cmdlet. Exchange uses this capability to restrict users to writing to specific data items within a larger set. For example, the default role assignment policy enables users to update their contact details but not

their display name through the Options section of Outlook Web App. You can interrogate Exchange to discover the role assignments that underpin this capability by looking for the assignments that control users' ability to use the Set-User cmdlet to update their mobile phone number (one of the data items permitted by the default role assignment policy) as follows:

```
Get-ManagementRole -Cmdlet Set-User –CmdletParameters MobilePhone|
Get-ManagementRoleAssignment
-GetEffectiveUsers -Delegating $False | Where-Object {$_.Effectiveusername
-ne "All Group Members"} | Format-Table Role, RoleAssigneeName, EffectiveUserName
```

This code does the following:

- Looks for all roles that permit access to the MobilePhone property through Set-User.
- Pipes the roles to Get-ManagementRoleAssignment to return a list of users who have been delegated access through a role assignment. (If the –Delegating parameter is set to $True, you will see a list of users who can delegate access to others.)
- Filters the list to remove All Group Members because you want to see only individual users.
- Outputs the role, assignee name, and username.

Here is the output:

```
Role                  RoleAssigneeName                 EffectiveUserName
----                  ----------------                 -----------------
Mail Recipients       Organization Management          Administrator
Mail Recipients       Organization Management          Halstead, Dean
Mail Recipients       Help Desk Level 2 Support        Ruth, Andy
Mail Recipients       EMEA Help Desk                   Pelton, David
Mail Recipients       EMEA Help Desk                   Smith, John
Mail Recipients       APJ Help Desk                    Akers, Kim
User Options          Organization Management          Administrator
User Options          Organization Management          Redmond, Tony
MyContactInformation  Default Role Assignment Policy   All Policy Assignees
```

CAUTION!

RBAC currently allows restrictions to be placed on cmdlets that write data (cmdlets beginning with New- and Set-). It does not support restrictions on the Get- cmdlets that allow users to retrieve the properties of an object. This means that restricted users can read all the data for an object even though they might not be able to update some or even all the attributes.

INSIDE OUT **Controlling access to Remove-Mailbox**

Some organizations do not like the fact that Exchange administrators can remove a mailbox and the underlying Active Directory account by using the Delete Mailbox option in EAC or by running the Remove-Mailbox cmdlet in EMS. The same argument can be advanced for the Remove-MailContact cmdlet, but most of the focus is on the ability to remove a user account from Active Directory. Exchange doesn't allow you to remove role entries from its built-in roles, so you can't remove the Remove-Mailbox entry to solve the problem. Instead, you have to make sure that no one except organization administrators has access to the Mail Recipient Creation role because it is the only role that includes access to Remove-Mailbox. (You can verify this fact by running the Get-ManagementRole –Cmdlet 'Remove-Mailbox' command.) You can use the following code to check who has received delegated access to the Mail Recipient Creation role and then remove it, but make sure that you absolutely want to take this course of action before running the command:

```
Get-ManagementRoleAssignment -Role "Mail Recipient Creation" –Delegating $False
| Remove-ManagementRoleAssignment
```

Creating roles for specific tasks

It is entirely possible that your deployment might require a new role to perform very specific tasks. A new role is better than attempting to modify one of the built-in roles included in Exchange because you can never be sure that a change made to one of these roles might not affect or break functionality in some way or create difficulties during a future software upgrade. In addition, by creating a new role, you isolate your changes in one place and therefore make it easier to understand and to debug any problems that might occur.

You can create a new role for this purpose by using the following rules:

- Custom roles have to be created using one of the built-in roles as a parent; they cannot be created from another custom role.
- A child role cannot hold more rights than its parent.
- It is usually better to start with a limited set of cmdlets and gradually add cmdlets until the set necessary to perform the desired tasks is determined rather than attempting to slim down a large set of cmdlets.
- The set of cmdlets that is included in a role must include a Get cmdlet to match every Set or Remove cmdlet. For example, if you look at the MyProfileInformation role that allows users to update their own personal contact data through the Options section

of Outlook Web App, you'll see that it includes entries for the Get-Mailbox cmdlet to retrieve the current personal data and for the Set-Mailbox and Set-User cmdlets to allow the user to update the data. Look at the details of the individual role entries for Set-Mailbox and Set-User; you will see that the parameters are restricted to a limited set including the display name, first name, initials, and so on.

As an example, assume that you want your help desk staff to be able to update the personal information for recipients because you don't want users doing this through Outlook Web App. The first step is to discover which built-in role to use as the parent. You could choose MyProfileInformation or Mail Recipients because both roles allow access to user properties. In this case, base your custom role on Mail Recipients.

```
New-ManagementRole -Name 'Help Desk User Updates' -Parent 'Mail Recipients'
```

The built-in Mail Recipients role includes a very large number of cmdlets that you don't want to make available to the role holders, so you remove everything except the Get-Mailbox cmdlet because it's easier to put back the few cmdlets you need than to edit them in place.

```
Get-ManagementRoleEntry -Identity 'Help Desk User Updates\*' | Where {$_.Name -ne
'Get-Mailbox'} |
Remove-ManagementRoleEntry -Confirm:$False
```

You can now add the cmdlets to your role and specify the parameters that you'll allow with each cmdlet. You want to allow the help desk team to update a number of user settings, so you need to permit access to six cmdlets in all (five in addition to the one left after you remove the set inherited from the parent role). Some trial and error by testing whether you can perform the desired task through EAC and EMS might be necessary before you determine the exact cmdlet set you require:

```
Add-ManagementRoleEntry 'Help Desk User Updates\Set-Mailbox' -Parameters Identity,
DisplayName, SimpleDisplayName
Add-ManagementRoleEntry 'Help Desk User Updates\Get-User'
Add-ManagementRoleEntry 'Help Desk User Updates\Set-User' -Parameters Identity,
FirstName, LastName, Initials, Office, Phone, MobilePhone, Department, Manager
Add-ManagementRoleEntry 'Help Desk User Updates\Get-CASMailbox'
Add-ManagementRoleEntry 'Help Desk User Updates\Set-CASMailbox' -Parameters Identity,
IMAPEnabled, OWAEnabled, OWAMailboxPolicy
```

You can now check that the custom Help Desk User Updates role includes the correct cmdlets and parameters with:

```
Get-ManagementRoleEntry 'Help Desk User Updates\*' | Format-Table
```

Assuming everything looks good, you are now ready to assign the new role to users or to security groups. Exchange already adds the Help Desk security group to Active Directory to

use with its Help Desk management role. You can add your new management role to this role group as follows:

```
New-ManagementRoleAssignment –Role "Help Desk User Updates" –SecurityGroup "Help
Desk"
```

The next time a member of the Help Desk security group logs on, he will pick up the new role assignment.

You can find a description of a good example of creating a new role to enable users to perform a specific task (to wipe ActiveSync devices) at *http://blogs.technet.com/b/exchange/archive/2012/09/12/rbac-walkthrough-of-creating-a-role-that-can-wipe-activesync-devices.aspx*.

Specific scopes for role groups

As discussed earlier, Exchange supports the concept of scoping for a role group. When you set a scope for a role group, members of the role group can manage only the objects covered by the defined scope. The most common scope is set by an OU in Active Directory, meaning that a role assignment made with this scope can process any object in the OU. You can create the scope in three ways:

- **RecipientOrganizationalUnitScope parameter** Specify an explicit OU in Active Directory.
- **CustomRecipientWriteScope parameter** Create a management scope with the New-ManagementScope cmdlet that points to an OU in Active Directory. This approach is preferable if you will reuse the scope several times. It also enables you to give the scope a human-friendly name that might be more recognizable than the name of the OU.
- **Database name** Exchange enables you to create scopes based on database names.

For example, a large organization might have a distributed help desk with different teams located in regions around the world to support local user communities. You might not want the American help desk to be able to manage European users and vice versa. In this case, you could create two role groups for the American and European help desks and set the appropriate scope so that both role groups have access only to the users they support. This approach depends on a certain organization of user accounts within Active Directory because you cannot apply a scope to a notional set of users. A clear division such as an organizational unit must exist to support a scope.

TROUBLESHOOTING

You can't add a scope to an existing role group by using Set-RoleGroup

The Set-RoleGroup cmdlet doesn't allow a scope to be retrospectively added to a role group. If you need to set a scope on a role group after it is created, use the Get-ManagementRoleAssignment cmdlet to fetch details of all the role assignments for the role group and then pipe this information to the Set-ManagementRoleAssignment cmdlet to set the new scope for each role assignment. For example:

```
Get-ManagementRoleAssignment -RoleAssignee 'Help Desk Level 2' |
Set-ManagementRoleAssignment

-RecipientOrganizationalUnitScope 'contoso.com/Exchange Users'
```

In this example, you create a new role group with a scope set to allow the members of the role group to manage European mailboxes.

```
New-RoleGroup 'European Help Desk' -Roles 'Message Tracking', 'Mail Recipients',
'Move Mailboxes' -Members 'Luka.Abrus@contoso.com', 'Mark.Harrington@contoso.com'
-ManagedBy 'Darren.Parker@contoso.com', 'Ellen.Adams@contoso.com' -Description 'This
group is used to manage European mailboxes' -RecipientOrganizationalUnitScope
'contoso.com/EMEA Mailboxes'
```

As an alternative, you could create the management scope beforehand to point to the name of the OU and then use it with the New-RoleGroup cmdlet.

```
New-ManagementScope -Name 'European Mailboxes' -RecipientRoot 'contoso.com/Exchange
Mailboxes/EMEA' -RecipientRestrictionFilter {RecipientType -eq "UserMailbox}

New-RoleGroup 'European Help Desk' -Roles 'Message Tracking', 'Mail Recipients',
'Move Mailboxes' -Members 'Luka.Abrus@contoso.com', 'Mark.Harrington@contoso.com'
-ManagedBy 'Darren.Parker@contoso.com', 'Ellen.Adams@contoso.com' -Description 'This
group is used to manage European mailboxes' -CustomRecipientWriteScope 'European
Mailboxes'
```

Database scoping

To create a database scope, define a new management scope based on a database list or a database filter. A database list contains the names of specific databases separated by commas and is an appropriate scope when you need to assign management responsibility to a fixed set of databases that you don't think will change often. For example, this command creates a database scope that is limited to two named databases:

```
New-ManagementScope -Name 'CEO Databases' -DatabaseList 'CEO-Database1,
CEO-StaffDatabase'
```

A database filter establishes a condition that Exchange can use to identify a set of databases. This is the most appropriate choice when you want a scope that is flexible enough to accommodate a changing set of databases, assuming that you can create a filter that

identifies the databases. This example creates a filter that selects databases with a name that matches a prefix of DUB-:

```
New-ManagementScope -Name 'Dublin Databases' -DatabaseRestrictionFilter {Name
-Like 'DUB-*'}
```

INSIDE OUT **Using database names to create filters**

Names are likely to be the basis for most database filters, but you can create filters based on other properties, including the database description. Filters that are not based on the database name (for example, those that use properties such as the database description or distinguished name) require a certain discipline in maintaining those properties; else, the scope is unlikely to locate the desired databases. See the "Understanding Management Role Scope Filters" section in the Exchange Help file for a full list of the supported properties that can be included in database filters.

When you create a database scope, you permit access to the cmdlets that manipulate databases, such as Set-MailboxDatabaseCopy. However, you have to be careful not to overlap server scopes because some operations are permitted by either a database or a server scope, and some depend on a specific scope. For example, a database scope controls the ability to create a new mailbox with the New-Mailbox cmdlet or to move a mailbox with New-MoveRequest if the target database falls under its scope. This is logical because a server scope cannot apply in this case—databases are not tied to servers.

Special roles

The list of roles included in the Organization Management role group includes the following five special roles that have to be delegated before they can be used:

- **Application Impersonation** This is a special-purpose role intended primarily for use by Service Accounts that need to take on the persona of a user to accomplish a task. Because the impersonate role allows access to the data held inside user mailboxes, its use has to be carefully controlled.

- **Mailbox Import Export** This role allows a user to import data into or export data from a mailbox. This is another role whose allocation needs to be controlled on an as-needed basis.

- **Mailbox Search** This role allows a user to search mailbox contents. The role is assigned to the Discovery Management role group, but the role group has no default members and needs to be populated before searches can be performed.

- **Support Diagnostics** This role allows access to diagnostics cmdlets such as Test-ReplicationHealth that are intended for use by Microsoft or other support personnel to retrieve diagnostic information from an Exchange server or organization. The role is not assigned to any user by default.
- **Unscoped Role Management** This role permits unscoped roles to be created and managed. Unscoped roles authorize access to custom scripts and cmdlets. The role is not assigned to any user by default, but it can be delegated to users by holders of the Organization Management role.

These are called delegated role assignments. The members of the Organization Management role group have the right to delegate the roles to users, but they do not have the right to use the role themselves unless they delegate the role to themselves.

The inclusion of the Mailbox Import Export role in this list might be surprising, but it is entirely justified if you consider that you probably want to control the ability to import or export mailbox data on an as-needed basis. No one wants to run the risk that a user might be inadvertently given the ability to export mailbox data belonging to another user. When this access is required, you can assign it to a user who needs the role as follows:

```
New-ManagementRoleAssignment -Role 'Mailbox Import Export' -User 'Darren.Parker@con-
toso.com'
```

It might be more convenient to assign the role to a distribution group because it is often easier to maintain membership of a group than to perform individual role assignments. The group has to be a USG rather than a universal distribution group or a dynamic distribution group.

```
New-ManagementRoleAssignment -Role 'Mailbox Import Export' -SecurityGroup 'Mailbox
Import-Export Team'
```

After the role is assigned, assignees can use the Export and Import mailbox options in EAC and the underlying New-MailboxImportRequest and New-MailboxExportRequest cmdlets in EMS. Users must restart EAC/EMS after they have been assigned the role to force a refresh of the RBAC data and allow the new assignment to become effective.

See Chapter 10, “Moving mailboxes,” for more information about the cmdlets used for mailbox import and export operations.

Unscoped roles

Unscoped management roles are interesting because they enable you to create tailored roles for administrative purposes. This area might receive little attention as deployment teams handle the initial implementation of RBAC within a company. As administrators

become more comfortable with RBAC over time, the notion of being able to create and assign custom-built roles will become more interesting.

The basic idea behind a custom unscoped role is as a mechanism that enables administrators to grant access to non-Exchange cmdlets or Windows PowerShell scripts to management role groups, individual users, or USGs. For example, assume that you have written some PowerShell scripts that can extract data for reporting purposes from Exchange, and you want to assign access to those scripts to people who work on the help desk so that they can generate the reports. Granting the ability to perform specific work without the need to grant access to a particular role or cmdlet is another reason you might create a custom role. For example, you might want to allow the help desk to create mailboxes but only in a specific OU, following a structured naming guideline, and populating mailbox properties in a certain manner. You could do this by writing a script that encodes all the business logic necessary to create the mailboxes and granting access to the script. The help desk users who receive the custom role can run the script, but they can't run the New-Mailbox cmdlet in any other manner.

To begin, you have to delegate the ability to create unscoped roles to an account. This ability is not granted by default, even to accounts that hold the Organization Management role. However, accounts that hold the Organization Management role can delegate the Unscoped Role Management role to themselves or other accounts with the New-ManagementRoleAssignment cmdlet. This command assigns the role to a USG called Exchange Admins, which has to exist before you assign a role to it:

```
New-ManagementRoleAssignment –Role 'Unscoped Role Management' –SecurityGroup
'Exchange Admins'
```

If the account you use is a member of the Exchange Admins group, you can now create an unscoped top-level management role. Remember that you have to reinitialize a new EMS session after your account is assigned the Unscoped Role Management role to enable RBAC to make the UnscopedTopLevel parameter available for the New-ManagementRole cmdlet. If you don't do this, EMS will report an error when you run the New-ManagementRole cmdlet and pass it the UnscopedTopLevel parameter.

```
New-ManagementRole –Name 'Exchange Admin Scripts' –UnscopedTopLevel
```

```
Name                                    RoleType
----                                    --------
Exchange Admin Scripts                  UnScoped
```

The management role is empty and now needs to be populated with the scripts you want to make available to the users to whom you eventually assign the role. The scripts need to

be copied to the default remote script directory on every server (\Program Files\Microsoft \Exchange Server\V15\RemoteScripts\). Thereafter, you can add a role entry for each script. For example:

```
Add-ManagementRoleEntry 'Exchange Admin Scripts\DBReportMail.PS1' -Type Script
-UnscopedTopLevel
```

This command associates the DBReportMail.PS1 script with the custom role and allows the holders of the role to run the script. You can assign the custom role to the users who need it, as usual:

```
New-ManagementRoleAssignment -Role 'Exchange Admin Scripts' -User 'Help Desk'
```

After the assignment is made, the users can run the script the next time they log on to EMS.

Which role groups do I belong to?

A simple question administrators often ask is to which role groups have they—or someone else—been assigned. Assignment information is held in the membership of the different role groups, so that's what you have to investigate to determine which roles a user possesses.

In this example, you run the Get-RoleGroup cmdlet and pipe its output to the Where-Object cmdlet to look for any entry in the membership of a role group that has a partial match with a user called Redmond:

```
Get-RoleGroup | Where-Object {$_.Members -Like '*Redmond*'} | Format-List Name,
Members
```

```
Name    : Organization Management
Members : {contoso.com/Exchange Users/Redmond, Eoin P, contoso.com/Exchange
Users/Redmond, Tony, contoso.com/Users/Administrator}

Name    : Discovery Management
Members : {contoso.com/Exchange Users/Redmond, Tony, contoso.com/Users/
Administrator}
```

The output shows that a partial match for Redmond is discovered in the membership of the Organization Management and Discovery Management role groups. You can then scan the membership information that's returned to find the user in whom you are interested.

Of course, if you just want to check the membership of a role group to see the list of users, the Get-ManagementRoleAssignment cmdlet is the correct tool. In this example, you retrieve the list of users who have an assignment for the Mailbox Import Export role group:

```
Get-ManagementRoleAssignment -Role 'Mailbox Import Export' | Format-Table
RoleAssigneeName, RoleAssignmentDelegationType, RoleAssigneeType
```

```
RoleAssigneeName            RoleAssignmentDelegationType       RoleAssigneeType
----------------            ----------------------------       ----------------
Organization Management     DelegatingOrgWide                  RoleGroup
Administrator               Regular                            User
```

You can see two types of assignments reported here:

- Members of the Organization Management role group are allowed to delegate access to the Mailbox Import Export role group, but they are not granted the permissions assigned to a regular member; members of the Organization Management role group can assign the role group to others, but they are not allowed access to the functions made available to role group members.
- The Administrator user account is a regular member. This is the result of a specific assignment. In fact, because the Administrator account is usually a member of the Organization Management role group, administrators can assign themselves membership to Mailbox Import Export.

The Get-ManagementRoleAssignment cmdlet is useful in terms of discovering who can do what to an object. For example, you can use it to determine who can write to different objects:

- A user mailbox:

  ```
  Get-ManagementRoleAssignment -WritableRecipient "Akers, Kim" -GetEffectiveUsers
  ```

- A server object:

  ```
  Get-ManagementRoleAssignment -WritableServer 'ExServer2' -GetEffectiveUsers
  ```

- A mailbox database:

  ```
  Get-ManagementRoleAssignment -WritableDatabase 'DB1'
  ```

If you use the GetEffectiveUsers parameter, all the users who can modify the object indirectly through membership of role groups and USGs are returned. If you omit this parameter, only the role groups, users, and USGs that are directly assigned the role are returned.

Assignment policies

So far, you have looked at direct assignment of roles to role groups by which users receive rights through their membership in the group that enable them to perform administrative operations such as viewing transport queues or conducting discovery searches. Every

RBAC system needs a default policy to provide a basic set of functions that users can run. Exchange 2013 includes the concept of a management role assignment policy to enable users to perform certain functions that have to be performed by administrators in previous versions of Exchange. Table 4-2 lists the roles covered by the default role assignment policy, which is assigned automatically to an end user when his mailbox is created unless the administrator overrides it by specifying another role assignment policy or by changing the default role assignment policy. The role assignment policy specified for a mailbox can be changed at any time by running the Set-Mailbox cmdlet. For instance:

```
Set-Mailbox –Identity JSmith –RoleAssignmentPolicy 'New User Role Assignment Policy'
```

Note

End-user roles are different from management roles in that they only affect data relating to the end users, such as their personal information, or the distribution groups that include the end users. By comparison, management roles have a much broader scope in that they can affect data relating to other users or other components of Exchange.

A mailbox can have only one management role assignment policy. Individual mailboxes or groups of mailboxes can be assigned different management role assignment policies. You can use the following command to see the roles included in the default role assignment policy:

```
Get-ManagementRoleAssignment -RoleAssignee 'Default Role Assignment Policy'
```

If you want to check the roles assigned to a specific user through a role assignment policy, you can substitute the user's name for the name of the assignment policy. For example:

```
Get-ManagementRoleAssignment -RoleAssignee 'Akers, Kim'
```

TABLE 4-2 User roles in default role assignment policy

Role	Use	Enabled by default
MyBaseOptions	Base option that allows end users to access Outlook Web App options	Y
MyContactInformation	Enables end users to update their phone and contact information	Y
MyProfileInformation	Enables end users to update their first name, last name, initials, and display name	N
MyVoiceMail	Enables end users to manage their voice mail options such as greetings	N

Role	Use	Enabled by default
MyTextMessaging	Enables end users to manage options for text messaging	N
MyDistributionGroupMembership	Enables end users to manage their membership in distribution groups (list groups, leave groups, join new groups)	Y
MyDistributionGroups	Enables end users to create new groups and to manage the membership of groups they own	N
MyTeamMailboxes	Enables end users to manage team mailboxes they own	Y
MyMarketPlaceApps	Enables end users to add apps to Outlook Web App	Y
MyRetentionPolicies	Enables end users to select personal retention tags to apply to items in their mailbox	N

You can remove any of these roles from the default role assignment policy and thus make them unavailable to users through Outlook Web App options. For example, to remove the text messaging options from Outlook Web App options:

```
Remove-ManagementRoleAssignment 'MyTextMessaging-Default Role Assignment Policy'
```

As you'll see in the discussion about EAC in Chapter 5, "Mailbox management," administrators can also change the default role assignment policy to make other options available to users. In addition, you have the flexibility to create a new role assignment policy and apply it to selected users to allow them access to a different set of tasks than is available to standard users. To set a new default role assignment policy:

```
Set-RoleAssignmentPolicy 'New End-User Default Role Assignment Policy' –IsDefault
```

Management role assignment policies are assigned with the New-Mailbox cmdlet or Enable-Mailbox cmdlet when you create a new user account or enable an existing account with a mailbox or with the Set-Mailbox cmdlet to change the policy for an existing mailbox. These assignments are explicit, whereas the assignment of the default policy is implicit. An explicit assignment always takes precedence over an implicit assignment. Here's how you would assign an explicit policy to a mailbox:

```
Set-Mailbox –Identity 'Jack Jones' –RoleAssignmentPolicy 'VIP Users'
```

Sometimes it is useful to be able to process a group of users. For example, assume that you want to run a Unified Messaging pilot in just one office and want to enable the users in that office to update their voice mail settings through Outlook Web App options. The voice mail options are not enabled in the default policy, so you must create a new policy, assign the

voice mail options to the policy, and then enable the policy for the mailboxes in the specific office. This set of commands does that. You create the policy, assign the necessary roles, including voice mail and the other roles users need to update their contact and personal information through Outlook Web App, and then assign the new role to all mailboxes that belong to the Chicago office:

```
New-RoleAssignmentPolicy -Name 'VoiceMail Pilot Users'
New-ManagementRoleAssignment -Role 'MyBaseOptions' -Policy 'VoiceMail Pilot Users'
New-ManagementRoleAssignment -Role 'MyVoiceMail' -Policy 'VoiceMail Pilot Users'
New-ManagementRoleAssignment -Role 'MyProfileInformation' -Policy 'VoiceMail Pilot
Users'
New-ManagementRoleAssignment -Role 'MyContactInformation' -Policy 'VoiceMail Pilot
Users'
Get-Mailbox -Filter {Office -eq 'Chicago'} | Set-Mailbox -RoleAssignmentPolicy
'VoiceMail Pilot Users'
```

Users will pick up the new management role assignment policy the next time they log on to Outlook Web App.

Implementing a split permissions model

Exchange uses two very important security groups to gain access to Active Directory information. These two groups provide the control over Active Directory objects that is implemented through cmdlets and made available to administrators and users through the role groups managed by RBAC.

- The Exchange Trusted Subsystem group enables Exchange services and cmdlets to manipulate objects that Exchange owns exclusively such as servers, connectors, and the properties that make an object mail-enabled.
- The Exchange Windows Permissions group enables Exchange services and cmdlets to manipulate objects that are often controlled by Active Directory administrators such as the creation of user and group objects.

The default Exchange installation ties the two together by making the Exchange Trusted Subsystem group a member of the Exchange Windows Permissions group. This arrangement allows Exchange services and cmdlets full access to anything held in Active Directory. For many companies, the unification of Exchange and Windows administrative permissions in a security model that permits administrators to control both Exchange and Active Directory works well, especially when no great differentiation exists between the administrators who manage Exchange and those who manage Active Directory and Windows. However, other companies have clearly delineated responsibilities between the two sets of administrators, and the default RBAC implementation in Exchange doesn't meet their needs as well.

The shared administrative model is the default for Exchange 2013, and you have to choose explicitly to implement a clear division between Exchange and Windows. If desired, you can decide to institute a split permissions model. However, you have to take this option when you run Setup to install the Exchange organization. You can do this through the GUI version of Setup or by running Setup.com to prepare Active Directory for Exchange. Launch Setup in a command box with elevated permissions with the following parameters:

```
C:> Setup /PrepareAD /ActiveDirectorySplitPermissions:True
```

It is possible to revert to the normal permissions model by running Setup again to reverse the changes it made to groups and role assignments. Use the following command:

```
C:> Setup /PrepareAD /ActiveDirectorySplitPermissions:False
```

In the background, the division is effected by removing the Exchange Trusted Subsystem group from the Exchange Windows Permissions group. Because all Exchange servers are members of the Exchange Trusted Subsystem group, this removes the ability of Exchange servers to manipulate many Active Directory objects unless code (such as cmdlets) is run by an account that has sufficient Windows permissions to update Active Directory. In addition, role assignments are modified to remove the ability of Exchange administrators to perform actions such as the creation of a new mail-enabled account.

Figuring out RBAC

RBAC probably won't make much difference in the way the administrator of a small Exchange deployment approaches her work. If you log on to the Administrator account and perform all tasks from there, the default assignment of the Organization Management role to the Administrator account makes the account powerful. It has full read and write access to any object in the Exchange organization in much the same way as the Administrator account can do anything with Active Directory. RBAC is much more interesting and useful within larger deployments, when finer granularity is often required for administrator roles.

INSIDE OUT **Heed good advice**

Although you can go to extremes in defining exactly what someone can do, Microsoft offers some good advice in that you should avoid special role assignments to individual users because this can create a complicated security environment that becomes hard to manage as people leave or join the organization.

Sorting out the connections among roles, role groups, role assignments, scopes, and users can be complex when you start to use the Exchange implementation of RBAC. It is wise to take some time to get to know how the different parts fit together before you begin to change the default assignments or add your own role groups and assignments to meet the needs of your organization. Sit down with a test server and see what you can discover about RBAC. Try some of the commands you have investigated in this section to become accustomed to the concepts and how they are implemented in Exchange. Eventually, it will all make sense.

On to management

Exchange boasts other management tools with which administrators will spend a lot of time, so become comfortable with the Exchange Administration Center and with the role-based authorization model that determines the access administrators have to different areas of the product. Next to be explored is therefore the Exchange Administration Center, new to Exchange 2013, and the many options it provides for user management.

CHAPTER 5

Mailbox management

Seeking perfection halts progress 170
Managing Recipients 180
The need for mailboxes 183
Naming mailboxes 185
Creating new mailboxes 187
Mailbox resources provisioning management agent . . 209
Setting mailbox quotas 213
Removing or disabling mailboxes 218
Reconnecting mailboxes 220
Discovery mailboxes 224
Health mailboxes 226
Setting mailbox permissions. 229
Shared mailboxes 240
Recalling messages. 241
Moderated recipients. 243
Mail-enabled contacts 250
Mail users. 252
Resource mailboxes 254
Enough about mailboxes. 266

There's not much point in running an email system unless people can use it to communicate. Learning how to use the tools available to manage these objects is a fundamental first step in this process. Microsoft has justifiably been criticized in the past for making Exchange Server complex to manage, largely through the use of badly documented and archaic registry settings to enable or manage features. In addition, customers have asked for tools that are flexible enough to accommodate the needs of a range of administrative personnel, from local administrators who take care of tasks for just one server to help desk personnel to organization administrators. For example, in Exchange Server 2007, if you want to give someone the ability to maintain user properties such as the office in which he works, you must give that person access to the full-blown management console. In addition to the tools, granting and maintaining permissions to enable people to work with Exchange is complicated, resulting in the potential for error in granting permissions that could expose data to unintended manipulation by untrained users. Apart from the limited set actions that can be recorded through administrative auditing, there is no good way to capture exactly what happens to an Exchange server by using the administrative tools.

Using the administrative model as implemented in previous versions of Exchange caused administrators to spend far too much time doing mundane things to keep an Exchange organization healthy when they could be more productive elsewhere. Microsoft improved matters in Exchange 2010 with the introduction of the browser-based Exchange Control Panel (ECP), an advance that enabled administrators to manage Exchange on any device that could run one of the supported browsers, including tablet and smart phone devices.

As good as it was to see the introduction of ECP, it meant that Exchange 2010 boasted three management interfaces—the Microsoft Management Console (MMC)–based management console, ECP, and the management shell. Having three tools was confusing, especially when some options (such as group naming policies) were only available in ECP, and others (such as database management) only appeared in Exchange Management Console (EMC). The influence of Exchange Online also mandated a need for change because more web-based rather than Windows-based management is a higher priority as we head into an era of cloud-based services.

The upshot is that Exchange 2013 has rationalized down to two management interfaces. The Exchange Administration Center (EAC) is very much like ECP in that it is browser-based and presents different options through portions of the user interface (UI) that are revealed or suppressed depending on a user's membership in role-based access control (RBAC) groups. EAC is very different from the previous Windows-based administration consoles, and it takes a little while for an administrator to become truly comfortable working with the tool. Exchange Management Shell (EMS) underpins everything and remains the most powerful management tool because of the wider range of parameters you can adjust and how you can build scripts to manage Exchange the way you want rather than the way an engineer has designed.

Seeking perfection halts progress

It's easy to criticize the appearance and functionality of EAC by saying that EMC was so much better and more powerful. This statement is doubtful in any case, but the real point is that MMC-based consoles and the operating model they embrace, which is to log on to servers to perform management tasks, is rapidly becoming an outdated mode for Windows server administration, especially in large deployments. Microsoft has acknowledged that this is the case by incorporating so much potential for automation through Microsoft Windows PowerShell in Windows Server 2012.

Exchange adopted Windows PowerShell as the basis for management much earlier than any other Microsoft server application and then went on to introduce browser-based management. EAC combines the two procedures as shown in Figure 5-1. This console is different in terms of layout and capability from previous consoles, but it is capable of running on many types of devices, from Apple iPads to Surface RT tablets to Android smart phones. The parts of EAC that aren't quite as functional as previous consoles will improve through future software releases, just as any other piece of software improves over time. In the interim, EMS is always available to handle the most difficult and complex tasks.

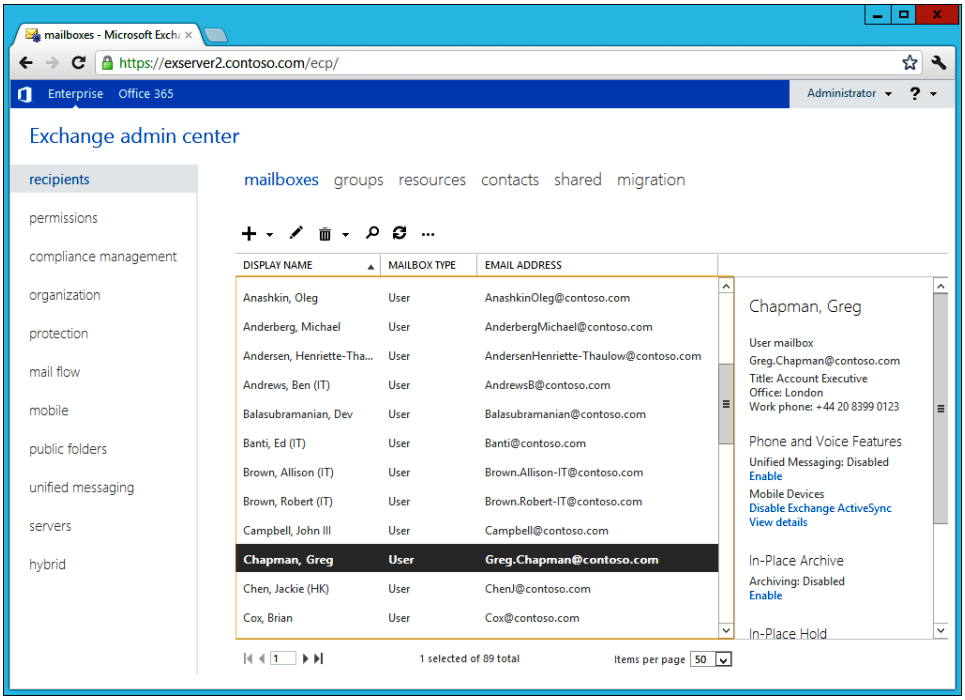


Figure 5-1 The Exchange Administration Center

Figure 5-1 shows how EAC organizes its view of management of the Exchange organization into the following major parts:

- **Recipients** Management of mailboxes, groups, contacts, room mailboxes, shared mailboxes, and mailbox migration (move).
- **Permissions** Management of RBAC-controlled administrative and user roles and Outlook Web App policies.
- **Compliance Management** Management of in-place hold (for mailboxes), discovery management, retention tags and policies, mailbox and administrator auditing, data loss prevention, and journaling.
- **Organization** Management of federated trusts for sharing with other organizations, third-party apps for use with Outlook Web App, and address lists.
- **Protection** Management of the Exchange 2013 anti-malware filter.

- **Mail Flow** Management of transport rules, delivery reports, accepted domains, email address policies, receive connectors, and send connectors.
- **Mobile** Management of ActiveSync quarantined devices, device access rules, and ActiveSync mailbox access policies.
- **Public folders** Management of modern public folders and public folder mailboxes. Management of traditional public folders is done through EMS or the Public Folder Management Console. You can't install or create traditional public folders if you haven't used them prior to the deployment of Exchange 2013.
- **Unified Messaging** Management of dial plans and IP gateways.
- **Servers** Management of Mailbox and Client Access Server (CAS), Database Availability Groups (DAGs), databases, virtual directories, and Secure Socket Layer (SSL) certificates.
- **Hybrid** Management of the connection between Exchange on-premises and Exchange Online (Office 365).

Like EMC, EAC is capable of operating in a multi-forest environment, assuming that the necessary trusts are in place to allow authenticated cross-forest access. For more information on this topic, see *http://blogs.technet.com/b/exchange/archive/2012/08/30/using-eac-to-manage-multi-forest-exchange-deployments.aspx*.

INSIDE OUT Some missing pieces of administrator functionality

Some functionality has been lost in the transition from EMC to EAC. A possibility exists that some will reappear in a future release or update that Microsoft issues for Exchange 2013. You might consider that losing any functionality means that EAC represents a step backward when compared to previous consoles and, on a feature-by-feature basis, this feeling is probably true. However, Microsoft will gradually whittle down the list of missing features and increase EAC functionality as it releases updates for Exchange 2013. For example, Exchange 2013 RTM CU1 reintroduced the ability for groups to manage groups. However, EAC offers better coverage in parts than other consoles. For instance, compliance management is generally better served in EAC than it is in the Exchange 2010 EMC, EAC includes the ability to import mailbox data to and export mailbox data from PSTs, and EAC includes alerts (see Figure 5-2) to notify administrators about important events. None of these options are available in EMC.

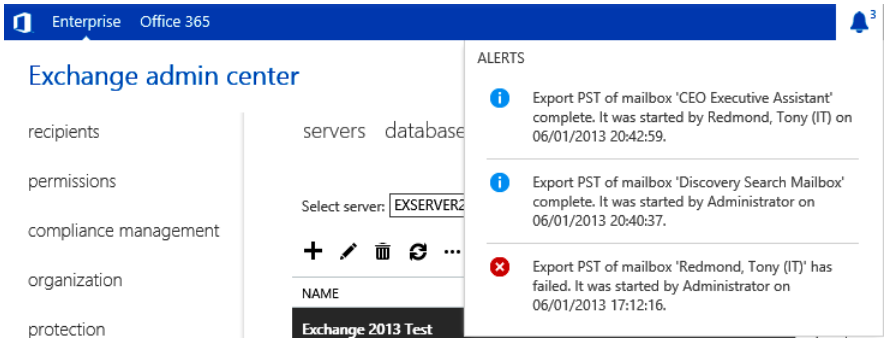


Figure 5-2 EAC Alerts

Many also prefer the way EAC presents information required to accomplish tasks because they think the EAC approach is easier to understand than the multistage wizard structure often employed by EMC. Building transport rules is one example of when EAC is arguably easier to understand than EMC.

Like EMC, there are always parts of the administrative task list that can't be handled through an option presented through the graphical user interface (GUI). These tasks are usually in the "rare and uncommon" category and require a reasonable degree of product knowledge before they can be addressed. For example, there's no option to view mailboxes the Store has quarantined because of a suspected corrupt item. Likewise, no option exists to run the Store fix-up-in-place cmdlets to resolve minor logical corruptions in a database's tables. You might never find options for these tasks presented through a GUI, if only because it's a way of underlining that you really need to know what you're doing when you take risks.

Starting EAC

Like EMS, EAC is based on remote Windows PowerShell and RBAC. When EAC starts, it contacts the server specified in the URL you typed into the browser to initialize a remote PowerShell session, for example, *https://exserver1.contoso.com/ecp*. You can see that EAC connects to the ECP virtual directory to give the same URL as used with the Exchange 2010 Control Panel. This is to provide a certain degree of backward compatibility. If you type **https://exserver1.contoso.com/eac**, you get an HTTP 404 error because Internet Information Services (IIS) has no EAC virtual directory to which it can connect.

The URL for EAC can have the relevant section appended to bring you to a particular location. For example, *https://exserver1.contoso.com/ecp/?exsvurl=1&p=Mailboxes* starts EAC and positions the console in the Mailboxes section, whereas *https://exserver1.contoso.com/ecp/?exsvurl=1&p=Mobile* starts EAC in the Mobile (ActiveSync) section.

INSIDE OUT Making sure that you see the right EAC

When you make a connection to EAC, Exchange queries Active Directory to discover the version running on the server that hosts your mailbox to ensure that the correct UI is displayed. This means that if your mailbox is still on Exchange 2010, you will see ECP rather than EAC. To force Exchange to display EAC, you need to add *?ExchClientVer=15* to the URL. For example: *https://server/ecp/?ExchClientVer=15*.

After EAC has established a remote Windows PowerShell session, it begins to retrieve the data necessary to fill in whatever part of the UI is selected. During initialization, EAC executes the cmdlets RBAC permits (for the account used to run EAC) to discover information about the organization, servers, and so on to build its cache with essential data about the Exchange organization. Later, EAC executes other cmdlets to retrieve information about specific objects as the user navigates from node to node. For example, if the user moves to Recipients and clicks Mailboxes, EAC runs Get-Mailbox to fetch the information to display. Because of the way it fetches and caches data, EAC performs better than EMC when dealing with large amounts of information, such as fetching details of 2,000 mailboxes. In this respect, EAC works in a similar manner to Outlook Web App when it navigates through mailbox folders that contain thousands of items.

RBAC ensures that an administrator sees only the options with which she can work. For instance, if your account doesn't hold the Discovery Management role, you might be able to create a new in-place search, but you cannot edit the query Exchange uses to locate items in user mailboxes. In other words, a regular administrator can set up the framework for a search, but the search can be activated only to collect items from user mailboxes by a user who is a member of the Discovery Management RBAC role group.

Likewise, even if your account holds the Organization Management role and is therefore able to see just about every option imaginable, you won't see the options to import or export mailbox data unless an explicit role assignment has been made to assign the Mailbox Import Export role to your account. Building a customized UI based on a user's role is a good thing because it stops frustration caused when someone attempts to take an action that he doesn't have the necessary permission to perform, even if it creates some new questions when users ask why their version of EAC is different from someone else's (or from what they read about in books or online materials). However, when loaded during the EAC initialization process, the RBAC data is inflexible in that if a change is made to a user's role, EAC will not reflect the change until the next time it loads and rebuilds its cache by reading RBAC information from Active Directory.

Unlike Exchange 2010, in which you have to install a language pack to use the management tools in different languages, Exchange 2013 installs the necessary language-dependent pieces to enable administrators to manage Exchange in their preferred language. When EAC starts, it uses the locale determined by the language setting in the user's mailbox. For example, if your language setting is en-us, you see the U.S. English version of EAC, whereas if it is fr-fr, you see the French (Figure 5-3). The language setting is populated the first time a user runs EAC or Outlook Web App or by running the Set-Mailbox cmdlet to write a value into the languages property. How to control the language setting for users is discussed in the "Languages" section later in this chapter.

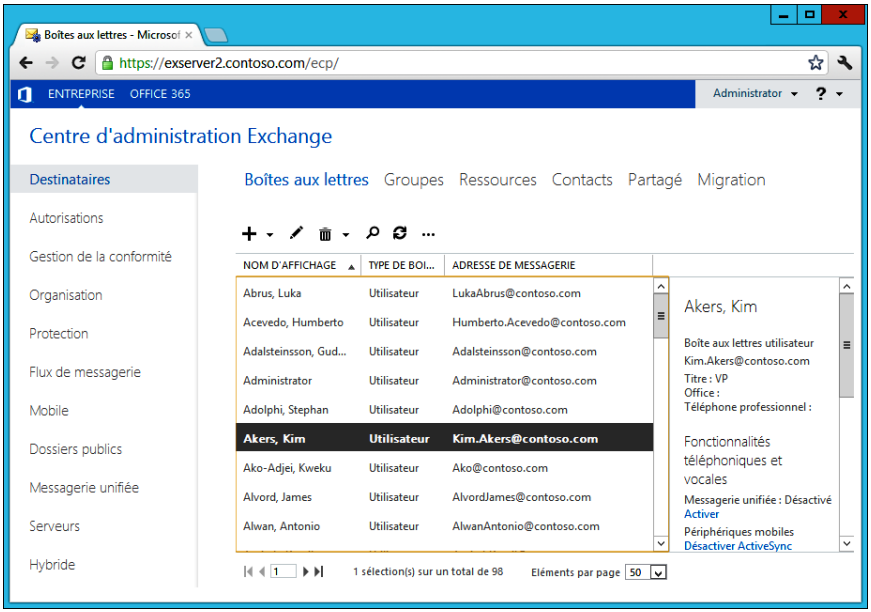


Figure 5-3 Running EAC in French

You see only the tasks allowed for your role

Like EMS, EAC uses RBAC to ensure that users see only the tasks their role allows them to perform. In other words, EAC can modify the options it displays to reflect the roles a user holds. Administrators who hold the Organization Management role see all options and can work with user data; administrators who hold a more restricted role see a limited set of options. EAC also respects any RBAC scopes that are in place to restrict users to working with a specific set of servers or databases.

How EAC accesses Exchange data

EAC has an absolute dependency on Active Directory. Throughout a session, EAC reads and writes data about objects it fetches from the Microsoft Exchange organization container in the Configuration Naming context. Other data (such as the current quota a mailbox uses) comes from mailbox databases, but the vast majority of information EAC displays is sourced from Active Directory. However, you should never manage Exchange objects through the Active Directory tools because these tools possess an incomplete knowledge of objects such as Dynamic Distribution Groups.

The information contained in Active Directory is static and is not intended to reflect information that changes in real time because this would generate constant replication requests to keep pace with status updates for Exchange objects. Even if Active Directory could keep up with the replication, it's likely that the activity would swamp networks and prevent other useful work from being done. The dependency on Active Directory is why EMS sometimes exposes transient data that you never see in EAC. For example, you can use the Get-MoveRequestStatistics cmdlet to view the percentage of a mailbox move that is complete and the current rate of data transfer between source and target server, but you never see this level of detail in EAC. Instead, EAC displays the status of the move requests in a migration batch from start to in progress to complete, but only if you refresh the display.

INSIDE OUT **Getting the latest information**

Because EAC essentially gives only a static snapshot of the set of objects at which you are looking, it is wise to use the refresh option before you start to do anything with EAC to make sure that you are dealing with the latest information rather than stale data. For instance, refreshing the set of mailboxes picks up new mailboxes that have been added and updates mailboxes that have had status changes, such as those that are being moved to a different server or those that have just been given an archive.

Note

When you view mailbox properties, you might see that a mailbox has been last logged on to by an account that doesn't own the mailbox. This occurs when another user has logged on to the mailbox by using delegated permissions that he has been granted.

Unlike EMC in Exchange 2010, which enables you to select a specific domain controller from which the console fetches Active Directory data, EAC automatically selects a domain controller from the set available in its local site and doesn't allow you to change this server during a session. Another difference between EMC and EAC is the way the two consoles fetch large numbers of objects. For performance reasons, EMC limits itself to fetching 1,000 objects unless explicitly forced to fetch more. The default value worked well for small installations but was not so good when large numbers of objects existed, such as the number of mailboxes supported by large organizations. EMC got over this difficulty by allowing administrators to modify the maximum number of objects to be fetched. Clearly, it took longer to fetch 10,000 objects than 1,000, but it was a reasonable solution. EMC also supports extensive filtering capabilities to enable an administrator to view a subset of objects, such as all the mailboxes located in a certain database or those that belong to a specific department.

EAC takes a completely different approach to fetching objects and uses a similar mechanism to the way Outlook Web App navigates through mail folders. A folder in an Exchange mailbox can hold tens of thousands of items, so it's unreasonable to expect any client to fetch all items when it opens a folder. Outlook Web App navigates on a page-by-page basis so that it fetches sufficient data to display enough objects to fill the current screen, which enables the user to begin working with data, and then fetches enough data to populate the next few screens to support the user navigating further within the folder. Outlook Web App can also move quickly between the top and bottom of a folder. In the case of EAC, the UI is designed to move through data by using 50 objects per page, but you can adjust it to display 100, 200, or even 500 objects per page to accommodate larger screen sizes. Behind the scenes, EAC caches more data so that you can move from page to page quickly. This approach limits the amount of data that has to pass between server and client while also enabling the UI to perform well, even when confronted with large quantities of data.

Changing EAC columns

Like EMC, the recipients section of EAC can be customized to add or remove columns to make the data shown more useful to an administrator. Click the ellipses and choose Add/Remove Columns to change the columns you see when you access different types of objects. Figure 5-4 shows the process in action. EAC does not offer the same facility for the other sections within the console.

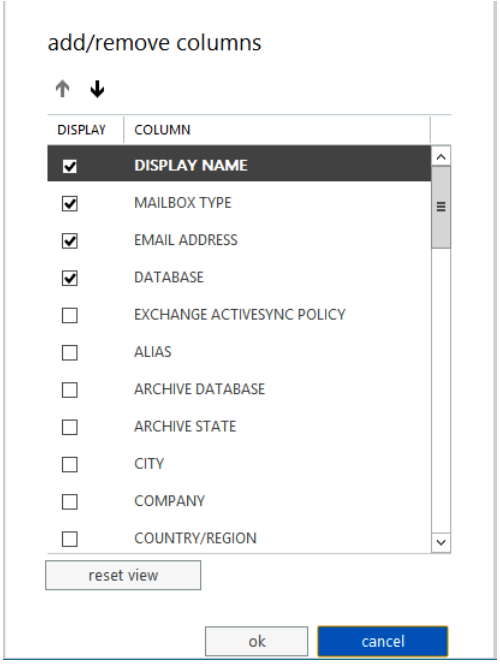


Figure 5-4 Selecting columns for EAC to display

Naming conventions

The topic of naming conventions should be covered during your planning for deployment. Server naming conventions was discussed in Chapter 2, "Installing Exchange 2013." It's clearly important for servers to be assigned names that make sense and convey some information about a server's purpose; this makes it much easier for administrators to manage the organization. Other important objects that deserve some attention in naming include the following:

- **User mailboxes** My strong preference for many years has been to use the Last Name, First Name convention because the convention mimics the way old-fashioned telephone directories work, so it's easier to navigate the large groups of users in the Global Address List (GAL) who share common surnames (such as Smith or Ng). The convention also works well for multinational companies that have to accommodate non-European surnames.

- **Room and equipment mailboxes** Most companies have already named rooms in buildings so it makes sense to follow the established convention. When an organization includes rooms in multiple buildings, you might want to prefix the room name

with a building identifier. For example, the Frankfurt conference room in Building 43-1 might be called B43-1-Frankfurt. Building names tend to be well understood by users, so you can afford to be a little cryptic in the names for these mailboxes.

- **Groups** Ideally, general-purpose distribution groups should convey the use of the group (for example, Exchange 2013 Interest List), and those intended for business-linked communication should indicate the business group and purpose (for example, Finance Department Planning Group). Common sense and consultation with the group owners to understand the purpose of the group usually leads to a sensible and easily understood name. Some companies, including Microsoft, add an indication to show users when a group is based on a query so that they won't bother looking up the directory to check group membership. Microsoft appends (QBDG) to the ends of these group names, so you end up with a group name such as All Users (QBDG). Exchange enables you to implement a group naming policy, which is described in Chapter 6, "More about the Exchange Administration Center."

- **Mail-enabled contacts** These objects should use the same naming convention as user mailboxes.

- **Public folders** Use the same approach to naming as for distribution groups. Above all, avoid any temptation to be cryptic because it can be hard enough to navigate the public folder hierarchy without creating another obstacle to user comprehension.

- **DAG** These objects are visible to administrators only, but it's still important to use a convention that informs administrators about the DAG's purpose.

- **Databases** Exchange 2010 and Exchange 2013 require databases to have names that are unique within the entire organization. The simplest convention is to assign names that indicate what mailboxes exist in the database. This could be the department name if you group mailboxes by department. Some companies indicate the mailbox size in the name so that the administrators know where to put mailboxes of a particular type and size when they are created. For example, UK Sales-1GB indicates users who belong to the U.K. sales department who have 1 GB mailboxes. Descriptive database names certainly work, but it becomes more difficult to think of good names to use after you have more than 20 or 30 databases to manage. See Chapter 9, "The Database Availability Group," for a more comprehensive discussion on how to name databases in large-scale deployments.

- **Connectors** Messaging connectors should have names that clearly indicate their purpose and the type of traffic they support, for example, SMTP to Internet or SMTP to Lotus Notes.

INSIDE OUT **Avoid retroactive naming policies**

Don't create a heap of objects and then attempt to apply a retroactive naming policy; it is dreadfully boring to have to go through objects to rename them. Take the time early on to decide on a naming convention and then communicate the convention with some examples to any administrator who has the permission to create objects in the organization.

Managing Recipients

Recipients cover the broad spectrum of any object that can receive mail, or mail-enabled objects. EAC groups four major types of these objects under Recipients. These are:

- Mailboxes
- Groups
- Resources (room and resource mailboxes)
- Contacts (including mail users)

The following sections discuss each of these recipient types and their management.

Recipient filtering

An email server such as Exchange deals with huge differences in terms of object numbers. The largest Exchange organizations running outside Office 365 have over half a million mail-enabled objects. However, the smallest organizations might deal with just a few dozen mailboxes. EAC is designed to handle anything from the smallest demonstration environment to the largest production deployment.

Clearly, only a limited amount of information can be presented on any computer screen. As it moves to the different types of recipients, EAC displays the initial set of objects in alphabetical order. To see different sets of objects, you must either scroll down through pages of data (tiresome if the object you want is a number of pages down) or use the EAC search capabilities.

EAC does not support the same kind of recipient-filtering capabilities as does EMC, which allows the console to focus on specific sets of objects such as all the mailboxes in a specific database. Instead, all the sections of EAC that might deal with large numbers of objects, including recipients, have a search box in which you can type some characters to request

EAC to display matching objects. Figure 5-5 shows how the search box works when dealing with mailboxes. In this instance, EAC has detected that you might be searching against the various name attributes stored for mailboxes. Not all mailbox properties can be used for searching. For example, you cannot search against the company attribute, but you can against the department.

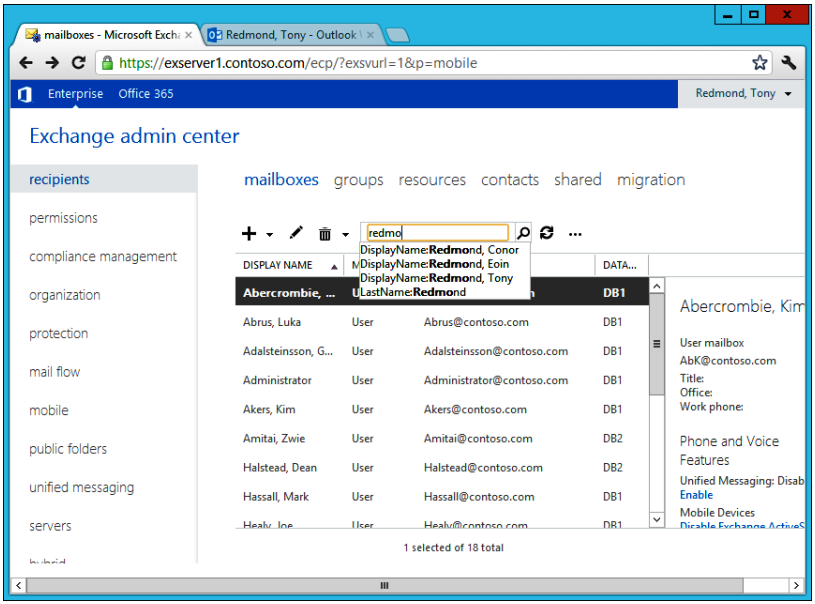


Figure 5-5 Searching for specific mailboxes in EAC

EAC also includes an Advanced Search option available by clicking the ellipsis. This option is available for all recipient types and can be used to look for any recipient of the selected type (mailboxes, groups, contacts, public folders), using values such as those stored in the 15 custom attributes available for Exchange mail-enabled recipients. Advanced Search is the closest to the kind of filtering that EMC offers and is just as powerful.

Exporting EAC information to CSV files

EAC includes an option to export the current list view to a comma-separated values (CSV) file. CSV files are often used to work with Exchange data because they can be loaded into applications such as Microsoft Excel and Microsoft Access and then manipulated before being reused for some purpose. For instance, the standard method used by Exchange 2013 to move mailboxes is to create and process migration batches (see Chapter 10, "Moving Mailboxes," for more information). You can input a set of mailbox names to be moved, or you can provide EAC with a CSV file containing the set. Say you wanted to move all the mailboxes that are currently in database DB2 to a new database. To do this, you'd specify

database = DB2 in an advanced search. After you execute the search, EAC displays all the mailboxes in the database. You can then click the ellipsis and choose to export the current view to a CSV file. You then select the fields to be exported (only the email address field is necessary for a migration batch) and click Export. EAC generates a CSV file you can open with Notepad (Figure 5-6).

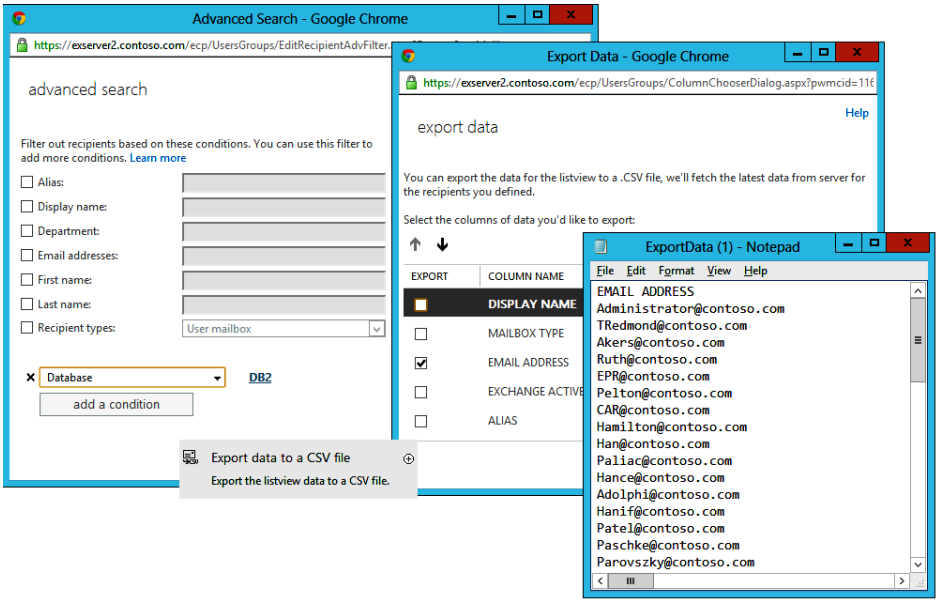


Figure 5-6 Building a CSV file from mailbox information

This is a very simple example of how mailbox information can be exported and reused within Exchange 2013. You can export information to a CSV file by using similar steps for any of the object types that are managed through the recipients section.

Some mysterious mailboxes

Although they exist, EAC does not reveal the existence of system mailboxes. In this respect, you have:

- Arbitration mailboxes, including the discovery search mailboxes (see the "Discovery mailboxes" section later in this chapter). Exchange uses arbitration mailboxes for many purposes, including message moderation. To see details of these mailboxes, you execute the Get-Mailbox –Arbitration command.
- Mailboxes created to test and measure system health. Exchange creates these for use by the Managed Availability system to test that different components of the

system are working correctly. For example, Exchange uses health mailboxes to send messages to each other to verify that the transport system is working as expected. You can see the set of health mailboxes within an organization by running the Get-Mailbox –Monitoring command.

- Mailboxes created for test purposes. For example, the New-TestCASConnectivityUser .ps1 script provided with Exchange creates a mailbox called extest_867e89ec8f7b4.

These mailboxes have a clearly intended purpose; otherwise, Exchange would not create them. For this reason, you should not delete or otherwise change their Active Directory accounts or mailbox details. The exception to this rule is when you want to remove a mailbox database that contains some arbitration or discovery mailboxes. In this case, you always need to move these mailboxes out of the database before you can remove it.

The need for mailboxes

Although it might be nice to be able to boast about the sheer scale of the organization you manage when you meet your peers at events such as the Microsoft Exchange Conference, managing a successful Exchange deployment is not determined by the number of mailboxes in the organization. Before you rush to create any mailboxes, you should lay out some guidelines for when mailboxes are created and when they are removed. Best practice for mailbox maintenance includes the following important points:

- Applications don't need mailboxes. Some administrators assume that it is a good thing to assign mailboxes for use by applications that need to create and send messages, usually by submitting a text message to an SMTP server. Applications do not need mailboxes for this purpose because they can create and submit messages to an SMTP server that supports submission from anonymous senders. The easiest way to support email submission for applications is to use the transport pickup directory. (For more information, see *Microsoft Exchange Server 2013 Inside Out: Connectivity, Clients, and UM*, by Paul Robichaux [Microsoft Press, 2013], Chapter 2, "The Exchange transport system.") If you do create mailboxes for application use, make sure that you secure the accounts associated with the mailbox so that they are restricted.

- Mailboxes have different types for a reason. Although it might seem fine to use normal mailboxes for resources (rooms and equipment), Exchange has a purpose behind the differentiation that it supports across mailbox types. Resource mailboxes are tied to disabled Windows accounts, and user mailboxes are not. Site mailboxes also use disabled Windows accounts. When you start to use normal mailboxes for resources, you create a potential security issue. Always assign the right mailbox type when you create a mailbox.

- Mailboxes shouldn't be kept forever. The information in a mailbox belonging to someone who leaves the company is probably of some interest, but interest wanes over time, and the information contained in most mailboxes belonging to an employee who has recently left is probably useless after three months. There will be exceptions, including mailboxes belonging to executives, which might be needed if an eDiscovery search is required for local information to respond to a legal action. Nevertheless, you should agree on guidelines to govern when mailboxes can be removed and make sure that old mailboxes and old Windows accounts don't linger past their best-by date (see Chapter 11, "Compliance").

Note

Apart from anything else, old mailboxes and accounts could represent a security weakness that an attacker can exploit. Some companies move all mailboxes belonging to departed employees to a special database so that they are grouped and are obviously different from live mailboxes.

Audit mailboxes regularly. You don't want to pay Microsoft any more for Client Access Licenses (CALs) than you should. CALs are often calculated on the basis of mailbox numbers, so it follows that keeping unnecessary mailboxes costs money. You should audit the mailboxes that exist in the organization at least every six months and remove any unused mailboxes. It's easy to report the last time a list of mailboxes in a database were logged on to detect potentially unused mailboxes. For example, the following command fetches details of all user mailboxes in a database and sorts them according to the last time a user logged on to the mailbox. Users who have never logged on to their mailbox are indicated by an error when Get-MailboxStatistics attempts to retrieve information from the mailbox. Other information that might indicate an unused mailbox, such as the total number of items in the mailbox, is also included. This report shows that approximately two months separate the most recent logon (my mailbox) from the oldest. It's reasonable to suspect that the mailboxes that have not been accessed in two months are no longer needed, or at least that they can be marked as being suitable for potential deletion if not required for regulatory purposes.

```
Get-Mailbox -Database DB2 -RecipientTypeDetails UserMailbox | Get-MailboxStatistics
| Sort-Object LastLogonTime | Format-Table DisplayName, LastLogonTime, ItemCount,
TotalItemSize
```

With these points in mind, you can create and manage some mailboxes.

Naming mailboxes

Email address policies enable you to define and apply different patterns for the SMTP addresses that are assigned to mail-enabled objects. The application of address policies makes sure that the SMTP addresses are consistent throughout the organization. Exchange 2013, unfortunately, does not provide a mechanism to control the generation of display names, which is the attribute that Exchange uses to sort objects in the GAL and EAC and for recipients and authors in message headers.

Table 5-1 lists the different attributes for the various names or name components Exchange uses that are stored in Active Directory. The default pattern for display names is %g %s; in other words, first name<space>last name or, in my case, Tony Redmond. This is an acceptable naming convention for small implementations in which everyone knows everyone else, and it is easy to find the correct recipient by browsing the GAL, but it becomes increasingly difficult to find people as the number of directory entries increases. The question, therefore, is what naming convention to use that is efficient and logical for users when they search for an object in the GAL. More variation occurs in surnames than in given names. Common given names, such as John or Mary, occur thousands of times in a large GAL, so if the GAL is sorted by given name, you might have a tiresome search before you locate the right recipient. It is easier to search using a surname, even with common surnames such as Smith, Chan, or Ng. Telephone directories are organized by surname, so it makes sense to carry the analogy forward and do the same thing for the GAL.

TABLE 5-1 Mailbox attributes and names

Attribute	Meaning
Alias	Unique name for the object
Name	Full name of the object composed of first name and last name
FirstName	First name of the user
LastName	Surname of the user
DisplayName	Name used to sort the GAL and for other display purposes (such as EAC and in message headers)
DistinguishedName	Name used to identify object in Active Directory
PrimarySMTPAddress	Primary SMTP email address (often first name.last name@domain)
UPN	User Principal Name, or the name of a user in email format that can be used to log on to a Windows server; recommended that the UPN has same value as the primary SMTP email address

INSIDE OUT Applying a different naming convention

Although the sequence used in naming conventions is hard-coded in EAC, it is possible to alter a convention. If you want to apply a different naming convention, the usual approach is to:

- **Allow EAC to create the mailboxes and contacts as normal and subsequently edit the Display Name.**
- **Create mailboxes and other mail-enabled recipients by using EMS scripts so that you have complete control over the format used for display names.**

There might be other circumstances in which you have mailboxes for which you don't want to use the last name, first name convention, such as those used for discovery results, but these can be dealt with on an exception basis. Figure 5-7 shows how Microsoft Outlook 2013 displays a GAL in which the mailboxes are organized using the last name, first name convention. As you can see, some of the entries have additional information to identify individuals who share common names or who have particular functions within the company. It is common to use department names, locations, or job titles to help users identify the correct recipient.

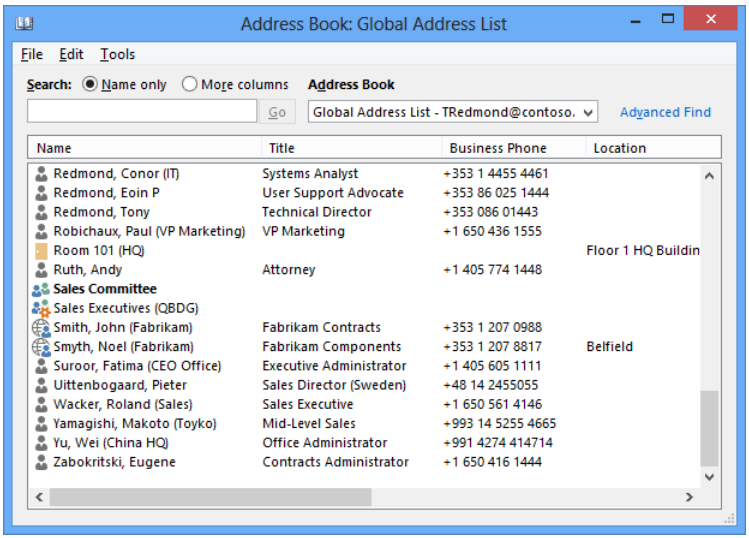


Figure 5-7 A well-ordered GAL

One problem with adding some detail to display names that deserves consideration is that it might expose some confidential information outside the company. For instance, assume that you have two people named John Smith in the organization, and you want to help users locate the correct person in the GAL, so you create display names that identify the two individuals by adding the department in which they work in parentheses. Thus, you might have:

- Smith, John (Corporate Acquisitions)
- Smith, John (IT Standards)

These names help users know which of the two John Smiths to whom they should address messages. However, the issue arises when the names of the departments are also communicated to recipients outside the organization. Anyone who receives a message from either John Smith will know for which department he works. This might not be important for generic departments such as Sales or Marketing or locations such as Dublin or New York, but it could be for departments such as Corporate Acquisitions or Talent Management, both terms that convey a lot about the role the user holds within the company. The lesson here is that you need to think about whether it matters if people outside your company know the information you add to display names to identify people in the GAL.

Some companies like to impose a special naming convention for distribution groups so that users know when they are sending a message to a group rather than to an individual recipient. The Exchange MailTips feature helps here; it can either warn users when they address a message to a large group or display a tailored tip to indicate the purpose of the group. One solution is to prefix groups with some characters. For example, you could create a group naming policy, DG, as a prefix so that your groups would have names such as DG: Sales Executives and DG: IT Department. The advantage of this approach is that all the groups are found in a single location in the GAL. Some take the idea further and use a prefix such as ## that places all groups at the start of the GAL. You then have names such as ## Sales Executives. This approach works but is not as user friendly as the other one.

Creating new mailboxes

Creating a new mailbox with EAC is easy. Open Recipients, select Mailboxes, and click the + (plus) sign to expose the dialog box to collect details about the new mailbox (Figure 5-8). Exchange 2013 supports the following mailbox types:

- **User mailboxes** The standard full-function mailboxes used by people to send and receive email and work with calendar, contacts, and other mail-enabled applications. User mailboxes can be associated with archive mailboxes to provide secondary longer-term storage. Archive mailboxes are discussed in Chapter 10.

- **Linked mailboxes** Mailboxes that are linked to a user account in a separate, trusted forest.
- **Shared mailboxes** Mailboxes that are accessed by multiple users who share a need to work with common information such as messages, calendars, and tasks; for example, the members of a project team.
- **Room mailboxes** Mailboxes that represent conference rooms so that users can book them for meetings through calendar requests.
- **Resource or equipment mailboxes** Mailboxes that represent other items of equipment (such as projectors, communication equipment, or whiteboards) that users might need to reserve for a meeting.
- **Discovery mailboxes** Mailboxes that are used to store items copied as the result of eDiscovery searches. The creation of discovery mailboxes is discussed in the "Discovery mailboxes" section later in this chapter.

Exchange 2013 also supports public folder and site mailboxes, both of which are discussed in Chapter 12, "Public folders and site mailboxes." Public folder mailboxes can be created through the public folder section of EAC, but you cannot create site mailboxes through EAC because they are created in SharePoint 2013 as part of the process of setting up a site. As for the other types, EAC groups user and linked mailboxes under Mailboxes and room and resource mailboxes under Resources.

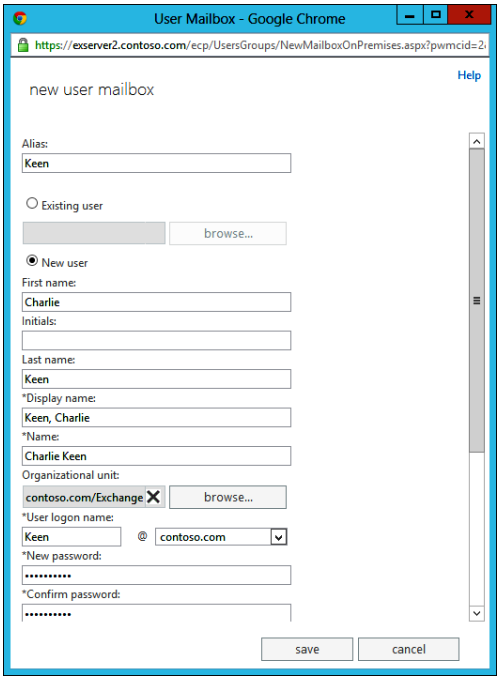


Figure 5-8 Creating a new mailbox

The fields presented to create a new mailbox are:

- **Alias** This value doesn't have to be unique, but it should be. There's no way to request EAC to determine whether an alias is unique; you have to do this yourself before attempting to create a mailbox. Exchange enables you to input an alias that already exists for an object, but this is bad practice because subsequent requests to find mailboxes might return multiple objects. If a mailbox is to be created for an existing user, you can select from the list of Windows accounts that are not already associated with an Exchange mailbox.
- **First name** The user's given name.
- **Initials** Any initials for the user, if they exist.
- **Last name** The user's surname.
- **Display name** By default, EAC creates a display name from the user's first name, a space, and the user's surname. This is fine if you want the GAL to be organized using this order. If you want to use a different order or to add some information to help users identify people in the GAL, you have to overwrite the display name with your preferred value.

- **Organizational unit** If left blank, EAC creates the new user account in the Users organizational unit (OU). Unless you operate a very small organization, it is much better if all your user accounts are not created in a single OU but rather divided into easy-to-manage units.

- **User login name** This value is used to create the UPN, Windows Standby Active Manager (SAM) account name, and primary SMTP address for the new mailbox, and it must be unique. It is recommended for the UPN to be the same as the primary SMTP email address for the mailbox.

- **Password** EAC requires you to set a password for the new Windows account and confirm the value by retyping it. You can also decide whether the user should reset his password when he first logs on.

Below the password information, you see More Options. This exposes fields for:

- **Mailbox database** You can decide to create the new mailbox in a specific database. If you don't, Exchange selects a database from any of the set available for auto-provisioning.

- **Archive** You can assign an archive mailbox to the new mailbox and select the database within which the archive mailbox is created. Note that archive mailboxes require an enterprise CAL.

- **Address book policy** You can select from available address book policies and assign one to the mailbox if the user is subject to a customized address book policy that allows her to see only a subset of the overall GAL.

It's obvious that the foregoing covers only a small proportion of the available properties that you can set to configure a new mailbox to meet your exact requirements. After the new mailbox is created, you can edit its properties to fit before handing the new mailbox over to its user.

Integrating mailbox creation into overall company policy

Before plunging into the details of how to create a new mailbox, it's important to recognize that creating a mailbox might be only one small part of a complete user onboarding process that covers multiple activities, including:

- **HR provisioning Allocating employee number, creating employee record in human resources (HR) system, generating identity badge, filing application for corporate credit card, and so on.**

- **IT provisioning Allowing access to operating systems (Windows might be only one of the operating systems the company uses), applications including email,**

instant messaging, telephony, and Web repositories or document management systems; allocating mobile devices; and provisioning security tokens or keys necessary to allow virtual private network (VPN) access to the company network from the Internet.

- **Facilities provisioning Assigning office space and physical items required by the employee, including PC and printer.**

Many large companies operate sophisticated workflow applications that take care of many of these activities. A similar workflow is usually present to remove access and "deprovision" an employee who leaves the company. Because the on-boarding process can be so complicated and require interaction with so many applications, it's wise to consider how you will integrate the creation of an Exchange mailbox into the process. For example, will you build a step into the workflow to create a mailbox automatically and have the characteristics of the mailbox, such as an appropriate quota, enabling an archive, and applying the correct retention policy, set automatically, depending on the employee's level and job code, or will you have a request generated and sent by email to the help desk? Microsoft provides a prototype tool to help with the creation of Active Directory, Exchange, and Lync accounts at *http://gallery.technet.microsoft.com /office/Z-Hire-Employee-Provisionin-e4854d6b* that could be used as the basis to build a suitable account provisioning tool for your company.

The most common problems administrators encounter when they attempt to create mailboxes with EAC are the following:

- Failure to provide a password for a new account that meets Windows requirements. If you enter a password that does not meet the policy implemented for Windows (for example, it contains the name of the user), EAC flags the error "The value provided for the new password does not meet the length, complexity, or history requirements of the domain" and will not be able to proceed to create the new mailbox.

- Inadvertent selection of features that require an enterprise CAL. Assigning a feature such as an archive mailbox increases the number of enterprise CALs your organization requires. EAC indicates whenever a feature requires enterprise CALs, but this information is often overlooked by stressed administrators who are focusing on the task at hand rather than reading all the text presented on screen, so the subsequent increase in Exchange client license fees might come as an unpleasant surprise at some point in the future.

Setting up mailboxes by entering values through EAC is easy if you have only a few mailboxes to create. When the load increases, it's time to master the intricacies of the cmdlets that are used, especially if you need to integrate mailbox creation with another process,

such as that used by the HR department to introduce a new employee to the company. Some prototype code that creates a new mailbox with a new Windows account is shown next. Table 5-2 lists the most common parameters to use when creating a new user mailbox complete with a Windows account. As you see later, other types, such as room and equipment mailboxes, use different parameters:

```
New-Mailbox -Name 'Rich Haddock' -Alias 'HaddockR' -UserPrincipalName 'Rich.Haddock@
contoso.com' -SamAccountName 'HaddockR' -FirstName 'Rich'
-Initials 'J' -LastName 'Haddock' -Password (ConvertTo-SecureString 'Exchange2013!'
-AsPlainText -Force) -ResetPasswordOnNextLogon $True -Database 'DB1' -DisplayName
'Haddock, Rich (Sales)' -OrganizationalUnit 'contoso.com/Exchange Users'
```

Creating a mailbox is only the start of the process of building a mailbox that is fully ready to use. The code that you've just viewed doesn't exploit the ability to assign special mailbox quotas, set up moderation for the mailbox, apply ActiveSync or retention policies, or create an archive mailbox. All these settings can be enforced as you create the new mailbox with the New-Mailbox cmdlet or immediately afterward with the Set-Mailbox cmdlet. In addition, Exchange provides a set of cmdlets to manipulate settings such as language and regional settings and autoreply and calendar settings. See the "Manipulating mailbox settings" section later in this chapter for more information on this topic.

TABLE 5-2 Properties used to create a new mailbox with New-Mailbox

Property	Use	Mandatory
Name	Name of the mailbox.	N
Alias	Unique identifier for the mailbox. This value can contain no special characters.	Y
UserPrincipalName	UPN; an identifier for the mailbox in SMTP format.	Y
FirstName	First name for the mailbox owner.	N
Initials	Initials for the mailbox owner.	N
LastName	Surname for the mailbox owner.	N
DisplayName	Display name for the mailbox as used in the GAL and message header. If omitted, Exchange creates a display name based on the language used to run EAC. For example, English language variants of EAC create display names from the first name and last name.	N
Password	Password for the Windows account. If omitted, Exchange prompts for a password for the new account.	Y
ResetPasswordOnNextLogon	Flag to indicate whether user will be forced to reset his Windows password the next time he logs on.	Y

Property	Use	Mandatory
Database	Database in which the new mailbox will be created.	N
OrganizationalUnit	Active Directory OU in which to create the new Windows account. If omitted, Exchange uses the default OU.	N
ActiveSyncMailboxPolicy	Name of the ActiveSync policy to apply to the new mailbox. If omitted, Exchange applies the default ActiveSync policy.	N
Archive	Flag to indicate whether an archive mailbox is created.	N
ManagedFolderMailboxPolicy	Name of the Managed Folder policy to apply to the mailbox.	N

If you select the option to create a mailbox for a Windows account that already exists, EAC uses the Enable-Mailbox cmdlet to create a new mailbox and associate it with the Windows account you select. The code is much simpler because the Windows account already has many established properties that you'd otherwise have to provide. All you need to state for the new mailbox is an identity, an alias, and a target database, but I also tend to add a display name (to make sure that the GAL is correctly populated) and a primary SMTP address. For example:

```
Enable-Mailbox -Identity Ccontoso.com/Exchange Users/Kevin Kelly' -Alias 'KellyKevin'
-Database 'DB2' –DisplayName 'Kelly, Kevin' –PrimarySmtpAddress 'Kevin.Kelly@contoso.
com'
```

Immediately after the new mailbox is created, Exchange applies the appropriate email address policy to create suitable email addresses for the mailbox and updates Active Directory with these values. See the "Email address policies" section in Chapter 7, "Addressing Exchange," for more information about how to create email address policies.

Completing mailboxes

You now have a new mailbox. Although it is functional in that it could be used to send and receive email, in the eyes of an administrator, the new mailbox is incomplete. Many operations might be performed on a new mailbox before it is completely ready to be handed over to the tender mercies of its owner. Tasks that might be performed include the following:

- Add the new mailbox to distribution groups. For example:

  ```
  Add-DistributionGroupMember -Identity 'DG Dublin Users' -Member 'Kelly, Kevin'
  –BypassSecurityGroupManagerCheck
  ```

- Update mailbox properties so that the new mailbox is discovered by the queries used to determine the membership of dynamic distribution groups (we'll discuss how those queries are resolved in Chapter 6). In this example, you put a value in one of the customized attributes:

  ```
  Set-Mailbox -CustomAttribute1 'Sales' -Identity 'Kelly, Kevin'
  ```

- Update other mailbox properties that are not revealed by EAC. Many of the properties people associate with a mailbox are actually updated with the Set-User cmdlet rather than the Set-Mailbox cmdlet because they belong to the base Active Directory user object rather than to the mailbox. In this example, you update a number of properties with the Set-User cmdlet to fill out the directory information for the new mailbox:

  ```
  Set-User -City 'Sydney' -Company 'Contoso' -CountryOrRegion 'Australia'
  -Department 'IT' -Manager 'Redmond, Tony (IT)' -Office 'Sydney'
  -Phone '8170-19944' -Title 'VP APJ' -Identity 'Bracy, Colleen (IT)'
  ```

- Enable a personal archive for the mailbox. This can be done during the initial creation, but some administrators prefer to leave this action until after users confirm that they want an archive.

  ```
  Enable-Mailbox –Archive –Identity 'Kelly, Kevin'
  ```

It can take quite a long time to populate all the properties for a new mailbox fully. For this reason, many administrators script the process of creating a new mailbox and use the EAC only for one-off mailbox creation. The scripts used to create a mailbox range from moderately simple (basically a collection of one-line EMS commands) to very complex. In the latter case, the scripts usually accept a feed from an identity provisioning or HR management system that maintains information about employees and produces the necessary commands to create a ready-to-go mailbox.

INSIDE OUT Be nice to new users—send them a message

It's a nice idea to send new users a welcome message that gives some advice on effective use of their mailbox and how they can find additional resources if they have any questions. Pat Richard, a longtime MVP, has written a great script for this purpose. You can download the Send-NewUserWelcome.ps1 from *http://www.ehloworld.com/175* and customize the code to meet the needs of your organization.

Bulk mailbox creation

The methods used to create new mailboxes range from once-off operations to sophisticated scripts used to generate hundreds or thousands of mailboxes following a company merger. Some companies write code to integrate mailbox creation into their new employee on-boarding process, whereas others are happy to send support tickets to their administrators to request new mailboxes on an on-demand basis. Using EAC to create a bunch of mailboxes isn't much fun because the joy of navigating multiple screens rapidly becomes a tedious slog after you've done it once or twice.

Because Exchange has supported scripting since Exchange 2007, many EMS scripts are available on the Internet that can be used to automate the bulk creation of mailboxes, including the population of properties such as retention policies and the addition of new mailboxes to distribution groups. A simple search will reveal many scripts that you can download and amend to create a bulk load script that's appropriate for your organization.

Creating new room and resource mailboxes

The code to create a new room mailbox is similar to that used to create a user mailbox, except that you use the –Room parameter to mark it as a room mailbox. For example:

```
New-Mailbox -Name 'Conference Room A' -Alias 'Confa' -OrganizationalUnit
'contoso.com/Exchange Resources'  -UserPrincipalName 'confa@contoso.com'
-SamAccountName 'Confa' -FirstName 'Conference' -Initials '' -LastName 'Room A'
–Room
```

To see a list of current rooms, use this command:

```
Get-Mailbox –Filter {RecipientTypeDetails –eq 'RoomMailbox'}
```

The code to create a new equipment mailbox is also similar. In this case, the –Equipment parameter is passed to mark the mailbox as an equipment mailbox. For example:

```
New-Mailbox -Name 'Projector Conference A' -Alias 'ProjConfA'
-OrganizationalUnit 'contoso.com/Exchange Resources'
-UserPrincipalName 'ProjConfA@contoso.com'
-SamAccountName 'ProjConfA' -FirstName 'Projector' -Initials ''
-LastName 'Conference A'  –Equipment
```

To see the current equipment mailboxes, type:

```
Get-Mailbox –Filter {RecipientTypeDetails –eq 'EquipmentMailbox'}
```

Room and equipment mailboxes are created with disabled Windows accounts. It is a good idea to group these accounts in their own OU and to consider assigning these mailboxes to their own database to isolate them clearly from regular mailboxes.

Languages

Exchange 2013 is a highly capable multilingual product that supports server components running in at least 11 languages and client components (Outlook Web App and EAC) in over 60 languages from Amharic to Welsh. The exact count increases all the time as Microsoft completes the task of translating the language-dependent parts of the product into a new language. See *http://technet.microsoft.com/en-us/library/dd298152.aspx* for the latest information on the list of supported languages. All the necessary code to support the different client languages is installed on servers that support the mailbox role. Only the language-dependent server components are installed on servers that support the CAS role. Thus, a multirole server has both server and client language-dependent capabilities.

Users select the language they want to use with Exchange the first time they connect to a mailbox with a client. Sometimes the selection is implicit, as when you connect with a client such as Outlook that's already running in a selected language, and sometimes it is explicit, as when someone uses Outlook Web App to open a mailbox, and a language setting is not present for the mailbox. Every time it starts, Outlook Web App checks whether the language and time zone values are set for the mailbox. If one or both of these values are missing, Outlook Web App displays the dialog box shown in Figure 5-9 and pre-populates the selection with whatever language is the default on the workstation. Of course, there is no compulsion on the user to accept the proposed language, and he is free to select from any of the languages Exchange supports. After the selection is made, Exchange updates the languages property of the user's mailbox, and Outlook Web App continues by displaying the mailbox contents in all the glory of the selected language. You can check on the language any user uses as follows:

```
Get-Mailbox –Identity TRedmond | Select Languages
```

You can set the user's language with the Set-Mailbox cmdlet. For example, to force the user to use Welsh, you execute the following command:

```
Set-Mailbox –Identity Young –Languages "cy-gb"
```

The next time the user connects to Outlook Web App, she will see Outlook Web App in Welsh (Figure 5-10). Exchange cannot do anything about translating the messages sent to a user into the user's preferred language (perhaps this will be a feature in a future release), but all the text strings and command prompts are translated into the chosen language.

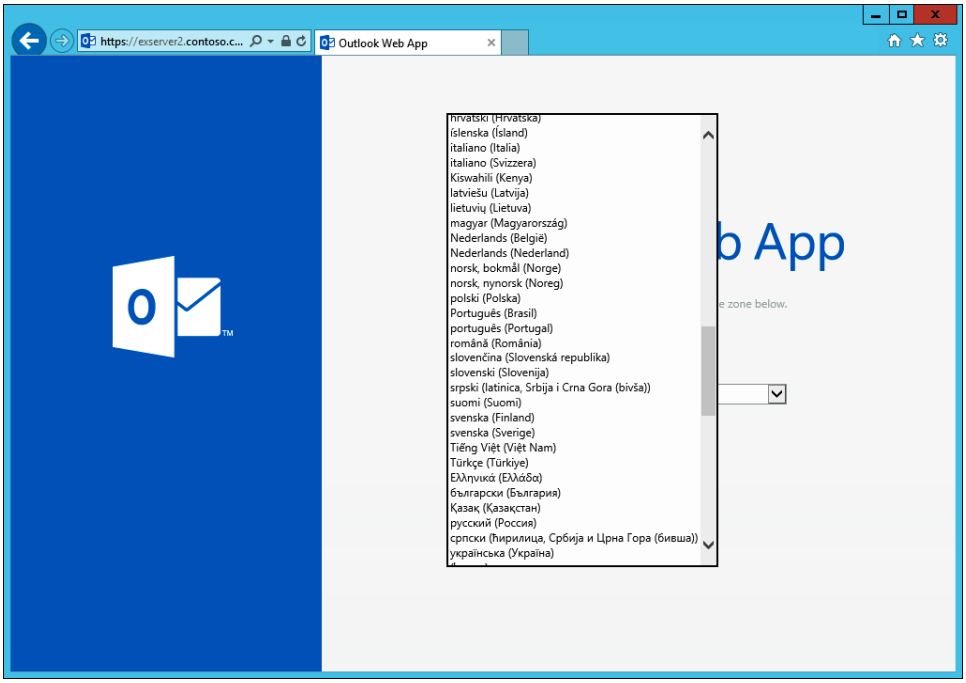


Figure 5-9 Selecting a language to use with Outlook Web App

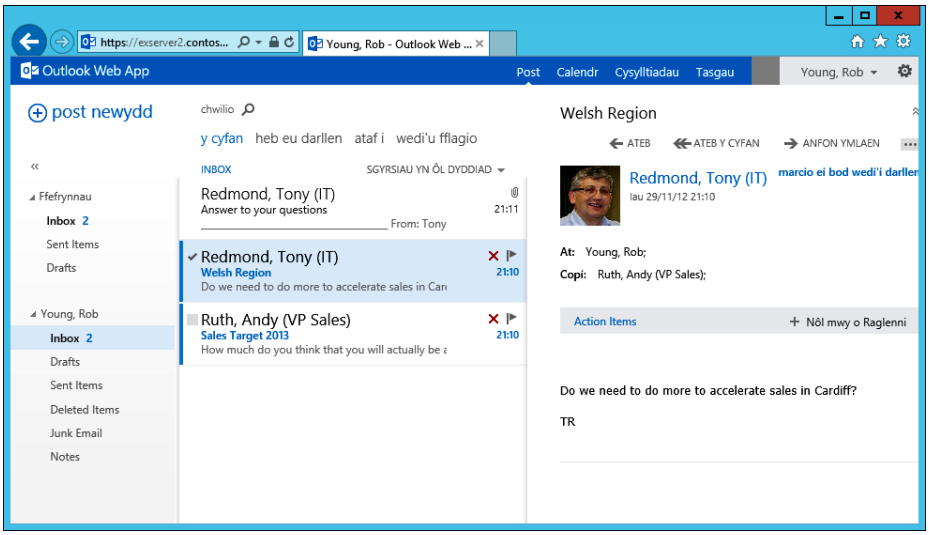


Figure 5-10 Running Outlook Web App in Welsh

Of course, if the user connects using a client that includes language-dependent code installed on a workstation such as Outlook, the change in language will have no effect because Outlook cannot switch languages in the same way Outlook Web App can. Outlook Web App has this ability because all the language-dependent text strings and other components it needs are installed on the Mailbox server, and Outlook Web App can retrieve them from there.

EAC uses the same language value as Outlook Web App to determine which language-dependent text it should present to a user. Assume that you assign the person shown using Outlook Web App previously the Server Management RBAC role. If she then started EAC, she'd see something like the screen shown in Figure 5-11. Note the effect of RBAC in that EAC has recognized that users who hold the Server Management role should not be able to see all the options available to more privileged administrations such as those who are members of the Organization Management role group.

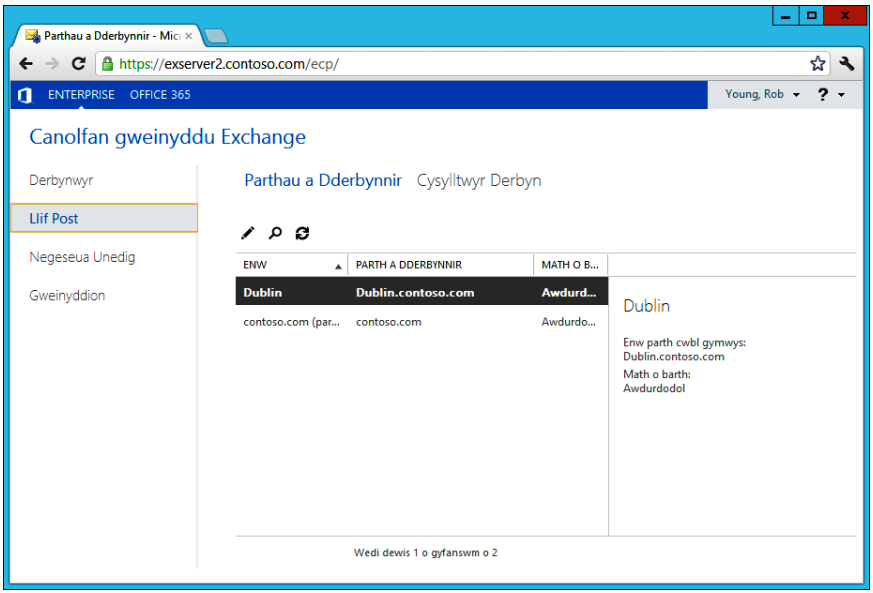


Figure 5-11 Running EAC in Welsh

For those who are interested, "cy" stands for "Cymru," which is Wales in Welsh, whereas "gb" indicates that Welsh is a language of the GB country. The same approach is taken to the different variations of English, in which you have the following:

- En-us English as in the United States
- En-ie English as in Ireland

- En-gb English as in the United Kingdom
- Some variations of English, such as en-au (Australia), which are not supported

You can assign multiple language values to a mailbox. For example, in this instance, you update a mailbox to create a list of languages comprising Welsh, U.K. English, and French. In most cases, this won't have much effect because Exchange always selects and uses the first language it finds in the list unless those language components are not available on the Mailbox server. Because Exchange 2013 installs all languages, the language the user prefers should always be available.

```
Set-Mailbox -Identity Young -Languages "cy-gb", "uk-gb", "fr-fr"
```

Some companies like to set default language and time zone values as part of the process of creating new mailboxes. Exchange facilitates this through the Set-MailboxRegionalConfiguration cmdlet, which enables administrators to provide a full set of regional information for a mailbox. This example sets my mailbox to use the Ireland variant of English as the mailbox language together with the appropriate time zone, date format, and time format for the region:

```
Set-MailboxRegionalConfiguration -Identity 'Tony Redmond' -Language 'en-IE'
-TimeZone 'GMT Standard Time' -DateFormat 'dd/MM/yyyy' -TimeFormat 'H:mm'
```

INSIDE OUT **Date and time formats are case sensitive**

Be careful with the format used to specify dates and times; Exchange rejects a setting if it is inappropriate for the selected time zone or doesn't match the required mask. Uppercase and lowercase characters are important. For example, d/M/YY will be rejected as a valid date format for the en-ie locale, whereas d/M/yy is acceptable. EMS tells you the values you can use when it signals the error. Alternatively, you could use Outlook Web App to set the regional options for a mailbox and then use the Get-MailboxRegionalConfiguration cmdlet to check the values written to the mailbox properties; then you could replicate those properties to other mailboxes as they are created.

INSIDE OUT Running EAC in another language

EAC will use the language specified in your mailbox when it loads its UI. However, some users who run EAC do not have mailboxes, as in the case of administrators who use highly permissioned accounts when performing system management and are required to log on to their personal accounts to use email. In this instance, EAC uses clues such as the regional settings for the workstation or the language selected for the browser to decide which language it should use. You can override the selection by including your choice in the URL entered to start EAC. For example, to start EAC in French, you would enter *https://server-name/ecp?mkt=fr-fr*.

Default folders

Exchange does not create the actual mailbox in the assigned database until the user first logs on to the mailbox or, if required, to deliver a message to the new mailbox. At this point, Exchange creates the various special folders it needs to process email, including the Inbox, Sent Items, and Deleted Items or the equivalent in a selected non-English language.

For instance, if a user selects Polish as her preferred language, the Inbox is then displayed as "Skrzynka odbiorcza," Sent Items becomes "Elementy wyslane," and Deleted Items is "Elementy usunięte," whereas switching to Portuguese produces "A Receber," "Itens Enviados," and "Itens Eliminados."

Exchange 2010 also enables users to switch languages, but if you switch from, say, English to French, the names of the special folders remain in English. This is an unsatisfactory situation, so a solution is presented in Exchange 2013 by which you can instruct Exchange to update the names of the special folders when you choose to use a different language (Figure 5-12). Note that you won't see this option presented if you are switching to a language that uses the same values for special folder names as your current language, as in the case of switching from English (U.S.) to English (Ireland) (en-us to en-ie) behind the scenes.

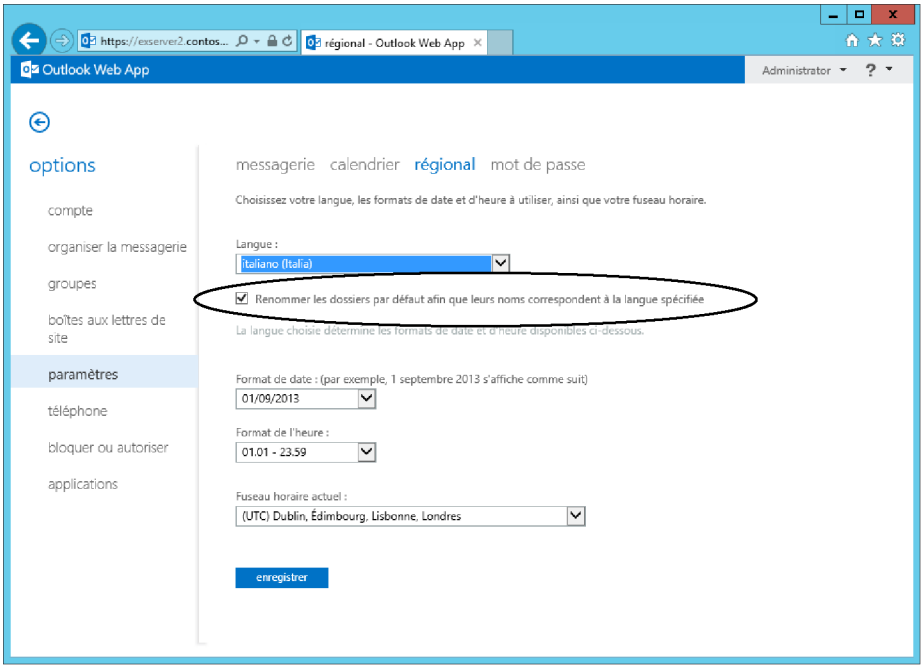


Figure 5-12 Opting to translate the names of the special folders

PST files are a special case

Exchange does not respect the locale of a PST file if you import content from it to populate items in a new mailbox. For instance, a PST that has been used with the French version of Outlook will include some of the French special folder names. (At the very least, a PST contains the language-dependent version of the Deleted Items folder; the other special folder names are created if you use the PST to receive new mail with Outlook.) If you now import the items from the PST into a new mailbox created on a server running an English locale (in other words, the Windows server where the import is executed is configured in English), Exchange creates English-language special folders before it imports data from the PST. You therefore end up with two sets of special folders, one in French and one in English. For this reason, it's best to be explicit with Exchange and instruct it about what the preferred language should be for a mailbox before you import any data into it.

Manipulating mailbox settings

Exchange provides a set of cmdlets by which administrators can query and set properties of mailboxes. You can think of this as administrative access to the options that users have to configure their mailboxes through Outlook or Outlook Web App. These properties are stored as hidden items in the root of user mailboxes and are not accessible to administrators in previous versions of Exchange. Table 5-3 lists the elements of mailbox configuration that administrators can manipulate and the cmdlets used for this purpose.

TABLE 5-3 Mailbox configuration cmdlets

Purpose	Cmdlets
Change regional preferences (language, date format, time zone)	Get-MailboxRegionalConfiguration Set-MailboxRegionalConfiguration
Change the auto-reply settings	Get-MailboxAutoReplyConfiguration Set-MailboxAutoReplyConfiguration
Change mailbox calendar settings and how the Calendar Assistant processes incoming requests	Get-MailboxCalendarConfiguration Set-MailboxCalendarConfiguration Get-CalendarProcessing Set-CalendarProcessing
List or add mailbox folders	Get-MailboxFolder New-MailboxFolder
Change general mailbox settings	Get-MailboxMessageConfiguration Set-MailboxMessageConfiguration
Change spelling settings	Get-MailboxSpellingConfiguration Set-MailboxSpellingConfiguration
Change junk mail settings	Get-MailboxJunkEMailConfiguration Set-MailboxJunkEMailConfiguration

Before you attempt to query or set the properties of a mailbox, make sure that you have the necessary access rights to the mailbox. The Add-MailboxPermission cmdlet can be used as follows:

```
Add-MailboxPermission –Identity 'Rob Young –User 'Europe Help Desk' -AccessRights
FullAccess
```

Mailboxes assume a default regional configuration based on the language and regional settings of the server on which they are created. If your server runs the U.S. English version of Exchange in the Pacific Time Zone, any mailboxes you create on that server inherit these settings along with other associated settings such as date and time format. Users can change these settings with a client, such as the options Outlook Web App presents, to select a language the first time a user logs on to his mailbox. To retrieve the default regional configuration for a mailbox:

```
Get-MailboxRegionalConfiguration –Identity 'John Smith'
```

Use the Set-MailboxRegionalConfiguration cmdlet to tweak the regional settings. In this example, the mailbox language, time zone, and date format are set:

```
Set-MailboxRegionalConfiguration –Identity 'John Smith' –Language 'Es-es'
–TimeZone 'Eastern Standard Time' –DateFormat 'dd-mmm-yyyy'
```

Get-MailboxAutoReplyConfiguration and Set-MailboxAutoReplyConfiguration query and set the auto-reply settings for a mailbox. Users receive auto-reply messages when they send email to mailboxes that have an auto-reply message configured and enabled. Exchange also uses the auto-reply message as a MailTip that is displayed to users when they address a new message to a user who has auto-reply enabled. To find out the mailboxes in a database that have auto-reply currently set and the time the auto-reply lapses, you can use the following command:

```
Get-Mailbox –Database 'VIP Data' | Get-MailboxAutoReplyConfiguration | Where
{$_.AutoReplyState –eq 'Scheduled'} | Select MailboxOwnerId, EndTime
```

If you review the auto-reply data maintained for a mailbox, you can see the auto-reply text the user has created to send to internal and external recipients:

```
Get-MailboxAutoReplyConfiguration –Identity 'Kim Akers' | Format-List AutoReplyState,
EndTime, External*
```

```
AutoReplyState    : Scheduled
```

You can also set up a new auto-reply for a user who has gone on vacation and forgotten to let anyone know. In this instance, you enable auto-reply, set a time limit, and create separate messages for internal and external audiences. You also tell Exchange that auto-replies go only to external correspondents who are known contacts for the recipient.

```
Set-MailboxAutoReplyConfiguration -Identity 'Kim Akers' -StartTime '12/10/2013 19:30'
-AutoReplyState Enabled -EndTime '12/15/2013 07:00' –InternalMessage 'Kim Akers is on
vacation and will respond to your message after she returns on December 15'
–ExternalMessage 'Kim Akers is on vacation' –ExternalAudience 'Known'
```

To turn off auto-reply for a user:

```
Set-MailboxAutoReplyConfiguration –Identity 'Kim Akers' –AutoReplyState Disabled
```

Users might continue to see an auto-reply MailTip even after it expires or is disabled. This is because the Mailbox server caches auto-reply information to prevent it from having to look up mailbox auto-reply data every time a message is addressed. The cached data is refreshed every hour.

The Get-MailboxCalendarConfiguration and Set-MailboxCalendarConfiguration cmdlets query and set the properties of calendar settings. For example, this command configures

the calendar to use Greenwich Mean Time (GMT) as the time zone with a starting time for the workday of 8:30 A.M.:

```
Set-MailboxCalendarConfiguration -Identity 'Kim Akers' -WorkingHoursTimeZone 'GMT
Standard Time' -WorkingHoursStartTime 08:30:00
```

The Set-CalendarProcessing cmdlet determines how the Calendar Assistant processes calendar requests that arrive in mailboxes. Although it is possible for administrators to manipulate these settings for a user through the EMC console, you can't do this with EAC. Instead, a user can change his settings through Outlook Web App (Options, Settings, Calendar), but administrators are limited to EMS. For example, this command sets the properties that control how forwarded meeting notifications and external meeting requests are processed.

```
Set-CalendarProcessing -Identity 'Akers, Kim' -RemoveForwardedMeetingNotifications
$True -ProcessExternalMeetingMessages $True
```

The Set-CalendarProcessing cmdlet is capable of doing much more than manipulating the limited set of properties revealed through Outlook Web App, especially for resource mailboxes. See the Exchange help file for more information.

The Get-MailboxFolder and New-MailboxFolder cmdlets can be used to retrieve a list of folders in a mailbox or create a new folder. You can decide to retrieve just top-level folders or the complete set, or you can look for specific details about a folder and its children. Note that even highly permissioned users cannot use these cmdlets to interrogate the folders within another user's mailbox. By design, the cmdlets only work inside the mailbox owned by the user who runs EMS. Here's the command to retrieve a complete list of mail-related folders from the current user's mailbox:

```
Get-MailboxFolder -Recurse -MailFolderOnly
```

To create a new folder called Exchange 2013 in the top level of the Administrator's mailbox:

```
New-MailboxFolder -Parent 'Administrator' -Name 'Exchange 2013'
```

Get-MailboxMessageConfiguration and Set-MailboxMessageConfiguration retrieve and set general properties of a mailbox. For example, to define that Exchange should append an autosignature to every outgoing message and to provide the text for the autosignature:

```
Set-MailboxMessageConfiguration -Identity 'John Smith' -AutoAddSignature $True
-SignatureText 'From the desk of John Smith'
```

Get-MailboxJunkEmailConfiguration and Set-MailboxJunkEmailConfiguration query and set the properties that control a user's junk mail settings. The most interesting of these properties are the trusted senders and blocked senders lists. Both are multivalued properties, so you have to manipulate them in a variable before updating the property with new values, for example, to add the cohowinery.com domain to the blocked senders list. This cmdlet generates an error if you attempt to run it against a mailbox that has never been logged

on to by a user because Exchange doesn't create the junk mail configuration for a mailbox until after it is first used.

```
$List = Get-MailboxJunkEMailConfiguration –Identity 'John Smith'
$List.BlockedSendersAndDomains += 'cohowinery.com'
Set-MailboxJunkEMailConfiguration –Identity 'John Smith' –BlockedSendersAndDomains
$List.BlockedSendersAndDomains
```

Of course, users can overwrite these settings by using Outlook or Outlook Web App after you've updated any of their mailbox settings. If you have given your account the permission to access a mailbox, make sure that you remove the access after you've adjusted all the required regional settings. This example uses the Remove-MailboxPermission cmdlet to remove the full-access rights to the mailbox with which you have been working from the Europe Help Desk account.

```
Remove-MailboxPermission -Identity 'John Smith' -User 'Europe Help Desk'
-AccessRights FullAccess
```

Automating mailbox settings

Many properties can be set to control exactly how a mailbox functions, but some are more important than others. All the properties can be manipulated with EMS, and the most critical through EAC. However, it's easy to forget to update or set a property. Automation comes to the rescue in the form of the Exchange cmdlet extension agents (see *http://technet.microsoft.com/en-us/library/dd335054(v=exchg.150).aspx* for more information), a feature that first appeared in Exchange 2010. One of the standard agents, called the scripting agent, is designed to support the automation of common tasks such as mailbox creation. The most common use of the scripting agent is to update the properties of new mailboxes after they are created.

For example, if you create a new mailbox by using EMS or EAC, its language and regional settings are not updated, and the user will be prompted to provide these settings the first time that he accesses the mailbox with Outlook Web App. The scripting agent gives you an easy way to ensure that default language and regional settings are applied to new mailboxes, thus avoiding the need for the user to become involved in the process. If the default settings are not correct, the user can select new values through Outlook Web App options.

The scripting agent is disabled by default. The agent uses an XML configuration file stored in the \V15\Bin\CmdletExtensionAgents folder to understand what processing it must perform and when it is invoked. Exchange provides a sample configuration file called *ScriptingAgentConfig.xml.sample* that you can edit to add your instructions. The sample file contains a number of interesting examples with which you can experiment, but your purposes require only a very simple file that can be created with any text editor and named ScriptingAgentConfig.xml.

The example ScriptingAgentConfig.xml that follows tells the scripting agent that:

- It should be invoked whenever the New-Mailbox or Enable-Mailbox cmdlets are run by any process. These cmdlets are used to create a new mailbox or enable a mailbox for an existing Active Directory account.
- The specified code is invoked when the cmdlets have completed processing (the OnComplete event).
- The Name parameter is to be retrieved from the provisioning handler (the framework that surrounds the scripting agent). The name is the identifier for the object being processed, in this case, a mailbox.
- Three cmdlets are to be run. Set-Mailbox sets a default language value of en-us; Set-MailboxRegionalConfiguration sets the appropriate date and time formats; and Set-MailboxCalendarConfiguration sets the start of the working day. You have encountered all these cmdlets previously in this chapter.

```
<?xml version="1.0" encoding="utf-8" ?>
 <Configuration version="1.0">
 <Feature Name="Mailboxes" Cmdlets="New-Mailbox, Enable-Mailbox">
 <ApiCall Name="OnComplete">
   if($succeeded) {
      $Name= $ProvisioningHandler.UserSpecifiedParameters["Name"]
      Set-Mailbox $Name -Languages "en-US"
      Set-MailboxRegionalConfiguration $Name -DateFormat "dd-MMM-yy" -TimeZone
"Pacific Standard Time"
      Set-MailboxCalendarConfiguration $Name -WorkingHoursStartTime "08:30:00"
      }
 </ApiCall>
 </Feature>
 </Configuration>
```

To enable the scripting agent so that it processes the code in its configuration file, run the Enable-CmdletExtensionAgent cmdlet:

```
Enable-CmdletExtensionAgent "Scripting Agent"
```

This is an organization-wide setting, so it is obviously important to have the same configuration file in place on every Mailbox server so that the same behavior happens throughout the organization. Make sure that you develop and test the configuration file on a test server before introducing it into production because any error in the file might prevent EAC or EMS from completing an operation. After the configuration files are deployed and the scripting agent is enabled, Exchange faithfully executes the specified commands to automate the finalization of mailbox settings.

What's in a mailbox?

After you assign quotas to mailboxes, users start to populate the mailboxes with various items. The Get-MailboxStatistics cmdlet provides a snapshot of what's in a mailbox and what type of items are stored. For example:

```
Get-MailboxStatistics –Identity 'Redmond, Tony' | Select DisplayName, ServerName,
Database, LastLogonTime, ItemCount, DeletedItemCount, AssociatedItemCount,
TotalItemSize, TotalDeletedItemSize
```

Among the interesting information this command reveals is the total items in the mailbox (ItemCount) and the storage required for these items (TotalItemSize). You also have a deleted item count (DeletedItemCount), or the number of recoverable items in the mailbox, and the size of those items (TotalDeletedItemSize).

The associated items (sometimes called folder-associated item [FAI] messages) are hidden items Exchange and Outlook use to store configuration data about the mailbox. For example, if you change a mailbox setting through Outlook Web App, this information is written to a hidden item in the mailbox. The number of associated items varies from mailbox to mailbox and depends on user activity, but it is commonly between 10 and 60 items. The lower end of the range is for mailboxes that are relatively new, and the upper end is typical of mailboxes that have been in use for some time.

You can also generate data for all mailboxes in a database or on a specific server with the Get-MailboxStatistics cmdlet. The second example pipes the output in CSV format to a file that you can open with Excel or Access and use for reporting purposes.

```
Get-MailboxStatistics –Database 'DB1'
Get-MailboxStatistics –Server 'ExServer1' | Export-CSV 'C:\Temp\Mailboxes.CSV'
```

If you use Get-MailboxStatistics with the –Server parameter on a server that hosts database copies from a DAG, you'll see an error for the database copies that tells you they are not mounted or are unavailable. This is because the cmdlet can't open these databases to read mailbox data because the databases are mounted in a special way that allows replication to occur but blocks other access.

Ready-to-go custom attributes

Exchange has always provided administrators with a set of custom attributes by which to store information about mailboxes, contacts, and groups. The logic here is that it is impossible for the designer of any general-purpose directory to include all the attributes required by every company that might use the directory. It's easier to provide a set of custom attributes than to go into the support nightmare that might occur if everyone attempted to extend the schema to add her own set.

To access the custom attributes in Exchange 2013, select an object in EAC, view its properties, and click Custom Attributes. Figure 5-13 shows the set of custom attributes for a mailbox. It's common to use attributes to store employee identifiers, department codes, job codes, identifiers for synchronization with other email directories, and an indicator of whether the mailbox is used by a permanent employee.

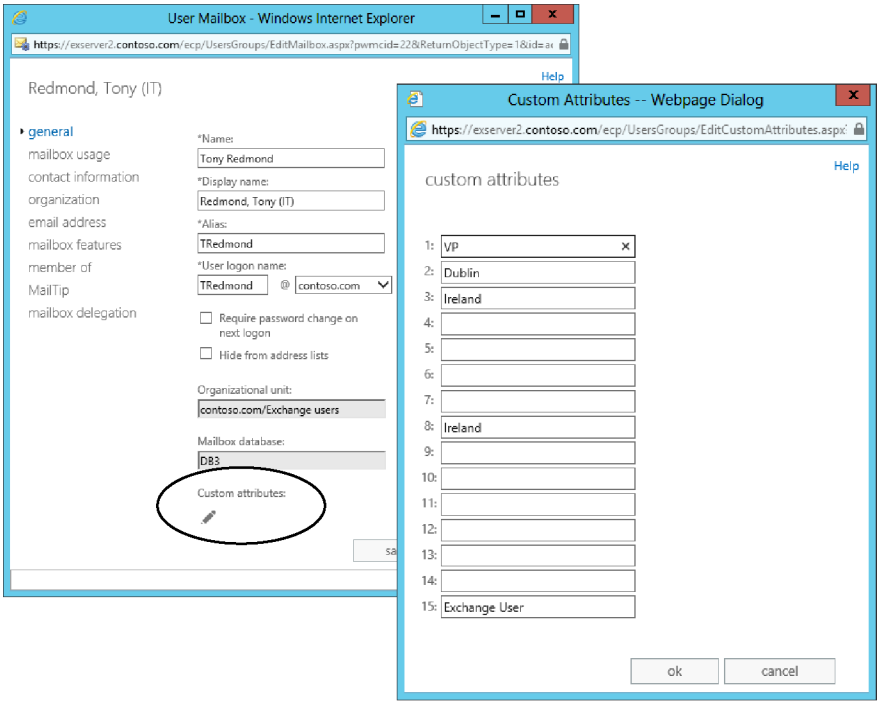


Figure 5-13 Accessing Exchange custom attributes

CAUTION!

Be careful to ensure that the data you hold about people in Active Directory comply with applicable privacy laws in any jurisdiction in which you operate.

The custom attributes are also accessible through EMS. For example:

```
Get-Mailbox -Identity 'EPR' | Select Name, Custom*
Set-Mailbox -Identity 'EPR' -CustomAttribute1 '8ZW' -CustomAttribute12 'Temporary
Assignment'
Set-DistributionGroup -Identity 'Sales' -CustomAttribute10 'Synchronized 10-Dec-2013'
```

```
Set-DynamicDistributionGroup –Identity 'Texas Users' –CustomAttribute3 'Dallas'
Set-MailContact –Identity 'Ruth, Andy' –CustomAttribute1 'Lotus Notes User'
```

Mailbox resources provisioning management agent

Exchange includes a mailbox resources provisioning management agent whose purpose is to assign new mailboxes to available databases. This agent is one of the standard set of cmdlet extension agents. Its purpose is to look for new mailbox creation requests that do not specify a database in which the new mailbox should be created. When it detects such a request, the agent scans for all healthy databases within the same Active Directory site as the server or workstation on which EAC or EMS is running, omitting any databases that are explicitly marked to be excluded from provisioning on a permanent or temporary basis. A healthy database is one that is online and operational. The agent exhibits no great intelligence and selects a database for the new mailbox on a random basis from the set of available databases. If you want to create a mailbox for an account that belongs to a remote site, you should connect to a server on the site that you want to own the mailbox before you create it; else, the mailbox will be created in a database on the local site.

The mailbox resources provisioning management agent does not take the current number of mailboxes in a database into account when it selects a target database. If you allow Exchange to allocate mailboxes to available databases, expect to rebalance load from time to time by moving mailboxes around. You can find out just how many mailboxes are assigned to the different databases with a simple EMS command, albeit one that takes some time to complete in a large organization due to the need to contact every database to count the mailboxes:

```
Get-Mailbox | Group-Object database | Sort Count -Descending | Format-Table Count,
Name –AutoSize
```

Equipped with this information, you can redistribute some mailboxes to the databases that are not heavily populated or suspend some databases from automatic provisioning so that they don't accept new mailboxes. The easiest way to redistribute mailboxes is to create a migration batch of the mailboxes you want to move from one database to another and run the batch over the weekend so that the moves occur at a time of low system load. See Chapter 8, "The Exchange 2013 Store," for more information on how to create and run migration batches.

When you use EAC to create a new mailbox and elect to enable Exchange to select the database to host the new mailbox (Figure 5-14), the only indication that a remote server is involved might be a slight delay when the new mailbox is created over what you'd expect when a mailbox is created locally.

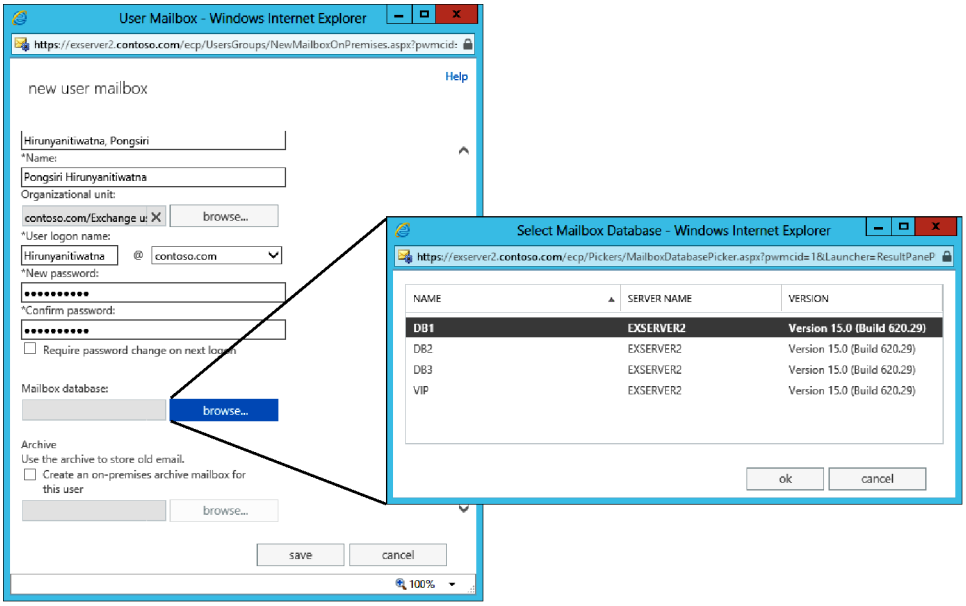


Figure 5-14 Specifying a database for a new mailbox

Three flags are available to help you control the databases that are available for automatic provisioning. These are as follows:

- IsExcludedFromInitialProvisioning means that a database should be disregarded for provisioning on a very temporary basis. Setting this flag does not prevent Exchange from using the database to hold new mailboxes. It is strictly advisory. This property is new in Exchange 2013.

- IsSuspendedFromProvisioning also marks a database that is excluded from provisioning temporarily. However, if you set this property to $True, Exchange ignores the database when it creates new mailboxes.

- IsExcludedFromProvisioning means that a database is disregarded for provisioning on a more permanent basis. The intention is for you to use IsSuspendedFromProvisioning if you just want to stop new mailboxes from being created in a database for a reasonably small amount of time, say, up to half a day. Use IsExcludedFromProvisioning for longer periods. The difference between the two properties enables administrators to configure some granularity in the provisioning status of a database.

Exchange sets a similar property, called IsExcludedFromProvisioningBySpaceMonitoring, if the Managed Availability system detects that space is running out on the disk used to hold a database. You cannot set this property.

If either the IsExcludedFromProvisioning or IsSuspendedFromProvisioning flag is set to $True, the database is excluded from provisioning and remains in that state until both flags are reset to $False. Both flags have the same practical effect of stopping Exchange from using a database for provisioning. The difference between them is how long the provisioning block exists. Excluded means "never use this database for provisioning"; suspended means "don't use this database for provisioning until I tell you it's okay to do so." The following explores how these circumstances might arise.

You might want to exclude a database from mailbox provisioning permanently if it is reserved for a particular purpose. For example, you might want to assign a database that has elevated quotas to a certain class of user or to users who belong to a particular department. You might also decide that automatic mailbox provisioning is a bad idea and that you'd prefer to assign new mailboxes to specific databases manually. In either case, you can use a command such as this to exclude the database from provisioning:

```
Set-MailboxDatabase –Identity 'VIP Mailboxes' –IsExcludedFromProvisioning $True
```

A temporary exclusion might be invoked when you know that a database should not be assigned new mailboxes for a period. For example, the database might be due to be moved to a new disk that allows it to grow, and you don't want to add any more mailboxes until the new storage is available. In this case, you can block provisioning temporarily with a command like this:

```
Set-MailboxDatabase –Identity 'DB1' –IsSuspendedFromProvisioning $True
```

When you are ready to allow the database to be assigned new mailboxes on an automatic basis, you set the flag back to $False.

```
Set-MailboxDatabase –Identity 'DB1' –IsSuspendedFromProvisioning $False
```

Unlike many other properties that are stored in Active Directory, a change to a flag to suspend or include a database in automatic provisioning is immediately respected by the mailbox provisioning agent.

By default, new mailbox databases are added to automatic mailbox provisioning after they are created. Databases that are made available for automatic provisioning also become candidates for selection as targets for mailbox moves and personal archives if the locations for these operations are not specified by administrators. In other words, if you don't specify a target database when you move a mailbox or create a personal archive for a mailbox, the mailbox provisioning agent will select a target from a list of available databases. My tests show that Exchange usually selects the same database as the primary mailbox to hold an archive, even if you don't select a specific target database. I also found that if you issue a move request for a mailbox that has an archive, Mailbox Replication Service (MRS) will move the primary and archive mailboxes together to a target database that it automatically selects.

TROUBLESHOOTING

A valid mailbox wasn't found and mailbox creation failed

If you create new databases from EMS, you can specify that the database is excluded on a permanent or temporary basis by setting the flags previously described when you run the New-MailboxDatabase cmdlet. You can go too far and exclude all available databases from automatic provisioning. In this situation, you see the error shown in Figure 5-15 when the mailbox provisioning agent scans for databases and discovers that none are available.

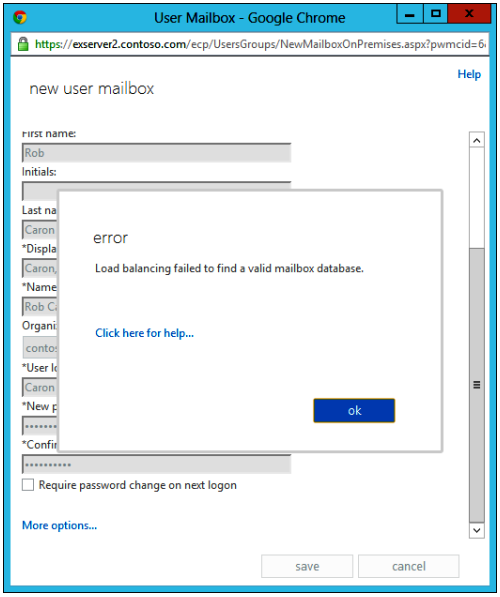


Figure 5-15 No databases available for load balancing

The fix is simple. Either reset the flag on a database to make it available or go back and select a different database. An explicit selection of a database tells the mailbox provisioning agent that it doesn't have to make a selection, so the error will be avoided.

The problem with using automatic assignment of mailboxes to databases rather than asking administrators to think about which database should host a new mailbox is that it ignores any consideration of with whom the owner of the new mailbox is likely to correspond. For years, it has been best practice to keep the mailboxes for people who share a common workgroup together in the same database to minimize the use of system resources to transport messages among members of the workgroup. Advances in technology have reduced the importance of this principle. The current Exchange database schema eliminated single-instance storage within databases, and Exchange 2007 began forcing all messages to move

through the transport system, even when they are addressed to a recipient in the same database. Even so, it's still good to consider mailbox placement from a wider perspective than a simple distribution across available mailboxes and retain some administrator involvement in the process instead of leaving everything for a computer to decide.

Setting mailbox quotas

Mailbox quotas vary enormously from deployment to deployment and should reflect the need users have for email to support their work. A factory worker might need a quota of only 50 MB to receive workplace notices and announcements, whereas an executive in an information-rich company could require a quota of 50 GB or more. In some respects, any discussion about setting mailbox quotas is often academic because administrators do not pay as much attention to this topic as they did in the past. This is for a few reasons.

Storage is much cheaper today than ever before. Therefore, it is has become feasible to provide large mailboxes on a very economical basis. As storage prices continue to drop, the tendency to contemplate larger mailbox sizes will also continue. Microsoft is looking at how servers and clients can manage 100-GB mailboxes, which is a huge amount of information when compared to the meager 25-MB mailboxes that were commonplace when Exchange 4.0 was launched in 1996.

Users expect that they will have large mailboxes at work, largely because consumer email systems continually increase the space they allow for free mailboxes. The question often asked is, if Gmail or Outlook.com can offer a huge amount of storage for email, why can't my company's email system do the same?

On a commercial level, the cloud-based versions of Exchange usually provide mailboxes of at least 25 GB, further confirming the user view that they should have the same storage available when at work. A proliferation of free storage repositories such as SkyDrive exist to offer users even more space.

A major engineering focus for Exchange over the last two versions has been how to enable users to keep as much information as possible in the Store rather than forcing users to keep moving data into PSTs, as they were forced to do in the past. Keeping everything in the Store is easier for users and makes it feasible for companies to operate a comprehensive compliance regime. Features such as archive mailboxes increase available storage even more, and retention policies allow unwanted information to be aged out and removed from mailboxes automatically.

Given the pressure for workers to be as productive as possible, it makes little sense to force users to spend time (sometimes on a daily basis) cleaning up mailboxes before they can process email. Time spent reviewing and deleting messages from a mailbox would be better spent dealing with more mission-critical business.

The result of these influences is that it is now common for administrators to set a reasonably large default quota and not worry too much more about the issue until users look for additional quota. Apart from letting administrators get on with more important work, the fact is that it takes even prolific users some time to fill a multi-gigabyte mailbox. Another approach often seen is that mailbox quota defaults are immediately set to higher values (usually 5 GB or higher) after a database is created so that any new mailbox created in the database receives a quota that should not need adjustment for a couple of years.

Even with these factors, the vast majority of users in large corporations probably still have a mailbox quota of less than 1 GB. The average mailbox size has been rising slowly. If you work for a company that still likes to exert some level of control over mailbox quotas, if only to calculate the storage requirements for databases with some degree of accuracy, you can set three mailbox properties that combine to control the quota available to the mailbox.

- **IssueWarningQuota** This value determines when Exchange starts to issue warning messages to the user that he is approaching his mailbox quota. Given that mailbox quotas vary so much, it is difficult to offer suggestions about how much short of when the user can no longer send mail this value should be set. Because you want users to be able to react to the warning they receive, it's a good idea to set the value to allow several days to elapse before the shutters come down and the user cannot work. For example, if user mailboxes typically grow by 20 MB each day during the working week, setting the warning point at 100 MB before Exchange stops the user from sending any new messages allows the user up to a complete working week to take the necessary action to clean out her mailbox or request additional quota.

- **ProhibitSendQuota** This value determines when Exchange stops the user from sending any new messages. When users refer to their mailbox quota, they usually refer to this value because it marks the point when they have to take action before they can send email again.

- **ProhibitSendReceiveQuota** This value determines the point when Exchange refuses to accept new messages in the mailbox. Messages arriving past this point are returned to the originator with an error indicating that the target mailbox is full. Typically, this value is placed between 100 MB and 200 MB higher than the ProhibitSendQuota value to account for situations such as vacations, when users might inadvertently leave their mailboxes almost full. The buffer established by the difference between the ProhibitSendQuota and ProhibitSendReceiveQuota values allows the mailbox to continue to accept new messages until users return and can delete old items from their mailboxes and restore full operation.

Exchange sends messages to warn users of approaching quota thresholds or to tell them that they have exceeded their mailbox quota according to a schedule maintained in the

QuotaNotificationSchedule property that is set on every mailbox database. You can see the schedule for a database by running a command like this:

```
Get-MailboxDatabase -Identity 'DB2' | Format-List Name, QuotaNotificationSchedule
```

When a new mailbox is created, it inherits its quota settings from those assigned to the database in which it is created. For example, any mailbox created in the database shown in Figure 5-16 that is not assigned an individual set of quotas begins receiving warning messages at 1.9 GB, stops being able to send new messages at 2 GB (the mailbox quota), and cannot receive any more new mail at 2.3 GB. These are the default values Exchange 2013 assigns to new mailbox databases.

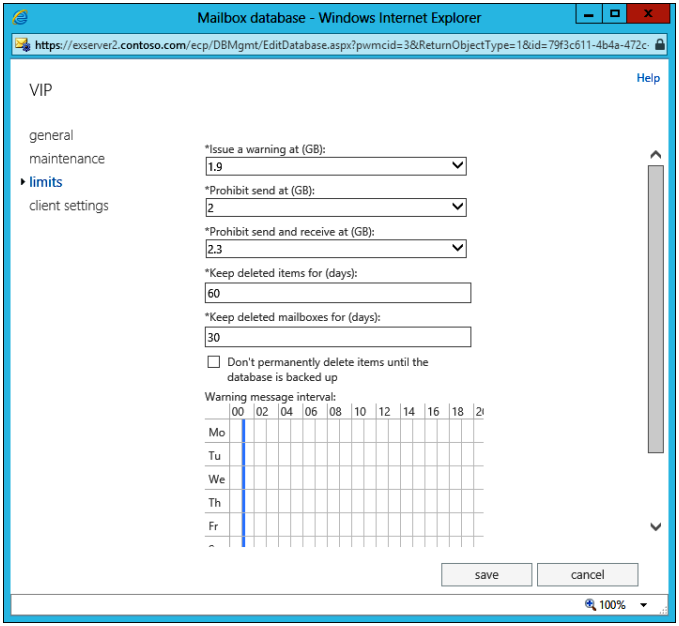


Figure 5-16 Viewing the default storage quotas for a mailbox database

Mailboxes that use the quotas inherited from the database settings have their UseDatabaseQuotaDefaults property set to $False, whereas those that have individual quota settings have the property set to $True. It's therefore easy to find out which mailboxes in a database have been assigned individual quotas with a command like this:

```
Get-Mailbox -Database DB2 | Where-Object {$_.UseDatabaseQuotaDefaults
-eq $False} | Format-Table DisplayName, IssueWarningQuota, ProhibitSendQuota,
ProhibitSendReceiveQuota
```

If the default mailbox quotas Exchange 2013 sets don't work for your installation, you need to update the properties of mailbox databases to set whatever values you want to use.

After updating the databases, you might need to review mailboxes that already exist in each database to determine whether their quotas need to be adjusted.

At any time after they are created, individual mailboxes can be assigned specific quotas to meet their business needs. To update quotas with EAC, select the mailbox, view its properties, choose Mailbox Usage, and click More to reveal the fields in which you can input the new quota settings (Figure 5-17). Click Customize The Settings For This Mailbox to make the three quota fields active. Note that EAC does not allow you to enter values using more than two decimal places.

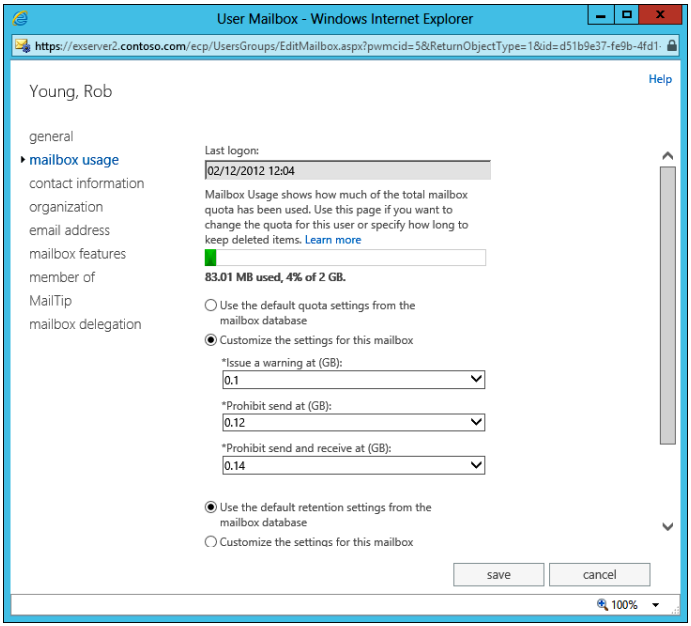


Figure 5-17 Setting quotas for an individual mailbox

You can also use the Set-Mailbox cmdlet to assign new quota values to mailboxes. For example:

```
Set-Mailbox -Identity 'Redmond, Tony' -UseDatabaseQuotaDefaults $False
-IssueWarningQuota 500MB -ProhibitSendQuota 520MB -ProhibitSendReceiveQuota 540MB
```

Quota values can be expressed in KB, MB, or GB. A one-line EMS command is a great way to set the same quota on a group of mailboxes. For example, you could set the same quota values for every mailbox in an OU:

```
Get-Mailbox -OrganizationalUnit 'contoso.com/Exchange Users' | Set-Mailbox
-UseDatabaseQuotaDefaults $False -IssueWarningQuota 500MB
-ProhibitSendQuota 520MB -ProhibitSendReceiveQuota 540MB
```

You can use the same approach to assign quotas to every member of a distribution group. In this case, the command to give members of the IT department the quotas they deserve is:

```
Get-DistributionGroupMember -Identity 'IT Department' | Set-Mailbox
-UseDatabaseQuotaDefaults $False -IssueWarningQuota 985MB
-ProhibitSendQuota 1GB -ProhibitSendReceiveQuota 1.1GB
```

To check that the right quotas are in place, you can do something like this:

```
Get-Mailbox -Identity 'Redmond, Tony' | Select *quota*
```

Exchange maintains several other quota settings for mailboxes. The RecoverableItemsQuota and RecoverableItemsWarningQuota settings are roughly equivalent to the ProhibitSendQuota and IssueWarningQuota settings except that they control the amount of data that a mailbox can store in the Recoverable Items folder. By default, Exchange 2013 allows 30 GB of recoverable data to be held in a mailbox. The RulesQuota setting limits the amount of rules data a mailbox can hold, and 64 KB is the default. The ArchiveQuota and ArchiveWarningQuota settings control how much information can be held in an associated personal archive if this feature is enabled. Exchange allows an unlimited quota for the archive, so in this case, the quota has been set back to a hard limit of 50 GB (still more than enough).

New quotas are not immediately effective because the Store first has to refresh the cached data it holds about mailbox settings. The cache is refreshed every two hours, so you could have to wait this long before the new quota settings are in place. You can update the Reread Logon Quotas Interval registry setting to force more frequent refreshes of this data at the expense of additional system overhead. (Refreshing the data every five minutes would definitely be a bad idea; every hour or so should be acceptable.) The DWORD value is set in seconds. (The default value is 7,200 seconds, or two hours.) This value depends on other values that control how often Exchange retrieves information from Active Directory. Another option is to restart the Information Store process to force a cache refresh. This approach is acceptable on a test server but probably not on a production server.

Details of how to alter the values in the Reread Logon Quotas Interval registry setting can be found at *http://technet.microsoft.com/en-us/library/bb684892(EXCHG.80).aspx*.

Estimating database size

From a planning perspective, it's important to realize that the simple age-old calculation of

```
(number of mailboxes * quota) = database size on disk
```

is highly unlikely to be accurate. There are other factors to take into account to deter mine the likely size of a database.

- **A quota is a maximum value allowed to a mailbox. Unless the quota is very small, users will take some time to fill their quota. A 2-GB quota might take a user two years or more to fill.**
- **Recoverable items require further space in the database. If you allow users to retain deleted items for 21 days, you'll use up to 10 percent additional space. This storage is governed by a separate quota but is not charged to the user's quota.**
- **Personal archives will occupy further space if they are stored in the same database as their primary mailbox (even if they are not, primary mailboxes from one database will be stored alongside mailboxes in another).**
- **White space will exist within the database. These are database pages that were previously used to store items that have been deleted and are available for reuse by Exchange to store new items and attachments. Exchange is very efficient at reusing deleted pages, but even so, the nature of email is that new items arrive on an ongoing basis, some of which are deleted and some of which are kept. The flow of creates and deletes within the database creates a churn of pages, so you can estimate that another 10 percent of mailbox sizes might be occupied by white space.**

A better calculation is to estimate the maximum predicted size to which a database might grow to place the database on a disk that has sufficient space. By adding 20 percent to the previous calculation, you can size the database to be:

```
(number of mailboxes * quota) * 1.2 = database size on disk
```

Don't forget to reserve some additional disk space to accommodate expected database growth over the next year or so and to account for temporary space required by activities such as mailbox moves or offline database maintenance.

Removing or disabling mailboxes

Mailboxes don't last forever, and eventually you will want to remove some from Exchange, usually after users leave the company. You might want to keep a mailbox for some time after a user leaves so that its contents are not immediately lost and remain accessible to other users or even for legal purposes. EAC provides two options for mailbox removal: you can either disable or remove a mailbox. The words seem similar, but there's a huge difference in what happens behind the scenes.

When you disable a mailbox, Exchange removes all the properties from the underlying Windows user account in Active Directory that associates the user with the mailbox. The contents of the mailbox are purged from its database after the deleted mailbox retention

period expires. When Exchange purges a deleted mailbox from a database, the only way you can retrieve the mailbox afterward is to retrieve it from a backup.

The equivalent EMS command to disable a mailbox is:

```
Disable-Mailbox –Identity 'Redmond, Tony'
```

When you remove a mailbox, in addition to marking the mailbox for purging after its retention period expires, you also remove the Windows user account from Active Directory. The EMS command EAC uses is:

```
Remove-Mailbox –Identity 'Redmond, Tony'
```

As with disabled mailboxes, Exchange retains the content of removed mailboxes in their original databases until their retention period expires. If you want, you can force Exchange to remove the Active Directory user account and mailbox contents immediately by running the Remove-StoreMailbox cmdlet. For example:

```
Remove-StoreMailbox –Identity 'Redmond, Tony'
```

Before disabling or removing a mailbox, EAC warns about the consequences of taking the action. Although the result seems dire, unless you opt to remove a mailbox permanently, you can always reconnect a mailbox to an Active Directory account as long as the mailbox is available in the database. If the mailbox has a personal archive, you are told about this in the warning message.

Another factor to consider is how to deal with mailboxes that have been placed on an in-place hold (or the older litigation hold feature; see Chapter 10 for more information). In these instances, a reason exists for the mailbox contents to be retained, and if you delete the mailbox, you might create a problem for the person or persons who requested the mailbox to be put on hold. Therefore, if a request comes in to remove a mailbox that is on hold, you should always check with your legal department, or whatever competent authority exists to authorize ignoring that a hold is in place, before you process the removal. EAC won't allow you to remove or disable a mailbox if a hold is in place.

An in-place hold might be in effect for multiple mailboxes, including the one you want to process, so you probably don't want to remove the hold from all the targeted mailboxes. In this case, you can disable or remove the mailbox without affecting the other mailboxes with EMS as follows:

To disable the mailbox:

```
Disable-Mailbox –Identity 'Cao, Jun' –IgnoreLegalHold
```

To remove the mailbox permanently:

```
Remove-StoreMailbox –Identity 'Cao, Jun'
```

In both events, EMS prompts you to confirm the action before it proceeds.

INSIDE OUT **Remove-Mailbox and Remove-StoreMailbox cmdlets**

The Remove-Mailbox and Remove-StoreMailbox cmdlets can both be used to remove a mailbox from a database permanently. The difference is as follows:

- **Use Remove-Mailbox to remove the Active Directory account for the user and mark the mailbox for deletion after its retention period expires.**
- **Use Remove-StoreMailbox when you want to remove the mailbox from its database immediately. In effect, this new cmdlet replaces the Remove-Mailbox cmdlet when it is used with the Permanent and StoreMailboxIdentity parameters.**

Why have two cmdlets to remove a mailbox permanently? The answer is that it forces administrators to make a distinct choice to remove a mailbox permanently. With Remove-Mailbox, it's easy for an administrator to overlook using the Permanent parameter and thus have mailboxes lingering in the database until their retention period expires. The use of Remove-StoreMailbox should be very easy to understand. It also makes the task of auditing easier because if the need arises, an auditor can just search for all uses of Remove-StoreMailbox rather than going through each instance of Remove-Mailbox to determine whether the Permanent parameter was used. More information about how to audit administrator actions is available in Chapter 11.

Reconnecting mailboxes

When Exchange disconnects a mailbox after running either the Disable-Mailbox or Remove-Mailbox cmdlet, it stamps the current date and time in the mailbox's DisconnectDate property. This marks the start of the clock that ticks down until the retention time expires. The retention time for deleted mailboxes is a property of a mailbox database and can differ from the default 30 days. For example, for legal reasons, you might want to remove or retain mailboxes for some classes of employees for different periods. This is easily achieved by placing the employee mailboxes in databases that are configured with different retention periods. You can use this command to see the current deleted mailbox retention period for all the databases in the organization:

```
Get-MailboxDatabase | Select Name, MailboxRetention
```

Setting a different retention period is a matter of using Set-MailboxDatabase to set the desired value. For example, to set a mailbox retention period of 60 days for the VIP Data database, you would use this command:

```
Set-MailboxDatabase -Identity 'VIP Data' -MailboxRetention 60
```

EAC includes the Connect Mailbox option (in the list revealed by the ellipsis) to view disconnected mailboxes and then connect a selected mailbox back to an Active Directory account (Figure 5-18). This option works by scanning all the mailbox databases on a selected server to detect disconnected mailboxes. The administrator can then select a mailbox and click the Connect icon (above Display Name in the figure) to force EAC to search for a suitable Active Directory account to which it can connect the mailbox. If EAC can find a suitable match in Active Directory, it reconnects the mailbox. If not, EAC displays a dialog box to enable the administrator to select the correct Active Directory account to which the mailbox should be reconnected. It is possible that no suitable Active Directory accounts can be found, as when all available accounts are already connected to a mailbox. If this happens, you can create a new Active Directory account and then rerun the option. EAC detects the new account and enables you to reconnect the mailbox to it.

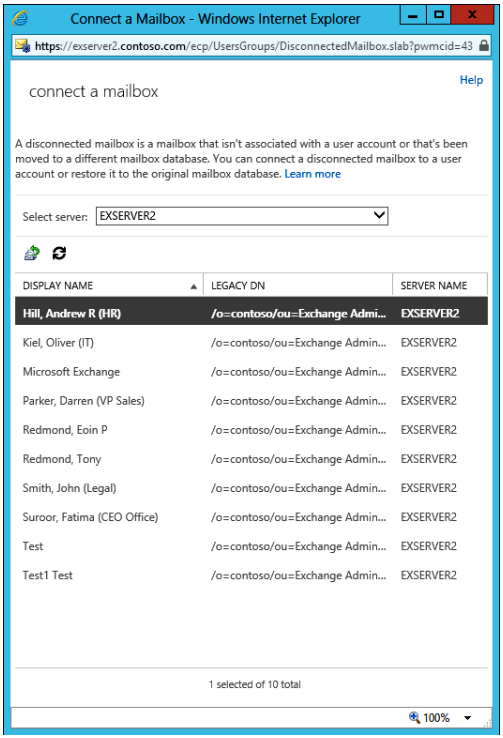


Figure 5-18 Viewing disconnected mailboxes

It's more fun to do the work through EMS because you learn more about Exchange by exploring the different steps that are involved in the process. First, you have to execute a

command like this to retrieve information from a database about the mailboxes that are in this state:

```
Get-MailboxStatistics -Database DB2 -Filter {DisconnectDate -ne $Null} | Format-List
DisplayName, DisconnectDate, DisconnectReason
```

If you want, you can execute much the same command to discover details about all disconnected mailboxes on a server:

```
Get-MailboxStatistics -Server ExServer2 -Filter {DisconnectDate -ne $Null} |
Format-List DisplayName, DisconnectDate, DisconnectReason, Database
```

You see values such as SoftDeleted and Disabled reported as the reason a mailbox was disconnected. Mailboxes shown as SoftDeleted are those that have been moved to another database. They are retained in the original database just in case the mailbox move is affected by a problem that leaves the new mailbox in an inconsistent or corrupted state for some reason. Keeping the original mailbox following a move enables an administrator to reconnect the user to this copy if the need arises.

Mailboxes that are reported as Disabled are those that have been disabled through EAC or by running the Disable-Mailbox command in EMS. Sometimes it can take a little while after a mailbox is disabled or removed before it shows up on the list of disconnected mailboxes. This could occur because Active Directory hasn't replicated the disconnected status for the mailbox to the domain controller EMS is using. It might also be that the Store has not yet stamped a disconnected date on the mailbox, so the filter EMS uses to find disconnected mailboxes doesn't pick it up.

After you have identified a disconnected mailbox that needs to be reconnected, you can use the Connect-Mailbox cmdlet to do the job. The easiest situation is when Exchange can match the disconnected mailbox to an Active Directory user account by reference to the name of the Active Directory account by using the values contained in the DisplayName and LegacyExchangeDN properties of the disconnected mailbox. For example, if you have an account called John Smith and a disconnected mailbox whose display name is also John Smith, it's likely that the two are a match; it's therefore acceptable to let Exchange join the two:

```
Connect-Mailbox -Database VIP -Identity "John Smith"
```

Life isn't normally so straightforward. At least, not all the time. Different naming schemes for Active Directory user accounts and Exchange mailboxes can conspire to stop any attempt to match one with the other. EMS flags an error if it can't run Connect-Mailbox as previously shown and find a matching Active Directory account. In this case, you can always use the globally unique identifier (GUID) to identify the mailbox in the database and direct Exchange to the precise Active Directory account to which you want to connect the mailbox.

First, retrieve the GUID:

```
$Guid = (Get-MailboxStatistics –Database DB2 –Filter {DisconnectedDate –ne $null
–and DisplayName –eq 'John Smith'}.MailboxGuid
```

This returns a value such as '50e2778f-e8ae-40d7-9dd8-bb22a101e8e5'. You can now use that value to reconnect the mailbox:

```
Connect-Mailbox -Identity '50e2778f-e8ae-40d7-9dd8-bb22a101e8e5' -Database 'DB2'
-User 'contoso.com/Exchange Users/John Smith' -Alias 'JSmith'
```

The mailbox GUID is the most precise method of identifying an Exchange object in situations like this. It also is useful if you want to delete a mailbox from a database before its retention period expires. This code scans a database for a particular mailbox, saves its GUID in a variable, and then removes the contents by using the Remove-Mailbox cmdlet. Note the use of the StoreMailboxIdentity parameter in this command.

```
$Mbx = Get-MailboxStatistics –Database 'DB1' | Where {$_.DisplayName –eq 'Redmond,
Tony'}
Remove-Mailbox –Database 'DB1' –StoreMailboxIdentity $mbx.MailboxGuid
```

When you delete mailbox contents in this way, you will have to use a backup to retrieve them if necessary afterward.

INSIDE OUT Avoiding errors

You might be concerned about the potential for error that can occur if an administrator uses Remove-Mailbox instead of disabling a mailbox and discovers that he's just deleted the user's Active Directory account. To avoid problems and to make sure that mailboxes are kept as long as they are required, many companies have adopted a simple process by which they do the following:

1. **Disable the Windows account to prevent any further access to the account.**
2. **Hide the mailbox from the GAL.**
3. **Change the SMTP email address to make it invalid and stop new messages from arriving in the mailbox.**
4. **Set up a suitable auto-reply message for the mailbox so that anyone who sends to the new SMTP address is told that her message won't be read.**
5. **Keep the mailbox online and accessible for up to 90 days.**

Between 45 and 60 days is usually sufficient to establish whether the mailbox contents are ever required. When the mailbox is no longer required, you can safely eliminate both the account and mailbox.

Discovery mailboxes

Discovery mailboxes are used as the repository for the metadata that drives eDiscovery searches and the output the searches generate from user mailboxes. How eDiscovery searches are executed is discussed in Chapter 11. If not present, two discovery mailboxes are created by the Exchange installation procedure. The first is the discovery metadata mailbox that holds information about completed and pending searches. You cannot perform eDiscovery searches unless this mailbox is online and available. It has a fixed name of SystemMailbox{e0dc1c29-89c3-4034-b678-e6c29d823ed9}. This mailbox is created as an arbitration mailbox, and you can locate it with:

```
Get-Mailbox –Arbitration
```

The second type of arbitration mailbox is a discovery search mailbox, which stores the items copied as a result of eDiscovery searches for later access and review by users who have been granted permission to open the mailbox. A single discovery search mailbox is created by the installation procedure and should be accessible by users who are members of the Discovery Management role group. As such, these users should be able to open the default Discovery Mailbox and peruse its contents. If you create additional discovery mailboxes, you must assign Full Access to the accounts that will use these mailboxes for mailbox searches if you want to open the mailboxes.

As explained below, you can create other discovery mailboxes as required. To stop users from attempting to send them email, new discovery mailboxes are automatically hidden so that they don't appear in address lists. If someone attempts to send a message to a discovery mailbox by using its SMTP address, Exchange will reject the message and return an NDR to inform the user that "delivery to this address is restricted".

You can locate all the discovery mailboxes that exist in the organization with EMS by using the following command. Knowing which database the mailboxes are in is important because they will have to process many transactions if items are copied following an eDiscovery search.

```
Get-Mailbox –RecipientTypeDetails DiscoveryMailbox | Format-Table Name, Database
```

The Exchange installation procedure creates the default arbitration mailboxes in the Users OU of the root domain. For this reason, unless you are logged on to the root domain, you might have to establish the correct Active Directory scope to find these mailboxes.

Creating additional discovery mailboxes

Exchange creates the default discovery mailbox in the mailbox database of the first Exchange Mailbox server you deploy. This is an acceptable configuration for small

deployments but might prove problematic for larger organizations, where the sheer volume of data uncovered by an eDiscovery search could be very large in terms of the number of items and the size of the storage required.

Storing the results of eDiscovery searches should not be an issue because the default quota assigned to the discovery mailbox is 50 GB. However, remember that the Mailbox server that holds the database containing the discovery mailbox has to do a lot of work to copy items a search unearths. For example, if a search locates 10,000 items that occupy 6 GB, the server has to be able to accept the workload to copy and store these items. The workload is composed of the CPU consumed during the search, the storage for the discovered items, and the transaction logs generated as the discovered items are created in the discovery mailbox. A search might be performed several times before the final information is captured, and each time, the server will be stressed, so think about the following:

- The number of discovery mailboxes that are created and available within the organization. One will suffice for small organizations, but perhaps it is better to create a number of discovery mailboxes on different servers for use by the teams that perform searches.
- The location of the discovery mailboxes. The ideal situation is that the mailboxes being searched, the users who perform the search, and the database hosting the target discovery mailbox should be on the same site; this eliminates any need for extended network connections to search, store, and review information. In any case, you need to consider whether the server hosting the database that contains the discovery mailbox has sufficient capacity to handle the load generated by searches.

Users in the Discovery Management role group can perform searches. This topic is covered in detail in Chapter 11. Part of creating a new search request is the selection of the discovery mailbox that holds the result of the search. The need to hold potentially huge amounts of data uncovered by searches is why discovery mailboxes are assigned a 50-GB storage quota. When data are captured by a search, users have to be granted Full Access to the discovery mailbox if they want to open the mailbox and access the search results. See the "Managing Full Access permission" section later in this chapter for information about how to grant Full Access to a mailbox.

You can create additional discovery mailboxes by using the New-Mailbox cmdlet with the Discovery switch. For example:

```
New-Mailbox 'Legal Action Discovery Mailbox' -UserPrincipalName 'LegalDiscovery@
contoso.com' -Discovery
```

Tip

After you create a new discovery mailbox, make sure that you assign Full Access permission to the mailbox to the groups that need to access the search results it stores. By default, the default discovery mailbox can be opened by the Administrator account, but permission to access this mailbox also needs to be granted to anyone who needs to access it. The "Managing Full Access permission" section later in this chapter discusses how to set Full Access permission on a mailbox.

If you attempt to delete a mailbox database that holds discovery mailboxes, Exchange reports an error, and you will need to move these mailboxes before you can delete the database. Although EAC does not display discovery search mailboxes in its mailbox list, it does in the dialog box in which you select mailboxes to include in a migration batch, so you can use EAC to move any discovery search mailboxes that exist in a database. However, EAC does not allow you to include the discovery arbitration mailbox in a migration batch, so if you need to move it to another database, you have to do this through EMS. For instance, this command moves the discovery arbitration mailbox to the DB1 database:

```
New-MoveRequest -Identity 'SystemMailbox{e0dc1c29-89c3-4034-b678-e6c29d823ed9}'
-TargetDatabase 'DB1'
```

Health mailboxes

Exchange 2013 includes a new Managed Availability feature that is incorporated in the system architecture so Exchange can detect and resolve common problems caused by malfunctioning components. Managed Availability runs on every Exchange 2013 server, where you see it represented as the Health Manager Service process (MSExchangeHMHost.exe) and the Health Manager Worker process (MSExchangeHMWorker.exe).

The basic idea behind Managed Availability is to deploy an extensive set of intelligent probes into the array of services that comprise Exchange. The probes can measure current activity against a norm, defined as the expected state of a service in a healthy state. The data fed back by the probes are assessed by a monitoring engine that compares what is currently happening against the norm. If a difference is detected, the monitoring engine determines whether the difference is sufficient to warrant intervention. In effect, the monitoring engine acts like a super-efficient system administrator who is constantly checking what's going on across every Exchange service to consider whether the server is healthy. Unlike human administrators, Managed Availability functions 24 hours a day, 7 days a week and never takes a break to refresh itself with coffee or any of the other brews favored by Exchange administrators.

If a condition arises that the monitoring engine considers problematic, it alerts a responder engine, which is equipped with knowledge of a range of actions that can be taken to resolve problems. Again, this can be compared to when a human administrator notices that something is not quite right when he examines some aspect of a server and then decides to do something to restore the situation to one comparable to what he would expect to see if everything is functioning normally. For example, the IIS application pool that services inbound Outlook Web App requests might not be responding. This is difficult for a human administrator to detect because she doesn't usually check this aspect of a server unless a user reports a problem connecting through Outlook Web App. However, it's relatively simple for a computer probe to monitor and then report. In this case, the action taken by the responder engine might be multistage, similar to the way a human might try one approach to fix a problem and, if that doesn't work, then try another. The responder engine can restart the application pool and then test to see whether that attempt worked by making an artificial Outlook Web App connection. If the connection succeeds, all is well, and the problem is resolved. If not, the responder engine escalates its intervention, restarts the underlying service, and again tests to measure the success of the step taken. At this point, the responder engine can do little if its intervention has not restored the server to full health. It could back off and signal a high-priority alert to make a human aware of the issue and seek his help, but all available humans might be in bed or otherwise unavailable. Therefore, the responder engine might proceed to the next step, which is to force a system restart in the hope that this resolves the issue. (Many experienced system administrators will immediately recognize the value of restarting a computer in an attempt to resolve unresponsive problems.)

Managed Availability is undoubtedly in its early days, and the hope is that it will improve and evolve in terms of sophistication and capability over time. It's worth noting at this point why Microsoft has incorporated such a facility. Briefly, it's because it has found that it is extraordinarily helpful to build as much automation as possible into servers that are deployed in massive online services such as Office 365. Human intervention is expensive, takes too much time, and is prone to error, whereas computers are very good at following well-defined steps to resolve well-understood problem conditions.

Synthetic transactions are a good way of measuring that everything is working properly in any transaction-based system. Even though it is an email server, you can consider the messages Exchange processes to be transactions. Therefore, it makes sense for Exchange to generate messages and use them to measure whether everything along the path of those messages handles them properly. To mimic the work human users do, the messages have to originate from somewhere and be sent to somewhere else, and that's where health mailboxes are used. Two health mailboxes are created (with archives) in every mailbox database as soon as the first mailbox is created in the database. The Health service will recreate any mailboxes that are missing, so if you remove the health mailboxes, they will reappear the next time the Health service restarts.

The health mailboxes are associated with user accounts created in the Users OU in Active Directory (Figure 5-19). You can also retrieve information about the health mailboxes with the Get-Mailbox –Monitoring command; an examination of their properties reveals that health mailboxes have their RecipientTypeDetails property set to MonitoringMailbox. A useful one-liner is the command to report on how much space is occupied by the health mailboxes:

```
Get-Mailbox –Monitoring | Get-MailboxStatistics | Format-Table DisplayName,
TotalItemSize, ItemCount
```

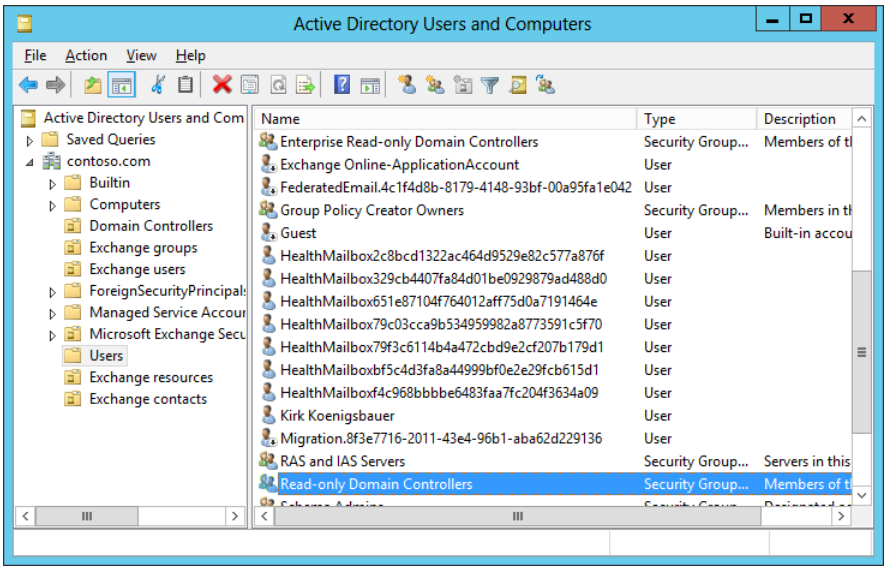


Figure 5-19 Health mailboxes in Active Directory

Exchange uses the health mailboxes to establish that email connectivity exists to the various databases in the system by sending artificial messages to and from the mailboxes every five minutes or so. This results in a number of observable side effects, including:

- That the health mailboxes are not empty and will report that they store some information if you examine them with Get-MailboxStatistics. This is not an issue because the amount of data is relatively small. If you spot that a health mailbox stores more than 100 MB, you should try to determine why this is so.

- That the transactions for health mailboxes contribute to a certain increase in transaction logs and replication between database copies in a DAG. Again, the overall increase and impact is very slight. In fact, the transactions generated by the health mailboxes help keep log replication ticking over because databases are never left without a transaction for very long.

- That the messages sent between health mailboxes are recorded in message-tracking logs.
- That the messages sent between health mailboxes are journaled if you do not exclude them from your journaling rules. One way of doing this is to mark the health mailboxes by setting a known value into one of the 15 customized attributes available for mailboxes and then excluding any messages generated by a mailbox with that value set.

At this point, you are still learning about the operational considerations you must take into account for both Managed Availability and health mailboxes. A review of the current knowledge on the topic as expressed in blogs and other Internet sources will be useful in understanding how to factor these elements into your deployment.

Setting mailbox permissions

The ability to change the way messages are processed in a mailbox became a fundamental part of email system functionality a long time ago. Exchange enables you to assign different levels of control over mailboxes to influence the way messages are delivered to the mailbox, who has the right to send messages from or on behalf of the mailbox (there is a difference), or who has complete control over the mailbox. These assignments can be made by changing mailbox properties through Mail Flow Settings or by assigning different Active Directory permissions to allow other accounts to impersonate or access contents in the mailbox.

Mailbox delegation

EAC provides three settings under the Mailbox Delegation section of a mailbox's properties (Figure 5-20):

- **Grant Send As permission** You can assign the ability to a user to send messages as if they were generated by another user. This is often referred to as the impersonation feature.
- **Grant Send On Behalf Of permission** You can enable another user to send a message on behalf of a mailbox's owner. Exchange clearly indicates that the message is generated by one user on behalf of another, so these messages are obviously different from those sent using the Send As permission. Outlook users can delegate the same permission to other users. Personal assistants who support other people commonly use this feature.

- **Grant Full Access permission** You can grant full control over the mailbox to any other mail-enabled Windows user or to a security group. The steps necessary to assign Full Access follow.

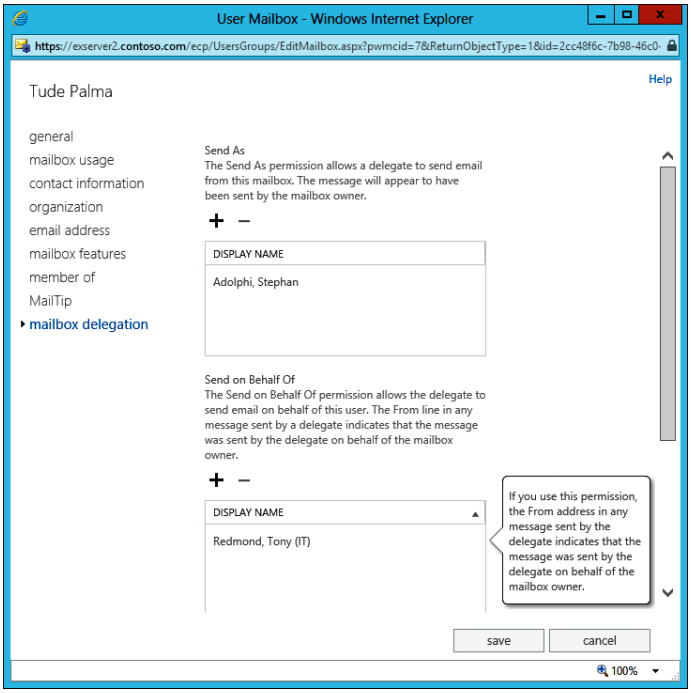


Figure 5-20 Editing mailbox delegation settings for a mailbox

> **Tip**
>
> **When you set the new permission, allow an hour or so before you attempt to use it to enable Exchange to refresh its Store cache from Active Directory; you won't be able to use the Send As permission until it is acknowledged by the Store.**

The difference between the Send On Behalf Of and Send As features comes down to the degree of impersonation used when a message is sent. When you use the Send On Behalf Of feature, the messages that are delivered clearly indicate that you have sent a message on behalf of someone else. This option is most useful when it's important to show that someone is assisting another user to deal with her email. When you use the Send As feature, the recipient sees no indication that you sent the message; it appears to come from

the person you impersonate. Usually, the Send As feature is used for functional mailboxes that are shared by groups of users, such as a mailbox dedicated to handling help desk queries.

Behind the scenes, the Send On Behalf Of feature requires Exchange to transport some additional information in the message header so that clients can display the name of the user who actually generated the message when the recipient views it. By comparison, the Send As feature requires a user to possess the Active Directory permission to impersonate someone else to send a message using his identity.

The process used to send messages with Send As permission varies from client to client. In the example shown in Figure 5-21, I created a message using Outlook 2013 and added the EMEA Help Desk as the sender in the From field. When the message was delivered, the user read it with Outlook Web App, and the only trace of any sender information is the EMEA Help Desk. In effect, the Send As feature enables you effectively to impersonate another user for sending email.

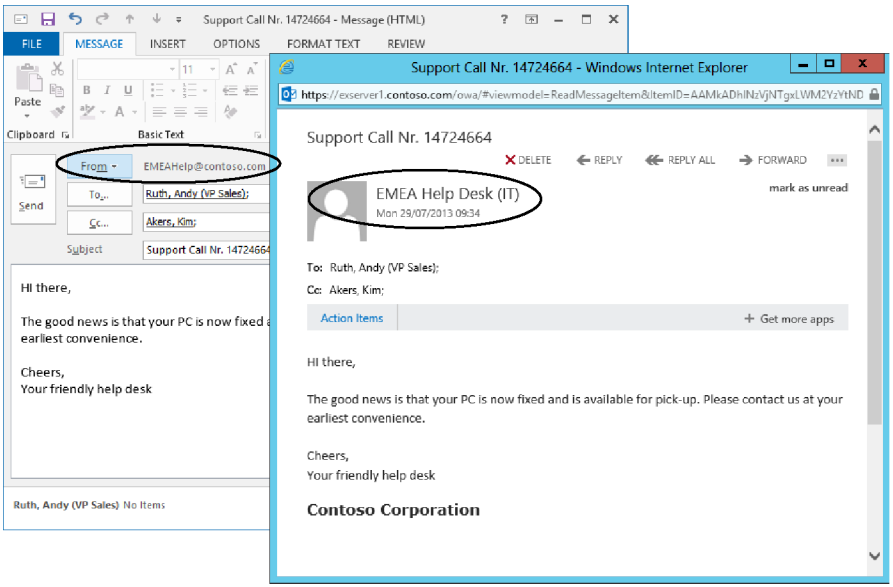


Figure 5-21 A message sent on behalf of the help desk

EMS is also available to manipulate mailbox delegation permissions. To begin, the equivalent EMS command to grant the Send On Behalf Of permission shown in Figure 5-20 is:

```
Set-Mailbox -Identity 'Tude Palma'-GrantSendOnBehalfTo "Redmond, Tony"
```

If you need to grant the Send On Behalf Of permission to several mailboxes at one time, you can just input a list of the desired mailboxes to the GrantSendOnBehalfTo parameter. For example:

```
Set-Mailbox –Identity 'Redmond, Tony' –GrantSendOnBehalfTo 'CEO Executive Assistant',
'Pelton, David', 'Akers, Kim'
```

Apart from mailboxes, you can grant the ability to send messages on behalf of distribution groups, dynamic distribution groups, and mail-enabled contacts. EAC supports delegation on behalf of most recipient types (it does not for resource mailboxes), but you can always achieve the desired result with EMS. For example:

```
Set-DistributionGroup –Identity 'Legal Department' –GrantSendOnBehalfTo 'Pelton,
David'
```

Naturally, you can assign the Send As permission to an account with EMS, but it's a little more complex because you need to manipulate Windows permissions rather than a simple Exchange property. The following code assigns the permission for the Help Desk account to my account. Note that it's using the Windows account name (as shown in Active Directory Users And Computers) rather than the Exchange display name or another value. Again, this is because you're manipulating Windows permissions behind the scenes.

```
Add-AdPermission –Identity 'Help Desk' –ExtendedRights 'Send-As'   –User 'Tony
Redmond'
```

To revoke the permission, use a command like this:

```
Remove-ADPermission -Identity 'Help Desk' -User 'Tony Redmond'  -InheritanceType
'All' -ExtendedRights 'Send-As' -ChildObjectTypes $Null -InheritedObjectType $Null
-Properties $Null
```

Managing Full Access permission

Assigning Send As permission for a mailbox enables a user to impersonate someone else, but apart from being able to create and send messages, the Send As permission does not grant any further access to mailbox contents. You need Full Access permission to open a mailbox and peruse its contents. To assign Full Access to a mailbox, select the mailbox to which you want to assign permission and edit its properties. Open the Mailbox Delegation section (Figure 5-22) and scroll to the bottom part of the screen, where you can choose the accounts or groups that need the permission. Notice that the Exchange Servers and Exchange Trusted Subsystem groups are already included in the list of users and groups. This is to allow Exchange access to the discovery mailbox so that it can store search results in it.

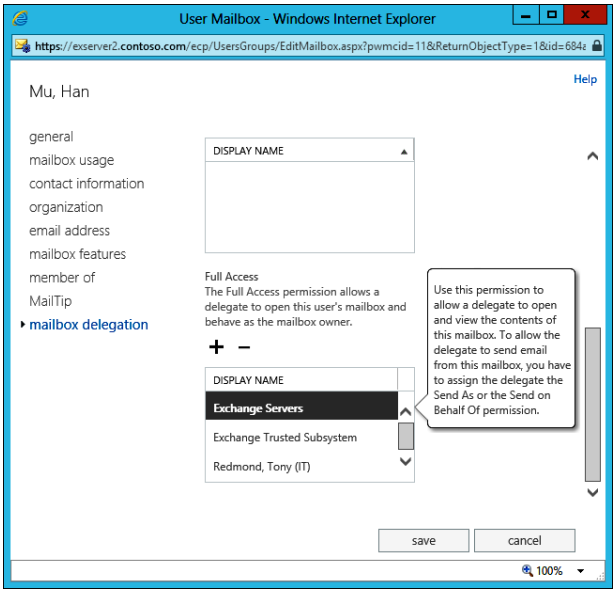


Figure 5-22 Assigning Full Access permission for a mailbox

You can add the names of individual users or the names of security groups, whichever is the more convenient way to assign permissions to mailboxes that need to be shared. You cannot assign permissions to normal distribution groups or dynamic distribution groups because these groups do not possess a Windows security principal. After you've entered the full set of users and groups to receive Full Access, click Save.

Behind the scenes: When EAC assigns permissions

Behind the scenes, the Add-MailboxPermission cmdlet makes the assignment. This cmdlet requires you to pass the name of the mailbox to which you want to assign Full Access. You can use any of the valid identifiers for this purpose. When EAC assigns permissions, it uses the fully distinguished name of the mailbox. The advantage of using a distinguished name is that you can be absolutely certain that you are working with the correct mailbox, which is always important when you deal with permissions. The disadvantage is that you have to be very careful about typing a distinguished name because it can be quite verbose or complicated. (For instance, the distinguished name for the default discovery mailbox is pretty long.)

One obvious example of when users need this facility is when they perform eDiscovery searches (see Chapter 11). Because Exchange places the output from these searches in discovery mailboxes (selected individually for each search), the investigators have to open them to review the results. Before the investigators can open the discovery

mailbox, an administrator has to assign them Full Access permission to the discovery mailbox. Exchange assigns the necessary permission for the default discovery mailbox by allowing anyone who is a member of the Discovery Management RBAC group to open the mailbox. However, if you create additional discovery search mailboxes to spread the load generated by eDiscovery searches as discussed earlier in this chapter, you must assign Full Access through EMS because EAC does not display discovery mailboxes.

If you set up additional discovery mailboxes, it's likely that you will give them names that are a little less obscure than the default one Exchange creates and should therefore be easier to enter. In this example, assign Full Access permission for a new discovery mailbox to the members of the Discovery Management group with this command:

```
Add-MailboxPermission -Identity 'CN=Legal Action Discovery Mailbox'
-User 'Discovery Management' -AccessRights 'FullAccess'
```

When you assign permissions to multiple mailboxes or groups, you have to use multiple Add-MailboxPermission commands. After all the permissions are assigned, you can check the permissions on the mailbox as follows. This output shows what you expect. The Discovery Management group is in the list of permissions, and you can see that an account called LegalTeam has been assigned access also. A user who receives Full Access permission can then open the mailbox with Outlook Web App.

```
Get-Mailbox -Identity 'Legal Action Discovery Mailbox' | Get-MailboxPermission |
? {$_.AccessRights -Like "FullAccess"} | Sort-Object Deny | Format-Table User,
AccessRights, Deny, IsInherited -AutoSize
```

```
User                              AccessRights   Deny    IsInherited
------                             -----------   ------  -----------
CONTOSO\Exchange Servers          {FullAccess}   False          True
CONTOSO\Discovery Management      {FullAccess}   False         False
CONTOSO\LegalTeam                 {FullAccess}   False         False
CONTOSO\Enterprise Admins         {FullAccess}   True           True
CONTOSO\Domain Admins             {FullAccess}   True           True
CONTOSO\Organization Management   {FullAccess}   True           True
```

Apart from the need to open and access content in discovery mailboxes, Full Access permission is commonly required in other scenarios such as when several users share a functional mailbox. For example, if you want to allow a group of users to open a mailbox belonging to a help desk so that they can see the problems and other issues reported to the help desk, you need to assign Full Access permission for the help desk mailbox to each

user who needs access. Granting access to a group rather than individual user accounts makes this process a lot easier to manage.

INSIDE OUT **What does Full Access really mean?**

Full Access permission allows users to access all the content in a mailbox, including the ability to create and save draft messages or add other content to the mailbox. However, Full Access permission does not allow the holder to assume the identity of the mailbox and use it to send messages because someone could access a shared mailbox and use his permissions to send insulting or otherwise inappropriate messages that could not be traced back to an individual. The term "Full Access" often creates an expectation that you can do everything with a mailbox that its owner can, but the reality is that you need to hold the Send As permission to send messages by using the mailbox's identity.

Remember that the Information Store caches permissions for mailboxes to improve performance. This means that it could take up to an hour before a permission change to allow Full Access or Send As for a mailbox becomes effective. Unfortunately, there are only two ways around this problem. You can restart the Information Store service to force it to reload its cache, or you can reduce the interval the Store uses to reload its cache. The first solution forces all clients to disconnect and isn't good to do during the working day; the second imposes an extra performance penalty on the server when it reloads the cache. Although some experienced administrators have reported good results when they reduced the logon cache interval to between 15 and 20 minutes, neither option is particularly attractive. The best idea might be to wait for Exchange to take its own good time to discover the amended permissions before attempting to use them—or, if possible, update permissions at a time such as midnight, when the delay enforced by caching won't affect users.

See *http://technet.microsoft.com/en-us/library/bb684892(EXCHG.80).aspx* for information about how to change the Reread Logon Quotas Interval registry entry to adjust the logon cache interval.

Mailbox auto-mapping through Autodiscover

Granting Full Access to a mailbox is one thing; a user being able to open the mailbox to access it is quite another. The traditional approach has been to modify the user's Outlook profile so that the mailbox is opened along with the other resources available to the user. To do this, edit the profile, select More Settings, and then choose the Advanced tab (Figure 5-23). Enter the name of the mailbox that should be opened.

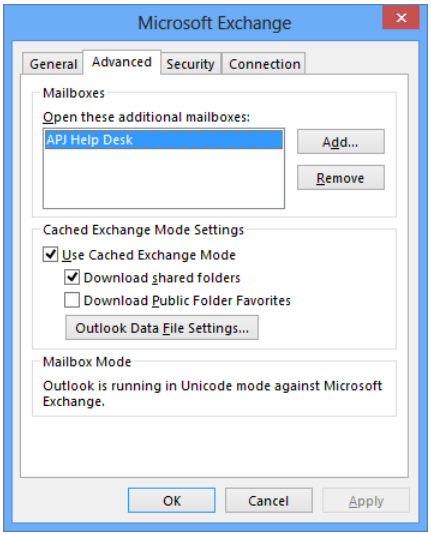


Figure 5-23 Telling Outlook to open another user's mailbox

This approach works at the expense of requiring an intervention by the user or an administrator to make sure that the profile is correctly configured. Fortunately, some Autodiscover magic happens to detect when users have Full Access to other mailboxes. Whenever a user is granted Full Access to a mailbox, Exchange updates the msExchDelegateLinkList property for the mailbox's Active Directory account. This property holds the distinguished names of all the accounts that have Full Access to the mailbox.

When Autodiscover runs, it checks Active Directory to find whether the account for which it is running has access to any other mailbox. If the account is listed in msExchDelegateLinkList for any other account, Autodiscover includes these mailboxes as alternates in the XML manifest it provides to Outlook; Outlook then inserts the mailbox in the resources it shows to the user. This feature is called mailbox auto-mapping, and it only works when Full Access is assigned to individual users rather than to groups. The same mechanism is used to provide Outlook with information about site mailboxes the user is entitled to open. (See Chapter 12 for more information about site mailboxes.) The information Autodiscover provides to Outlook looks like this:

```
<AlternativeMailbox>
        <Type>Delegate</Type>
        <DisplayName>Contoso PR queries</DisplayName>
        <SmtpAddress>ContosoPR@contoso.com</SmtpAddress>
</AlternativeMailbox>
```

Sometimes, you might not want a mailbox to show up automatically in Outlook, so auto-mapping can be disabled if necessary by including a parameter when you run the Add-MailboxPermission command to assign Full Access to a mailbox. For example:

```
Add-MailboxPermission "Sales Enquiries" -User TRedmond -AccessRights FullAccess
-AutoMapping:$False
```

This command assigns Full Access to the mailbox to the nominated user but does not populate the msExchDelegateLinkList property for the account. Full Access remains undetected by Autodiscover, and Outlook is not instructed to open the mailbox. If necessary, the traditional approach can be used to open the mailbox.

Opening another user's mailbox with Outlook Web App

Outlook Web App doesn't use profiles, nor does it use Autodiscover, so it cannot use this method to find out about newly acquired access to a mailbox. However, it's often convenient to use Outlook Web App to open another user's mailbox, like when an investigator needs to open a discovery search mailbox to examine items copied by an eDiscovery search.

To open another user's mailbox, click the down arrow shown beside the name of your mailbox on the far right of the Outlook Web App screen and then type the name of the mailbox that you want to open into the Open Other Mailbox dialog box. Click OK, and if your account has the appropriate permissions, Outlook Web App will open the mailbox.

Sending messages on behalf of other users

If you plan to send messages on behalf of other users, you should enable the From field in the client interface so that users can select the mailbox from which they want to send the message. For example, in Outlook 2010 or Outlook 2013, you select the Options tab and then click the Show From button. Note that if you configure your Outlook profile so that it opens multiple Exchange mailboxes, Outlook will always display the From field when you create a new message.

When the From field is enabled, the user can select the email account from which he wants to send by clicking the From button. This action reveals a drop-down list in which the user's own account is listed first (Figure 5-24). The user can then select Other E-Mail Address to have Outlook display the list of mailboxes on behalf of which she might send mail. It's also possible to type the name directly into the From field of a mail-enabled recipient on whose behalf she has the right to send mail, including a normal or dynamic group but not a mail-enabled contact. If a mailbox is selected from which the user doesn't have the permission to send mail, Exchange returns the message as undeliverable. When a recipient replies to a message that was sent on behalf of a user, Exchange delivers the response to the mailbox of the user for whom the message was sent.

The process to enable the From field to send messages on behalf of another user varies across clients and client versions, so you should consult the documentation for the specific

version of your client to determine the necessary steps. It's also true that some email clients might not support the feature, even when they are connected to Exchange.

One aspect that might surprise you is that the default for sending messages on behalf of another user is to retain the copy of the sent message in the impersonating user's mailbox. In other words, if Jill sends a message on behalf of Bob, the sent message is stored in the Sent Items folder in Jill's mailbox, and Bob sees no trace of the message that was sent on his behalf. The logic here is impeccable because although Jill has the permission to send messages on Bob's behalf, she has not been assigned the permission to store anything in his mailbox. Before the sent message can be stored in Bob's mailbox, you have to assign Jill Full Access permission over Bob's mailbox, which might or might not be what you want to do.

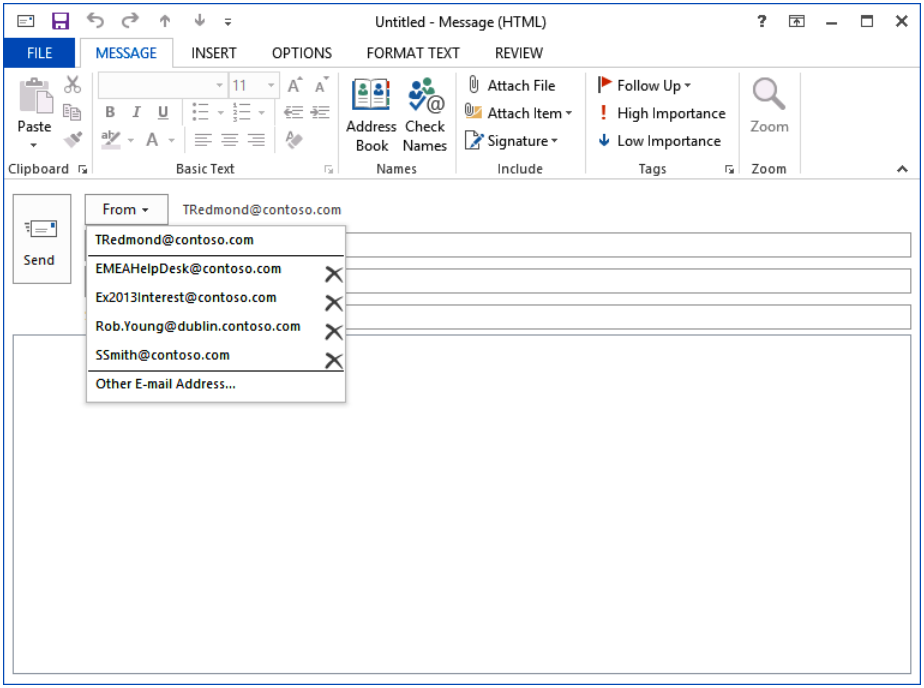


Figure 5-24 Selecting a mailbox on behalf of which to send a message

The easy workaround is always to add the person on whose behalf you are sending a message as a CC or BCC recipient. In this respect, a CC recipient is better because it allows the person to participate in future messages sent in the thread. However, a BCC recipient is quite sufficient if you just want to provide a copy for reference.

If you deem the workaround insufficient and want to force all messages sent on behalf of another user to be stored in that user's mailbox, you can make a change on the server

so that Exchange captures copies of messages sent by delegates no matter which client is used, including Outlook when running in online mode.

Auditing mailbox access

You can audit the actions a user (including administrators) who opens another user's mailbox takes by enabling auditing on the mailbox. (See Chapter 11 to capture details of actions such as hard and soft deletes, emptying the Deleted Items folder, and so on.) You can also audit actions mailbox owners take when they are connected to the mailbox.

Enabling mailbox auditing is a good option to consider for sensitive mailboxes such as those belonging to high-level executives or others in positions that might require you to provide data in response to discovery actions to demonstrate compliance with regulatory or legal requirements. Administrators can then run audit reports with EMS to retrieve details of actions the different users who worked with the mailbox took and have the reports mailed to a compliance officer or other user for his review.

Outlook delegate access

Apart from the control an administrator can exercise over mailbox access, Outlook users can also control access by using the Delegate Access option, found in the backstage area for Outlook 2013.

When you give delegate access to another user with Outlook, you automatically allow her to have Send On Behalf Of access, so she can send messages on your behalf. You also have the opportunity to delegate folder-level permissions over the Inbox, Calendar, Tasks, Contacts, and Notes folders, the logic being that these are the most commonly used folders to share information between a user and his administrative assistant or someone else who might need access to the mailbox. Access to other folders cannot be given through Outlook. If you need a delegate to access a different folder, you have to grant Full Access over the mailbox to that delegate or run the Set-MailboxFolderPermission command in EMS.

In Figure 5-25, you see that three users have delegate access to the mailbox. As stated previously, the act of adding each user as a delegate automatically grants her Send On Behalf Of permission. Each of the three users can have a different degree of access to the folders, represented by separate folder-level permissions. You can see these permissions by running the Get-MailboxFolderPermission command. For example, to see the access granted to the Calendar folder for my mailbox, I could run:

```
Get-MailboxFolderPermission -Identity Tredmond:\Calendar | Format-Table FolderName,
User, AccessRights
FolderName User                        AccessRights
---------- ----                        ------------
Calendar   Default                     {AvailabilityOnly}
Calendar   Andersen, Henriette-Thaulow {Editor}
Calendar   Akers, Kim                  {Editor}
Calendar   Amitai, Zwie                {AvailabilityOnly}
```

The default entry tells you that other users can access the calendar to fetch availability information when they want to see whether I am available for a meeting. The next two entries tell you that these two users have editor control over the calendar, meaning that they can create, edit, and modify entries. The last tells you that another user is a delegate but has no access to the calendar. This person can send email on my behalf but might not have been given access to any other folder.

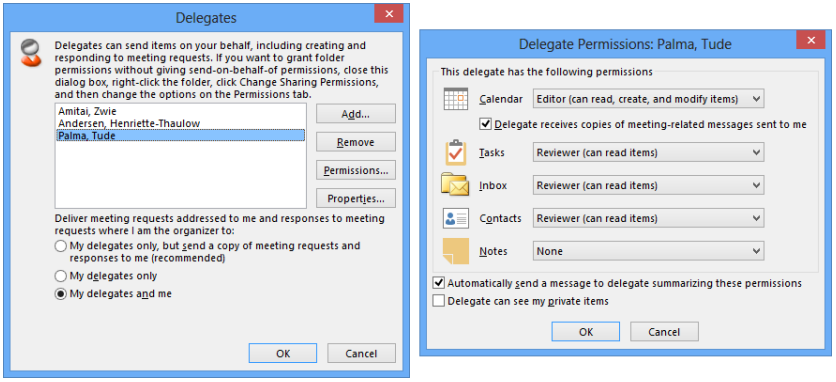


Figure 5-25 Setting delegate access with Outlook 2013

Shared mailboxes

A shared mailbox is a good example of Full Access permission in action. These mailboxes are used by teams of people who share a need to access common information such as a calendar, email, or tasks. Examples include a team of receptionists for a building, security staff, help desk staff, or generic email addresses that receive inbound customer queries for the company. (For instance, a shared mailbox with the sales@contoso.com address might be used to receive sales queries.)

Shared mailboxes have Active Directory accounts, but these accounts cannot be logged on to by users, mostly because the password for the account isn't shared. Instead, the users who need access to the information held in a shared mailbox are granted permission when the mailbox is created or by setting mailbox delegation properties for a shared mailbox afterward.

Exchange regards shared mailboxes as a slightly modified version of regular user mailboxes. They share the same UserMailbox recipient type but are assigned a recipient type detail of SharedMailbox. Thus, you can scan for all shared mailboxes in an organization with the following command:

```
Get-Mailbox –Filter {RecipientTypeDetails –eq "SharedMailbox"}
```

To create a new shared mailbox, open the recipients section of EAC, select Shared, and then click New (+). You will notice that a major difference in creating a shared mailbox is the immediate focus on assigning Full Access to the new object. Even though users cannot log on to the account used for the shared mailbox, they are automatically granted access to the shared mailbox through auto-mapping (see the "Mailbox auto-mapping through Autodiscover" section earlier in this chapter). Remember that Send On Behalf Of permission is also required if it is intended for people to send messages from the shared mailbox. This permission is granted separately from Full Access.

You might have regular user mailboxes created prior to Exchange 2007 that are accessed by multiple users in much the same way as a shared mailbox is intended to be used. These mailboxes can be converted into shared mailboxes by running the Set-Mailbox command. For example:

```
Set-Mailbox –Identity "EMEA Help Desk" –Type Shared
```

After you run this command, the mailbox is removed from the set of user mailboxes and shown by EAC as a shared mailbox. Existing permissions are maintained and continue to function.

Recalling messages

Exchange clients have offered users the ability to recall a sent message for many years. Unfortunately, this feature seldom works. It used to, just like it used to be a very useful feature, but the speed of modern servers and the capacity of modern networks have conspired to render the Recall Message feature unproductive in all but the most favorable situations. It's not really a problem for Exchange, but it can be a problem in terms of setting user expectations by explaining why message recall might not be possible after a user has looked for help in retrieving a potentially embarrassing message that he has just sent.

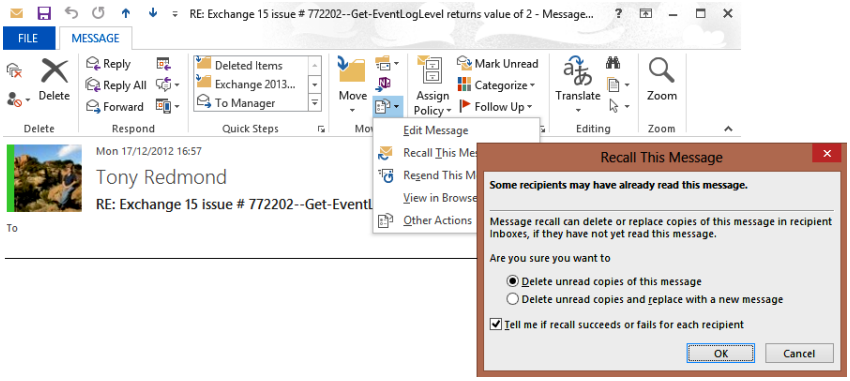


Figure 5-26 Recalling a message with Outlook 2013

When a user issues a Recall Message command, the client sends the request to Exchange. The server knows the unique identifier of the message and can use that information to track its progress. If the message is within the control of Exchange (in other words, within the boundary of the same organization from which the message was sent, and it has not been read by the recipient, it is possible for Exchange then to recall the message. However, many factors make the recall impossible, including:

- The recipient is not using a Microsoft client, specifically Outlook. In these instances, a request to remove the offending message from the recipient's mailbox is met by blank amusement on the part of the client, which probably does not understand the request Exchange issues. The message therefore remains in place. Given the pervasive use of mobile devices today, a very high probability exists that the message was delivered to a mobile device and is thus completely out of reach. In fact, even if Exchange manages to recall a message, a copy of that message might already have been pushed to a mobile device to render the exercise null and void. Of course, the recipient might then receive an informational message telling her that you attempted to recall the original message, which only heightens the desire to read its content.

- The message has been delivered outside the organization. This is a case of "out of sight, out of reach." After the message has been transmitted across a connector to an external system, it is gone. The speed of the Exchange SMTP-based transport system means that it does not take long for a message to be processed and dispatched through a send connector.

- The recipient is using Outlook but working in cached Exchange mode and has gone offline. When this happens, all bets are off because the server cannot influence what happens when Outlook is disconnected. If the message has been synchronized into the Outlook cache before the client disconnected, it will in all likelihood be read before the client reconnects to Exchange.

- A rule or some other process has moved the message from the Inbox. The message might have been moved into an archive mailbox or even into a PST. (Some clients still use rules that move incoming messages into a PST to reduce disk usage on the server, no longer possible with Outlook 2013.) If Exchange cannot find the message in the recipient's Inbox, its attempt to recall the message will fail because it cannot force a search for the message across all available message stores.
- The message has been read. There is no point in attempting to recall a message if it has already been accessed and read by the recipient. All modern clients have a reading pane that opens the currently selected message. Usually, the act of opening the message in this manner marks it as read after a certain period, perhaps 15 seconds. Thus, a high probability exists that the message has been read.

From this list, it should be apparent that message recall can be expected to work only in very limited circumstances:

- The user and the recipient are part of the same Exchange organization.
- The user executes the recall message request very soon after he sends the original message.
- The recipient does not use a mobile device that synchronizes email with her Exchange mailbox.

When these conditions occur, and the sun, moon, and stars align properly, message recall works.

Moderated recipients

Exchange enables you to apply moderation to mailboxes, contacts, and distribution groups. When moderation is enabled, any message addressed to the recipient must first pass through a moderation phase to approve or reject the message. Rejected messages are returned to the sender, and approved messages are allowed through for delivery to the original recipient. Moderation occurs in the transport service as a form of rule applied to messages after they are submitted by the Mailbox Transport Submission service. The scenarios to which moderation caters include the protection of sensitive mailboxes (such as those of executives) by forcing communications to be moderated by someone such as an administrator or of external contacts that might be published in the GAL to which you want to restrict communication from specific people who represent your company. Because they are often used to debate issues and post information that is of wide interest to many people, groups are possibly the most obvious target for moderation, so this discussion begins by looking at how to protect groups as an example of how the feature works.

Moderated groups

Many companies use distribution groups as a convenient way of communicating with large audiences. It is very convenient to be able to send a message and have it delivered to many people, but sometimes you want to ensure that only appropriate content is sent to large distributions. Moderation is a very effective way of accomplishing this goal to restrict the mailboxes and groups that can send new messages to a group. Figure 5-27 shows the properties of a group as it is enabled for moderation. In this case, a group has been created to facilitate discussion about Exchange 2013.

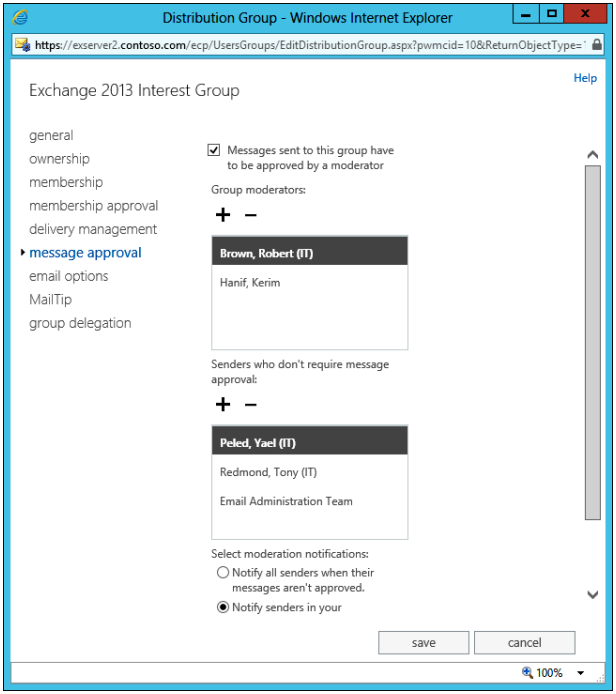


Figure 5-27 Setting the moderation properties for a group

People (including individuals outside the organization) might send inappropriate messages to the group (such as advertisements for products or other services). You can avoid the problem of external messages going to group members by making the group available to internal senders only, which is the default for Exchange 2013 (see the delivery management section of a group's properties). However, in terms of the actual content that is circulated to group members, you want to keep the discussion focused. To do this, you can nominate one or more moderators to review all the messages sent to the group. The users nominated as moderators do not have to be members of the group, and Exchange does not automatically add the moderators to the group membership. All moderation means is that the

specified users receive copies of messages sent to the group for approval before general distribution. Logically, it also means that if a moderator sends a message to the group, that message is delivered without needing to be approved.

You can also see that you can exclude some senders from moderation; any messages they send are posted without checking. The IT department probably knows something about Exchange 2013, so let the messaging administrators post directly to the list. You also configure the properties so that only internal users receive notifications if their messages are declined. It is a bad idea to send these notifications to Everyone because external spammers could receive a confirmation that they have sent a message to a valid email address if you allow external people to send to the group.

The EMS code to set the same properties to moderate messages going to the group is as follows:

```
Set-DistributionGroup –Identity 'Exchange 2013 Interest Group'
–ModerationEnabled $True –ModeratedBy 'Brown, Robert (IT)', 'Hanif, Kerim'
–ByPassModerationFromSendersOrMembers 'Email Administration Team', 'Redmond, Tony
(IT)', 'Peled, Yael (IT)' -ByPassNestedModerationEnabled $True
–SendModerationNotifications Internal
```

Note the use of the ByPassNestedModerationEnabled parameter. When set to $True, any nested groups that also require moderation are governed by the decision of the moderator of the group to which the message is addressed. In other words, if you send a message to a group called Investment Approvals that contains another moderated group called Management Committee, Exchange first sends the message to the moderator of Investment Approvals. If approval is received, Exchange then validates whether nested moderation is enabled. If yes (the flag is $True), Exchange distributes the message to individual recipients (those who are not within the nested group) of Investment Approval and sends an approval request to the moderator of the Management Committee group, who can approve or reject the request for distribution to the members of this group.

Tip

To avoid excessive delays for messages (and create less work for moderators), it's a good idea to enable nested moderation for all but the most sensitive groups.

After a group is set up for moderation, Exchange automatically displays a MailTip to warn users when they address a message to the group. The warning tells the user that his message might not be delivered immediately because it has to go through a moderation process to gain approval before final delivery (Figure 5-28). In this case, you see two MailTips. One is the default shown for any moderated group; the second is the customized message

for the specific group that is displayed simply because a MailTip has been defined for the group.

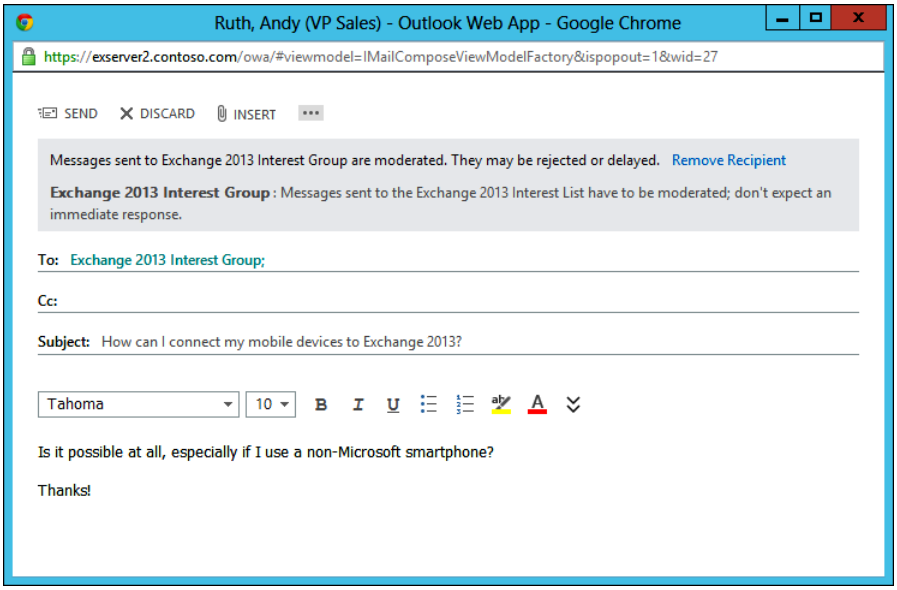


Figure 5-28 Viewing the MailTip that warns about a moderated group

How to enable moderation for dynamic distribution groups

Dynamic distribution groups support moderation, but you could not enable moderation for these groups through the EMC in Exchange 2010 because the EMC UI does not support this feature. However, the problem goes away in Exchange 2013 because EAC enables you to set moderation for a dynamic distribution group in exactly the same way as a normal group. You can also enable moderation of a dynamic group through EMS, but the range of parameters is a little less functional than is supported for normal groups. For example, the discussion about moderation of messages sent to nested groups is moot because dynamic distribution groups don't support this feature, so any message sent to a dynamic group whose resolved membership includes other moderated groups will require separate approvals for each group.

To enable moderation for a dynamic distribution group, you need code like this:

```
Set-DynamicDistributionGroup -Identity 'Contoso UK' -ModerationEnabled $True
-ModeratedBy 'Redmond, Tony (IT)' -ByPassModerationFromSendersOrMembers 'Sales
Department' -SendModerationNotifications Internal
```

Processing moderation requests

Moderators receive messages similar to the one shown in Figure 5-29 to enable them to approve or reject a message. Provided that a supported client is used (Outlook 2013, Outlook 2010, or Outlook Web App; other clients do not include the necessary UI), the process is very straightforward and requires only a simple click from the moderator to allow the message to proceed. When multiple users are assigned to be the moderators for a mail-enabled object, Exchange delivers copies of messages for moderation to every moderator. The first moderator who processes the request determines its outcome. In other words, if moderation requests go to two moderators, and the first moderator approves the request, Exchange respects that decision even if the second moderator attempts to decline the request a couple of seconds later. After a response is received, Exchange removes the moderation request from the mailboxes of the other moderators.

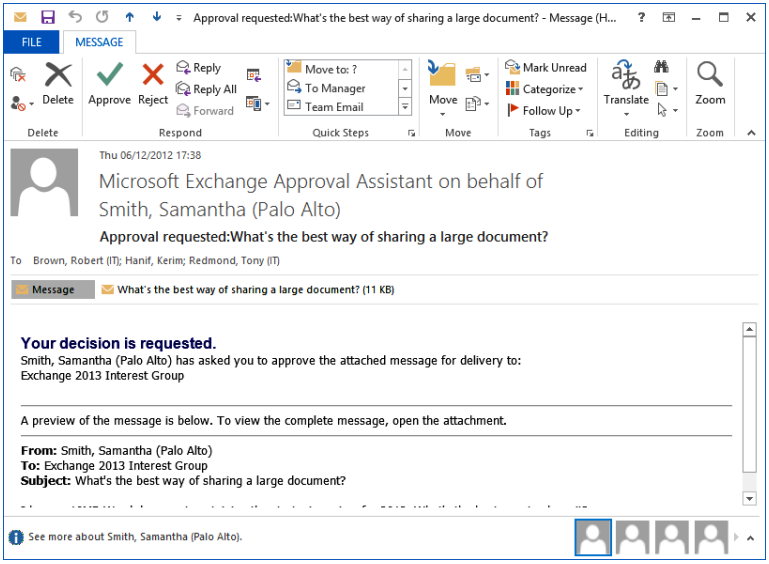


Figure 5-29 A request to approve a message posted to a moderated group

Logically, moderators are automatically excluded from moderation because a moderator is always regarded to be a trusted sender. Group owners are also excluded from moderation because if someone owns a group, he should be able to send to the group membership without hindrance. Also, if you don't assign a moderator to a group, the group owner automatically fulfills the moderation role and receives the requests for approval to send to the group.

Behind the scenes, the transport service is responsible for detecting when a message is sent to a moderated recipient. Some previous versions of Exchange do not support moderated

recipients, so moderation does not occur if a message sent to a group is processed by an Exchange 2007 hub transport server. The solution to this problem is to configure the moderated group always to use either an Exchange 2010 hub transport server or an Exchange 2013 Mailbox server. For example:

```
Set-DistributionGroup –Identity 'Exchange 2013 Interest Group'
–ExpansionServer 'ExServer3'
```

Of course, mailboxes and mail-enabled contacts don't use expansion servers, so this solution is not available for these objects. Instead, to process moderation properly, the servers that host these objects should run either Exchange 2010 or Exchange 2013 because this takes care of the problem.

When the categorizer running in the transport service detects a moderated recipient, it routes the message to an arbitration mailbox. This is a temporary holding location where the Store keeps moderated messages until they can be processed by a human. In this case, messages remain in the arbitration mailbox until they are approved or rejected by the group moderator who receives the requests for approval in her mailbox alongside regular messages.

Approved messages are then rerouted to the group and delivered as normal; declined messages are returned to the original sender. A process called Information Assistant is responsible for monitoring messages in the arbitration mailbox and routing them after approvals or rejections are received from a moderator. Information Assistant also cleans up the arbitration mailbox by removing old or orphaned requests that can accumulate there.

Note

The default expiry time for moderated messages is five days; this interval cannot be changed. When a message expires, Exchange returns it to the originator with a note to tell him that delivery didn't occur because the moderator failed to make a decision.

Users can see the current status of a message awaiting moderator approval through delivery or message-tracking reports but cannot do anything to encourage moderators to take action except send them a message (which they might ignore) or phone them.

Exchange does not maintain a special queue for messages awaiting moderation that an administrator can view to prompt a moderator to take action or to redirect a message if a moderator is unavailable for any reason. In addition, an administrator cannot log on to the arbitration mailbox to act on a message awaiting approval. Everything awaits moderator approval; if it doesn't come and a message expires, the moderator cannot take further action, and the message is returned to the originator with a reject status.

Messages can be sent to recipient lists that include moderated and nonmoderated recipients. In this case, the transport service bifurcates the message and delivers a copy immediately to the nonmoderated recipients. A separate copy is delivered to the arbitration mailbox to await attention from a moderator. A moderated group can contain subgroups, some of which also require moderation. You can allow a separate moderation process to occur for each group, or you can mark the group to allow automatic approval for all subgroups by setting its BypassNestedModerationEnabled flag. For example:

```
Set-DistributionGroup -Identity 'Exchange 2013 Interest Group'
-BypassNestedModerationEnabled $True
```

Journaling messages

If you enable journaling on either a per-mailbox or per-database basis, Exchange journals the messages as they pass through the arbitration mailbox. The following stages are captured:

- **The approval request from the arbitration mailbox to the moderator. The original message is captured as an attachment to this message.**
- **The approval or rejection decision from the moderator to the arbitration mailbox.**
- **If the message is approved, journaling captures the final message sent to the members of the distribution group.**

An arbitration mailbox is created automatically when you install Exchange. Apart from other functions such as storing mailbox search metadata, this mailbox is used to process moderated messages for every moderated object unless you decide to create and use additional arbitration mailboxes. You are unlikely to need additional arbitration mailboxes unless you need to spread the processing load across multiple sites because you make heavy use of moderated recipients. Exchange does not load-balance moderated recipients across available arbitration mailboxes; you must do this manually by setting the ArbitrationMailbox property on the recipient to force it to use a specific arbitration mailbox. For example:

```
Set-DistributionGroup -Identity 'Exchange 2013 Interest Group'
-ArbitrationMailbox 'ArbMbx London'
```

Moderated mailboxes

Moderated mailboxes are another common scenario in many Exchange deployments when the need exists to protect mailboxes the company might deem sensitive. Moderation provides an effective solution to this problem, but you can set up moderation for mailboxes only through EMS. In this example, moderation for the CEO's mailbox is set so that his

executive assistant does the processing of incoming messages. A MailTip to indicate that the mailbox is moderated is displayed when users address email to the CEO's mailbox. Ideally, you should customize the MailTip to add text to tell users who will moderate their messages as shown in the example code that follows.

Chapter 7 discusses how Exchange 2013 implements MailTips and the clients that support this feature. In the meantime, this command sets up moderation for the CEO mailbox, establishes a list of users whose messages bypass moderation, and creates the customized MailTip.

```
Set-Mailbox –Identity 'CEO Mailbox' –ModeratedBy 'CEO Executive Assistant'
–ModerationEnabled $True –ByPassModerationFromSendersOrMembers 'Executive Committee'
–MailTip 'Messages sent to the CEO are reviewed by their Executive Assistant (851-
1187) before delivery'
```

You can assign the responsibility for moderation only to other mailboxes; you cannot assign this task to a distribution group, even a security group. Instead, if you want to assign moderation to multiple users, you have to specify each mailbox individually. For example:

```
Set-Mailbox –Identity 'CEO Mailbox' –ModerationEnabled $True –ModeratedBy
'CEO Executive Assistant', 'CEO Support Team'
```

Finally, you can protect mail-enabled contacts. This command shows how you might apply moderation to a mail contact that points to an external recipient for the public relations (PR) agency. You don't want everyone in the company to communicate with the agency, so you apply moderation and set up a bypass for the members of the marketing department.

```
Set-MailContact –Identity 'PR Agency' –ModeratedBy 'PR Administrator'
–ModerationEnabled $True –ByPassModerationFromSendersOrMembers 'Marketing Dept'
```

Mail-enabled contacts

Mail-enabled contacts are a convenient way to add external correspondents (people who have email accounts outside your Exchange organization) to the GAL. Typically, contacts represent people working in other companies to whom a number of users need to send messages on a frequent basis. They are commonly used to facilitate contact with specific individuals or utility mailboxes for external vendors such as PR agencies. Exchange 2013 groups both mail users and mail contacts under Contacts in the recipients section of EAC.

To create a new contact, under Recipients, open the Contacts section and click + (New). Note that the drop-down menu provides options to create a new mail contact or a new mail user but that the default is to create a new mail contact. The screen EAC displays is straightforward; essentially, all you're doing is creating an Active Directory object to hold some details of the contact, including her email address.

Creating a mail contact with EMS is more interesting because more options are exposed. Here's an example of a command that creates a new mail-enabled contact. Note that the email address and the name properties must be unique.

```
New-MailContact -ExternalEmailAddress 'SMTP:David.Jones@fabrikam.com' -Name 'Jones,
David (Fabrikam)' -Alias 'DavidJones' -FirstName 'David' -Initials '' -LastName
'Jones' -OrganizationalUnit 'contoso.com/Exchange Contacts' –DisplayName 'David,
Jones (Fabrikam)'
```

It's also common for external contacts to be given a suffix so users understand that this object represents someone outside the organization, which might be important to know when they are addressing a message or browsing the GAL and have to select between two objects that have the same name. If the contact belongs to a specific company, you can use the company name as the suffix; otherwise, a general-purpose value such as *(External)* does nicely.

This command uses only a small subset of available parameters to manage the new mail contact. For example, you can define that the mail contact should receive messages only in plaintext format of a maximum message size.

```
Set-MailContact –Identity 'Jones, David (Fabrikam)' –MessageFormat 'Text'
–MessageBodyFormat 'Text' –MaxReceiveSize 500KB –UsePreferMessageFormat $False
```

Like other mail-enabled objects, contacts support moderation, so you can assign a moderator for the contact so that any messages sent to it are redirected to another user for approval before Exchange transmits them to the contact. Assume that you have a contact for a PR agency, but you don't want everyone in the organization to send requests for interviews, new advertising campaigns, and other requests to the agency. You could channel these messages to a moderator with a command like the following. Notice that I like to use the new MailTips feature (see Chapter 7, "MailTips") to provide users with an immediate indication that sending to a contact might not result in immediate delivery.

```
Set-MailContact –Identity 'PR Agency' –ModeratedBy 'Cook, Kevin' –ModerationEnabled
$True -SendModerationNotifications 'Always' –MailTip 'Messages to the PR Agency are
moderated by Kevin Cook'
```

Another way of putting a block on users sending to a contact is to set it up so that Exchange allows only messages sent by specific users. Using a group for this purpose is the most convenient method from an administration perspective.

```
Set-MailContact –Identity 'PR Agency' –AcceptMessagesOnlyFromSendersOrMembers 'PR
Department' –MailTip 'Only members of the PR department are allowed to communicate
with the PR agency'
```

Mail users

Mail users are much like contacts in that both object types have external email addresses. However, there is one important difference. Contacts are usually linked to people who have no relationship with your organization, but mail users are linked to Active Directory accounts. You can therefore think of mail contacts as being most suitable for people who work outside your company, whereas mail users are often the most appropriate choice for those who work inside the company.

INSIDE OUT **Mail users and security groups**

In essence, Exchange supports this recipient type to enable you to incorporate users in the GAL even when they use a different email system. Because mail users are linked to Active Directory accounts, they also are security principals and can therefore be added to security distribution groups that grant access to resources. You can add contacts to security groups, too, but this has no effect other than allowing the contacts to receive any messages sent to the group.

Mail users don't have their own section in EAC because they are treated as a form of contacts. Open Recipients, choose Contacts, and then click the drop-down menu beside the + (New) option. Select Mail User from the menu. When you create a mail user, you associate him with an existing Active Directory account or create a new account. In the latter case, you have to provide details for the account in much the same way as you do when you create a new account for a mailbox. Figure 5-30 shows a new mail user being created. In this case, an existing user account is selected, so no further information is required.

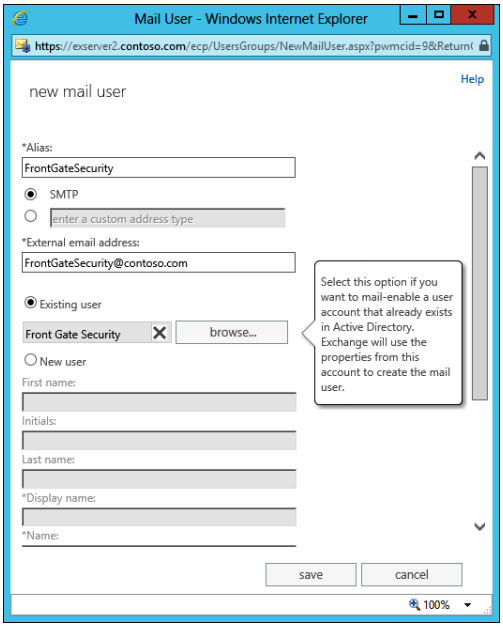


Figure 5-30 Creating a new mail user

The EMS code to create a brand-new mail user looks very similar to the code that creates a new mailbox. You must encode the password string into a variable before using it with the New-MailUser cmdlet (or, if you execute this code interactively and omit any mention of the password, EMS prompts you for it). Make sure that the alias and name properties are unique.

```
New-MailUser -Name 'Hamlin, Jay' -Alias 'HamlinJay' -OrganizationalUnit 'contoso
.com/Exchange Users'   -UserPrincipalName 'Hamlin@contoso.com'
-SamAccountName 'Hamlin'  -FirstName 'Jay' -Initials ''  -LastName 'Hamlin'
-ResetPasswordOnNextLogon $True  -ExternalEmailAddress 'SMTP:Jay.Hamlin@sendmail
.contoso.com' -Password $Password –DisplayName 'Hamlin, Jay'
```

Like contacts, mail users support moderation and MailTips. However, EMS is a little inconsistent in its cmdlets; you cannot set up a MailTip with the New-MailUser cmdlet. Instead, you have to apply the MailTip after the mail user is created with the Set-MailUser cmdlet. For example:

```
Set-MailUser –Identity 'Hamlin, Jay' –MailTip 'Messages sent to this address only
support plain text messages'
```

Resource mailboxes

Exchange 2013 supports mailboxes that are configured to represent rooms that can be added to meeting requests. Equipment mailboxes are a further mailbox variation that can be attached to rooms to represent the various items that support holding meetings in the room such as whiteboards, projectors, and tables. Although all mailboxes occupy space in databases, room and equipment mailboxes are differentiated through the type assigned to the mailboxes and the properties you can set on the mailboxes. Collectively, room and equipment mailboxes are referred to as resource mailboxes.

Note

Resource mailboxes have disabled Windows accounts. To create a separation between normal user accounts and resource mailboxes, it's a good idea to place these accounts in a separate Active Directory OU. No one ever needs to log on to a room or equipment mailbox to process the meeting requests they receive. As you'll see, the Resource Booking Attendant handles these requests automatically to confirm the booking or deny it because someone else has already booked the room.

Despite the fact that Exchange provides a useful All Rooms address list you can use to filter out room mailboxes when you browse the GAL (Figure 5-31), it's still important to use a suitable naming convention to identify room mailboxes to users and to populate information such as capacity and location to help people find the right room. Some companies name their rooms after cities, others after important or well-known people, and others after scientific inventions. Whatever convention you adopt, make sure that it is used consistently.

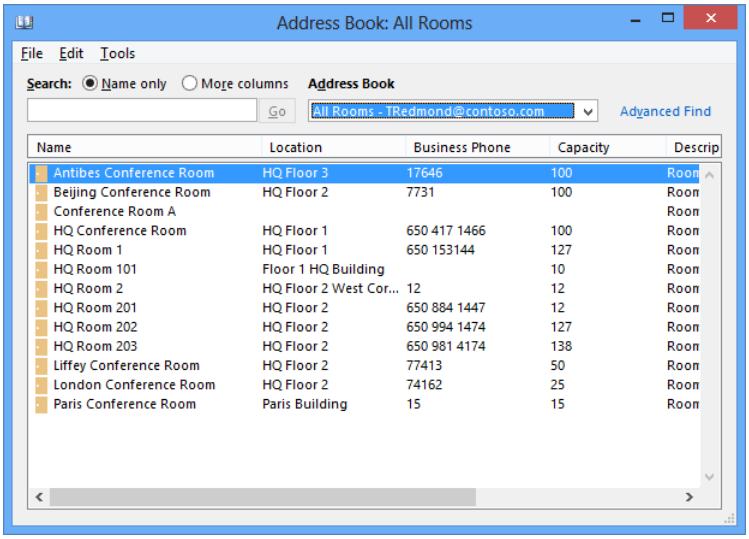


Figure 5-31 Viewing room mailboxes in the GAL

Outlook and Outlook Web App clients can include room mailboxes in calendar requests and can use a special form of distribution group called room lists to select from subsets of room mailboxes when scheduling meetings. This is a useful feature in large organizations when hundreds of room mailboxes might be in the GAL. Room lists are discussed in Chapter 6.

You can discover the current set of room mailboxes with this command:

```
Get-Mailbox -Filter {ResourceType -eq 'Room'} | Format-Table Name, Res* -AutoSize
```

Defining custom properties for resource mailboxes

You see two properties listed that can be used to communicate details about the rooms to users when they decide which room they'd like to book. The ResourceCapacity property states the number of people that can fit in the room. This is purely advisory; Exchange doesn't apply any intelligence to meeting requests to check that the number of attendees listed doesn't exceed the capacity of the selected room. The ResourceCustom property enables administrators to indicate whether anything special is available in the room. Before you can populate any value in the ResourceCustom property, you have to create a set of custom properties for the resource configuration with the Set-ResourceConfig cmdlet. For example, this command creates a basic set of items you might find in a room:

```
Set-ResourceConfig -ResourcePropertySchema ("Room/TV", "Room/Concall", "Room/WiFi",
"Room/Whiteboard", "Room/Video", "Room/ComfortableChairs")
```

The resource values can't have spaces and can't include hyphens (as in Wi-Fi). There is no UI in EAC to manage or expose the set of custom properties. If you need to update the set of custom properties, you can either rewrite the complete set or update the current set with a couple of lines of EMS code. For example:

```
$CurrentConfig = Get-ResourceConfig
$CurrentConfig.ResourcePropertySchema+="Room/PictureWindow"
Set-ResourceConfig -ResourcePropertySchema $CurrentConfig.ResourcePropertySchema
```

Note that the set of custom properties can also contain properties that differentiate equipment mailboxes. All the entries in the set you have seen so far are prefixed with "Room/" so Exchange knows that these properties apply to room mailboxes only. Properties used with equipment mailboxes are prefixed with "Equipment/" and can be added in exactly the way described earlier. After the resource configuration is populated, if a room was equipped with a Wi-Fi access point, that could be indicated by setting the property as follows:

```
Set-Mailbox -Identity 'Liverpool Conference Room' -ResourceCustom ('WiFi')
```

This command overwrites any existing custom properties that exist for the mailbox. If multiple custom resources are available in a room, you can populate them like this:

```
Set-Mailbox -Identity 'Liverpool Conference Room' -ResourceCustom ('WiFi', 'Concall')
```

The Exchange 2010 EMC reveals these properties through its GUI. However, EAC does not, so you have to manipulate resources custom properties through EMS.

Providing policy direction to the Resource Booking Attendant

Another important property is shown on the Resource General tab. When you enable the Resource Booking Attendant, you instruct Exchange that the attendant should monitor incoming meeting requests for the room to decide whether the requests should be accepted. The policy set for the room mailbox determines the action the Resource Booking Attendant takes. Each room mailbox can have a distinct booking policy you can see through the booking options section of a room mailbox's properties (Figure 5-32). If you compare the information and options presented by EAC to the equivalent shown by the Exchange 2010 EMC (on the Resource Information and Resource Policy tabs of a room mailbox's properties), you can quickly see that Exchange 2013 has considerably simplified the set of options available to administrators. However, this is only true for EAC; all the properties that can be tweaked to make the Resource Booking Attendant deal with a room precisely as you wish are still available through EMS.

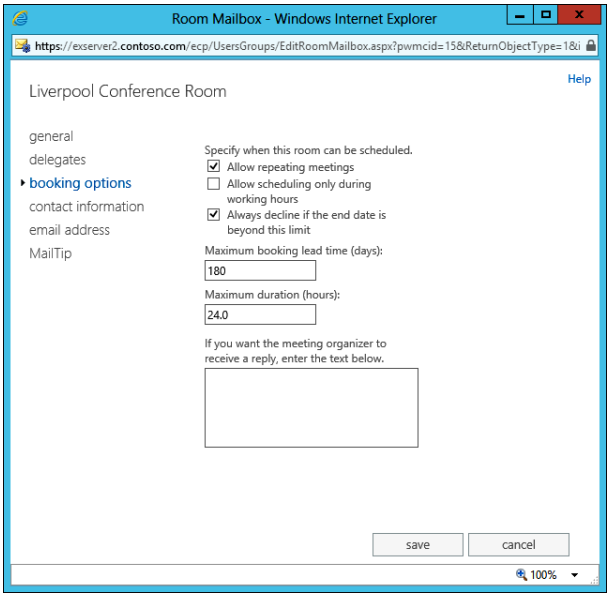


Figure 5-32 Viewing the booking options for a room mailbox

Unlike in Exchange 2010, EAC does not display the properties that control how the attendant publishes information extracted from incoming requests that it adds to the calendar of the room mailbox. This information is exposed if a user browses the room's calendar to look for a suitable time to schedule a meeting. The default is to remove any attachments, comments, and subjects for meetings because they often contain confidential information that you don't want all to see if they look for a meeting slot. You can change what items the Resource Booking Attendant publishes with the Set-CalendarProcessing cmdlet. The most important property you can manipulate is the value set for AutomateProcessing, which tells the Resource Booking Attendant how it should interact with the room mailbox. For now, assume that this property is set to AutoAccept, meaning that the Resource Booking Attendant can automatically process incoming calendar requests to reserve a time slot in the room's calendar. When this is established, a number of other properties influence what the Resource Booking Attendant does with a calendar request, including:

- **DeleteAttachments** Delete any attachments that are included with a calendar meeting request. Attachments are of interest to the humans that receive the request but are not needed by a room mailbox.
- **DeleteComments** Delete any comments sent along with a calendar meeting request.
- **DeleteSubject** Delete the subject of the meeting.

- **DeleteNonCalendarItems** Because a room mailbox is a mailbox with an SMTP address, users can send it email. It's pointless, but it can be done. The best thing is to suppress any noncalendar items that arrive in a room mailbox, so Exchange does this by default. Apart from anything else, this stops the mailbox from filling up with useless material that will never be read by anyone.

- **AddOrganizerToSubject** When people browse a room mailbox's calendar, they might want to know who has a room booked at a particular time. By default, Exchange adds the name of a meeting organizer so that this information is available to others. Sometimes this is extremely useful, like when knowing whom you might be able to persuade (or force) to give up a valuable time slot. However, it might be construed that letting others know who has booked a room gives away just too much information, in which case, you can update this property to $False.

- **RemovePrivateProperty** Meetings that are flagged as private have the private flag removed if a room mailbox is included. If you want to keep meetings really private, you probably shouldn't book a room—or arrange for this property to be set to $False.

- **OrganizerInfo** If set to $True, the Resource Booking Attendant sends meeting organizers an information message if a meeting request is declined because of a conflict. It's then up to the humans to resolve the conflict.

- **AddNewRequestsTentatively** Incoming meetings that are auto-accepted should be marked as tentative on the room mailbox's calendar.

INSIDE OUT Handling the effects of simplification

Even though EAC presents a more simplified view of calendar processing properties than is visible through the Exchange 2010 EMC, you should not assume that anything has really changed in the background. All the properties are still respected by the Resource Booking Attendant and can be manipulated through EMS. So even if you have built the most complex room-booking scheme possible using Exchange 2010, you can be safe in the knowledge that Exchange 2013 will continue to process meeting requests as before, even if it doesn't quite reveal this fact.

The complete set of calendar properties that influence the processing of calendar requests can be accessed with the Get-CalendarProcessing cmdlet. Note that these properties are present for all mailboxes and not just for room mailboxes. For now, examine the calendar processing properties for a room with:

```
Get-CalendarProcessing -Identity 'Leixlip Conference Room' | Format-Table
```

The Calendar Assistant can be configured to remove old meeting requests and responses from a room mailbox, but you can also apply a retention policy to room mailboxes to have old items cleaned up from their calendar after a suitable period (maybe two years).

At the bottom of the booking options section, you can see a field to allow administrators to provide some additional text the attendant inserts into messages that confirm or deny meeting requests. Administrators often use this property to describe the room or provide some information about how people can find the room if it's located in one of the cube-filled labyrinths favored by large corporations. You can see how this text appears to users in the bottom of the rejection message illustrated in Figure 5-33.

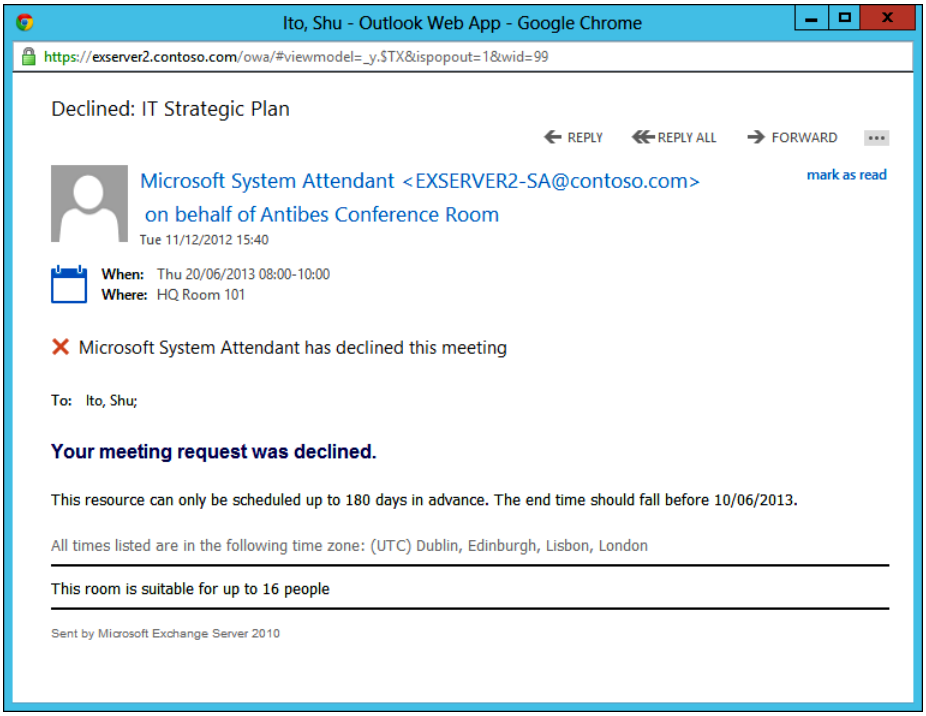


Figure 5-33 A meeting request declined by the Resource Booking Attendant

Booking options (Figure 5-32) for a room mailbox define basic rules for the Resource Booking Attendant to follow when it examines incoming meeting requests. Basic conditions include items such as whether the room is available outside normal working hours, how long meetings can last (the default is 1,440 minutes, or 24 hours; this allows for all-day meetings, but you might want to shorten the value), and whether to accept requests to schedule repeated meetings such as a team meeting that occurs at the same time every

week. It's also possible to prevent people from booking rooms too far in the future (180 days is the default) and to apply some level of automation to the resolution of conflicting requests. For example, the ConflictPercentageAllowed property (only available through EMS) sets a threshold for the instances of a recurring meeting that create conflicts, so if this property is set to 20, then any request for a recurring meeting will be declined if more than 20 percent of the meeting instances cause a conflict.

INSIDE OUT Meeting room conflicts

In real life, the most senior person will get to hold his meeting in the room, but because Exchange is unaware of company structure and politics, all it can do is allow you to define whether conflicting requests are accepted or immediately declined and, if accepted, how many conflict instances are permitted for a time slot. It would be silly to allow an unlimited number of conflicts to be scheduled because this would create some interesting situations when multiple teams turn up in the same place at the same time for their meeting. It might be acceptable to allow one or two conflicts if you have someone who checks the room's calendar from time to time to introduce some human intelligence into the conflict resolution algorithm.

To allow access to the calendar of a room mailbox, you use the Add-MailboxPermission cmdlet to assign the ChangePermission right to the user to whom you want to grant access to the calendar.

```
Add-MailboxPermission -AccessRights FullAccess -User 'Smith, Samantha'
-Identity 'Galway Conference Room'
```

When the permission to access the calendar is assigned, the assignee can open the calendar just like any other shared calendar and browse the entries to resolve conflicts. It's often easier and better for all concerned when a human resolves conflicts because he has much better knowledge of the organization context than any computer-driven process. In this instance, you can see that the CEO has requested use of a room, and that this conflicts with two other requests. Because Exchange does not assign a weighting to calendar requests that come from different types of users, a computer deals with the CEO's request as it deals with any other request. In most cases, a human will quickly realize that the right course of action is to let the CEO have the room and tell the other requesters that they'll have to find a different room—or argue with the CEO.

Calendar-related mailbox properties can be updated with the Set-CalendarProcessing cmdlet. For example, to restrict booking slots to 45 minutes and to accept booking requests only up to 120 days in advance:

```
Set-CalendarProcessing -Identity 'Leixlip Conference Room'
-MaximumDurationInMinutes 45 -BookingWindowInDays 120
```

Note

The Get-CalendarProcessing and Set-CalendarProcessing cmdlets replace the Get-MailboxCalendarSettings and Set-MailboxCalendarSetting cmdlets used in Exchange 2007, so you will have to update code if you have any scripts that use these cmdlets.

You can set the available hours for a room mailbox with the Set-MailboxCalendarConfiguration cmdlet. First, find out what the current settings are with the Get-MailboxCalendarConfiguration cmdlet. An edited version of its output is shown here:

```
Get-MailboxCalendarConfiguration -Identity 'Leixlip Conference Room' | Format-List
```

```
WorkDays              : Weekdays
WorkingHoursStartTime : 08:00:00
WorkingHoursEndTime   : 17:00:00
WorkingHoursTimeZone  : Pacific Standard Time
WeekStartDay          : Sunday
```

Because the booking of meeting rooms is a time-sensitive activity, it's important to set the mailbox's time zone correctly so that it corresponds to the location of the meeting room. The default time zone is inherited from the server on which the mailbox is created. In this case, you know that Pacific Standard Time (PST) is inappropriate because the room is located in Dublin, so update the time zone property as follows:

```
Set-MailboxCalendarConfiguration -Identity Leixlip 'Conference Room'
-WorkingHoursTimeZone 'GMT Standard Time'
```

Processing meeting requests according to policy

The AutomateProcessing property of a room mailbox tells the Resource Booking Attendant how to handle incoming meeting requests within the structure established by the booking policy for the mailbox. Three values can be present:

- **None** Exchange does not process incoming meeting requests.
- **AutoUpdate** This used to be the default setting for all mailboxes (including room mailboxes), and it allows the Calendar Assistant to place tentative meeting requests in user calendars without any intervention from the user. The user has to open and respond to the meeting request before it is confirmed or rejected.

- **AutoAccept** This is the default value set for room mailboxes created with Exchange 2013. It enables the Resource Booking Attendant to process incoming requests and accept or reject them according to policy.

To check which values are set for your room mailboxes, use this command:

```
Get-Mailbox -Filter {ResourceType -eq 'Room'} | Get-CalendarProcessing | Format-Table
Identity, AutomateProcessing
```

Requests to include room mailboxes in meetings are either in policy or out of policy. In-policy means that the request complies with the policy because it's inside the time permitted to book the room, is a suitable length, doesn't clash with other requests, and so on. Table 5-4 lists the properties that influence how the Resource Booking Attendant applies policy to incoming meeting requests for a room.

TABLE 5-4 Properties that influence processing of room bookings

Property	Meaning
AllBookInPolicy	If $True, the assistant automatically approves in-policy requests to book the room from all users.
AllRequestInPolicy	If $True, the assistant tentatively accepts in-policy requests to book the room from all users. Requests must be approved by a room delegate unless the AllBookInPolicy property is $True.
AllRequestOutofPolicy	Governs whether out-of-policy requests are acceptable for the room. All such requests have to be approved by a room delegate.
BookInPolicy	Lists the users whose requests to book the room are automatically accepted.
RequestInPolicy	Lists the users who are allowed to submit in-policy requests to book the room. All requests have to be approved by a room delegate.
RequestOutofPolicy	Lists the users who are allowed to submit out-of-policy requests to book the room. All requests have to be approved by a room delegate.
ProcessExternalMeetingMessages	Determines whether meeting requests will be accepted from users outside the Exchange organization.

As noted earlier, Exchange 2013 attempts to simplify calendar processing by reducing the number of options available through EAC. In previous versions, you could specify the names of users who are allowed to submit in-policy requests that the Resource Booking Attendant would immediately accept. You could also specify a separate list of other users who would be forced to have their meeting requests approved by a room delegate before acceptance. You can specify room delegates through EAC (Figure 5-34), but because EAC

does not expose the properties for the more advanced policy settings through its UI, they must be updated through EMS if you want to exert the maximum control over how incoming meeting requests are processed for a room mailbox.

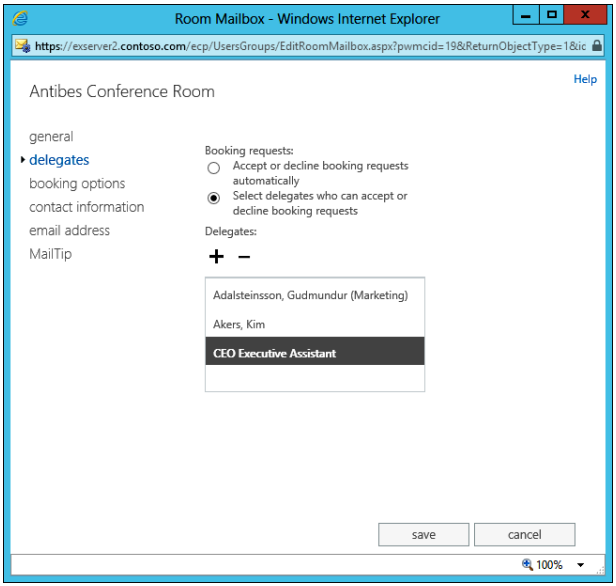


Figure 5-34 Viewing delegates for a room mailbox

The default situation is as follows:

- **AllBookInPolicy is set to $True, and AllRequestInPolicy is set to $False** This combination means that the Resource Booking Attendant can automatically approve any in-policy request to book the room from all users. An in-policy request doesn't exceed the maximum meeting time, is booked within the maximum request period, and so on.

- **BookInPolicy is blank** No special setting is in place to list users whose requests to book the room will be automatically accepted by the Resource Booking Attendant (if in policy). This is what you'd expect because AllBookInPolicy is set to $True.

- **RequestInPolicy is blank, too, for much the same reason** It wouldn't make sense to create a list of users who are forced to submit meeting requests for approval if the Resource Booking Attendant automatically accepts every in-policy meeting request.

- **AllRequestOutofPolicy is $False** Users are not allowed to request out-of-policy slots in the room's calendar.

- **RequestOutofPolicy is blank** Because no out-of-policy meetings will be accepted, you don't need to tell Exchange which users can set up such meetings.
- **ProcessExternalMeetingMessages is $False** You do not want to receive meeting requests from outside the organization. If set to $True, the Resource Booking Attendant applies the same policy conditions that exist for internal requests.

Now assume that you want to create special processing for a meeting room that needs to be tightly controlled, such as the meeting room adjacent to the CEO's office that is typically used for meetings that involve the CEO. You therefore want to keep this meeting room as available as possible for the CEO and her staff.

The first thing to do is amend the properties of the CEO Meeting Room so that it has at least one delegate. If you don't do this, the Resource Booking Attendant cannot forward meeting requests for approval, and the meeting will remain in a perpetual state of tentativeness. If you specify one or more delegates and set Select Delegates Who Can Accept Or Decline Booking Requests, EAC sets the AllBookInPolicy property to $False and the AllRequestInPolicy to $True when you save the changes. As soon as the change is made, Exchange routes any meeting request booked for the room to the delegates for approval.

The request is noted as tentative in the room's calendar until a delegate decides to accept or reject the request by responding to the message requesting a decision that the Resource Booking Attendant sends to him. Only one of the delegates has to give her approval for the meeting to be accepted (Figure 5-35). Note that a delegate cannot approve her own request for a room. To avoid this situation, you should add room delegates to the list of users whose request to book a room is automatically approved.

The CEO and his executive staff will probably not be impressed if a meeting they set up in the room needs to be approved. The best solution is to add the members of the executive staff to the BookInPolicy property so that any meeting requests they create are automatically accepted. You can identify users or groups by name, alias, UPN, or SMTP address. It's easiest if you use a distribution group to assign the exception because it's much easier to type the name of the group than all the individual users. A command like this should do it:

```
Set-CalendarProcessing –Identity 'CEO Meeting Room' –BookInPolicy 'Executive Staff'
```

If you do want to add individual users to the exception list, put a comma between each name.

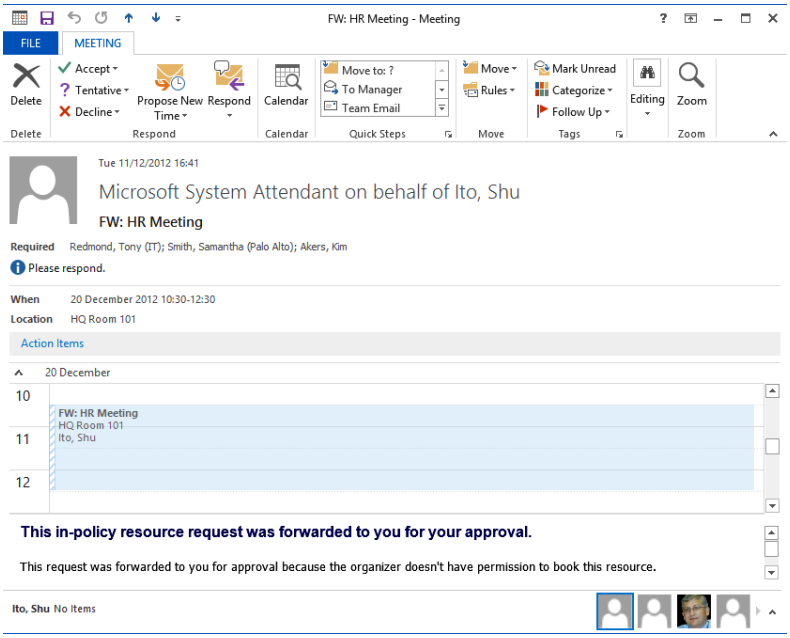


Figure 5-35 Viewing a room request sent to a delegate

Equipment mailboxes

Equipment mailboxes are very similar to room mailboxes in the properties you can assign to set a policy for their booking and the way the Resource Booking Attendant can monitor and process incoming meeting requests. Use this command to find the equipment mailboxes within your organization:

```
Get-Mailbox -Filter {ResourceType -eq 'Equipment'}
```

From a user perspective, equipment mailboxes are added to meetings like any other recipient. Unlike room mailboxes, neither Outlook Web App nor Outlook provides a specific list to enable users to select equipment to include in meetings, nor is there an All Equipment address list through which to browse to find equipment. For this reason, it's important for the naming convention applied to equipment mailboxes to identify these mailboxes clearly in the GAL.

Enough about mailboxes

The point is probably well made now that Exchange supports many forms of mailboxes, from users to health. It's time to consider other mail-enabled objects and components that have to be managed to make your Exchange organization successful. Let's move on to Chapter 6, "More about the Exchange Administration Center."

CHAPTER 6

More about the Exchange Administration Center

Groups 267
Dynamic distribution groups 300
Sharing 313
Certificate management 313
Mail flow and rules. 318
Delivery reports 320
Running EAC without an Exchange mailbox 329
Setting diagnostics for Exchange servers. 330
But what will you manage?. 332

Mailboxes are clearly very important objects to any server running Exchange Server, but the spectrum of objects that require management within an Exchange organization is much wider than just mailboxes. This chapter looks at other important components you need to understand and master, beginning with groups.

Groups

It's almost guaranteed that every Exchange organization includes lots of distribution groups, and the largest deployments usually include tens of thousands of groups. It is certainly useful to be able to address many recipients through one common address. However, distribution groups are seldom managed as well as mailboxes and tend to linger on in the directory long after they have ceased to be used. I worked at Hewlett-Packard until 2010, and even then, I could find groups called Windows 2000 Launch Team and Windows NT Forum in the Global Address List (GAL), both of which were artifacts of work long done.

Microsoft Windows supports four types of groups. Exchange really uses only two of these: universal groups and dynamic or query-based groups. Universal groups are available anywhere within the forest because their membership is published to global catalog servers. From an Exchange perspective, this means that you can be sure everyone in the group receives messages addressed to the group because every Mailbox server can expand the group's membership in the same way when it routes messages sent to the group.

Universal groups can hold security principals and be used as the basis for access control and communication, so they are very useful objects. The Exchange recipient type for mail-enabled universal groups is MailUniversalDistributionGroup unless the group is created as a security group; then it is of type MailUniversalSecurityGroup.

Like other Exchange objects, groups have a version number that indicates the version of Exchange that created (or last edited) the object. If you have groups created on versions prior to Exchange 2010, their versions will be updated to version 14 the first time you edit any metadata (but not membership) with Exchange 2013.

A new distribution group is devoid of members. Before it is useful, you have to populate the group with members, and the maintenance of group membership is an ongoing activity for the group owners. Dynamic distribution groups score on this point because their membership is evaluated by the transport system each time the group is included in an email header. A dynamic distribution group does not have the membership list found in normal distribution groups. Instead, its properties include an OPATH query that the transport system runs against Active Directory to determine the current group membership. Mail-enabled objects leave and join dynamic distribution groups based on their Active Directory properties. For example, if the query looks for every user located in Texas with an Exchange mailbox, the transport system sends a copy of a message addressed to this group to every user it can find in Active Directory whose state or province attribute is set to Texas. The advantage of dynamic distribution groups is that their membership always returns the most up-to-date information available in Active Directory. The disadvantage is much the same because if you don't populate Active Directory with accurate data, a query won't return the membership you expect or need.

Look at the previous example again. A query to find everyone in a specific country will succeed if the Active Directory objects you want to locate all have the same values in the country attribute but will fail if administrators use different spellings or ways to identify a country when they populate user accounts. *US* and *U.S*. are completely different values, as are *UK*, *U.K.*, and *United Kingdom*. Because the membership of dynamic distribution groups is evaluated only when required and is not represented by links to other Active Directory objects, these groups can't hold security principals.

Now that the difference between normal distribution groups and dynamic distribution groups has been established, further discussion about this topic will be left until later in this chapter.

Behind the scenes, Windows holds the membership for distribution groups as a set of backward pointers to the objects that represent the members of the group. These objects can be mail-enabled accounts, contacts, public folders, or other groups. In technical terms, the membership is stored in a single multivalued Member attribute that contains the pointers. Figure 6-1 shows the membership of a group displayed through Exchange Administration Center (EAC) (left), Active Directory Users and Computers (right), and ADSIEdit (foreground). In the past, the way Active Directory held group membership in a single multivalued attribute placed a practical restriction on the number of objects that could be members of a group. This limitation was removed with the introduction of linked value

replication (LVR) in Windows 2003, and you can now store well over 5,000 members in a group. However, maintaining the membership of very large groups can be a real pain, and most administrators try to keep groups under 1,000 members. Exchange 2010 introduced the ability for users to join and leave groups through the Exchange Control Panel (ECP) to relieve some work for administrators. Exchange 2013 moves the feature to the Options section of Outlook Web App.

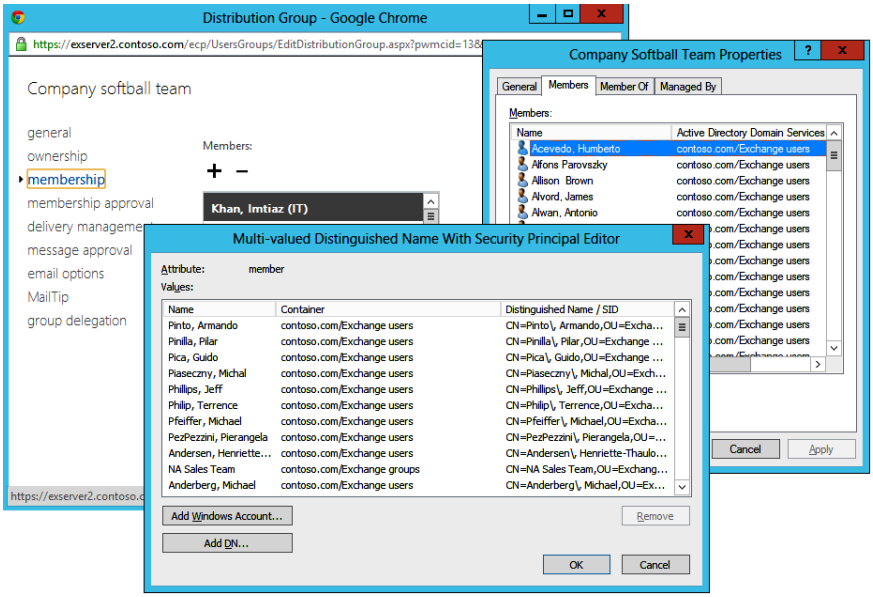


Figure 6-1 Three ways of viewing group membership

Managing groups

Distribution groups have long been a fundamental part of a messaging system. Traditionally, Exchange has allowed administrators to create and manage groups, including group membership, through its Administration console. Users access groups through clients such as Microsoft Outlook. However, the interaction users have with groups is largely read-only, and only a relatively small proportion of users have reason to maintain group membership. In versions prior to Exchange 2010, this function was often performed through Outlook. Outlook is an acceptable tool for group maintenance when the group is owned by the same domain as the account that attempts to update it, but it has proven problematic in multi-domain deployments if Outlook does not connect to a domain controller that holds a writeable copy of the group. Although less problematic than before, some problems (documented in *http://support.microsoft.com/kb/2586832*) still exist when using Outlook to manage Exchange groups.

Given that Outlook is not designed to be a directory maintenance tool, many companies have created self-service, web-based portals based on the Lightweight Directory Access Protocol (LDAP) to enable users to manage groups. These portals enable users to create, manage, and delete groups and are most useful in environments where multiple email systems share a common directory.

The majority of group management is performed through EAC. Groups are loosely divided into the following major categories:

- **Public groups** These are groups that anyone can join without permission.
- **Closed groups** Logically, these are groups whose membership is restricted. An administrator or group owner must add new recipients to these groups.
- **Moderated groups** These are groups users can ask to join and whose request is moderated by a group owner who makes the decision whether to allow the user to join.
- **Dynamic groups** These groups are based on an OPATH query Exchange resolves against the Active Directory to determine group membership whenever it is required, such as when the transport system needs to determine the addressee list for a message. Users never see the membership of dynamic groups, even when they receive a message addressed to such a group.
- **Room lists** These are a special form of group whose membership is composed completely of room mailboxes. The purpose of these groups is to provide a convenient mechanism for selecting rooms to book when creating a meeting. You cannot create a room list through EAC; you must use Exchange Management Shell (EMS). How to create these is discussed in the "Room lists" section later in this chapter.

Like other recipients, you can hide groups so that, for instance, they don't appear in the GAL. Users can still send messages to these groups by using the group's SMTP address.

Creating new groups

EAC uses a simple and straightforward process to create a new group. Figure 6-2 shows the basic elements of a group being populated. The following information can be entered:

- **The group name (mandatory)** This name is descriptive and should indicate the group's business purpose or the set of recipients who are addressed through the new group. The name of the group must be unique.

- **The group alias (mandatory)** An alias is necessary to identify the group. It is best practice for the alias for any object to be unique because Exchange uses it to generate the email address for the group. However, the alias does not have to be unique for a group, and Exchange will generate a unique email address for a group that uses the same alias by adding a number to the group's alias. For example, the second use of StrategicPlanners as an alias results in an email address of StrategicPlanners2@contoso.com.

- **Description** Some descriptive text is necessary to indicate the purpose of the new group. This information is displayed to users when they view group properties through the GAL.

- **Ownership** This is a list of one or more users who are entitled to manage the properties of the group, including its membership. EAC automatically adds the user who creates the new group as an owner. In the left-side screen of Figure 6-2, you can see that Administrator has been added because this is the account that is running EAC. It is usually best to change the ownership immediately by removing accounts such as the administrator and adding more appropriate owners (people who can manage the group over the long term). However, bear in mind that although any account that holds the Recipient Management role can delete a mail-enabled group, only a group owner can delete a security-enabled group, so it's best to keep an administrative account in the ownership of all security-enabled groups.

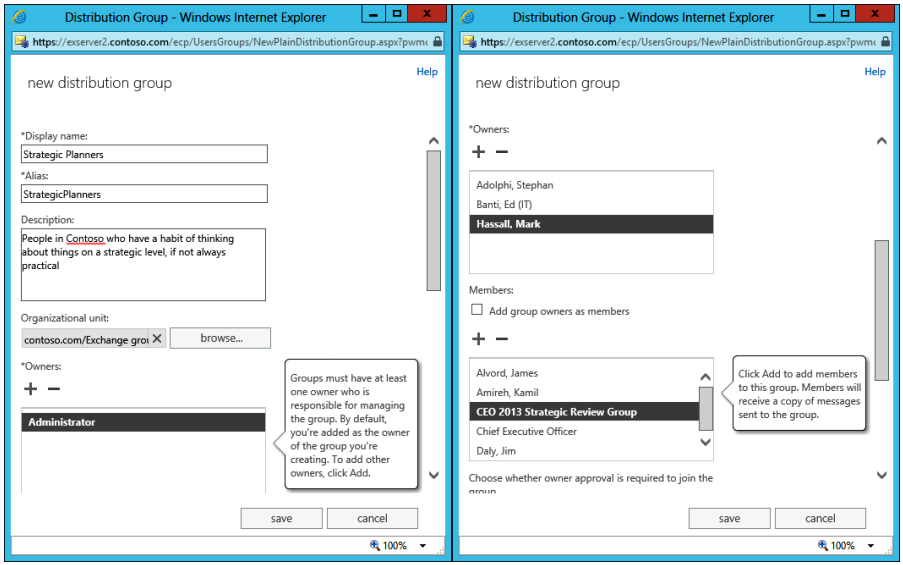


Figure 6-2 Creating a new group

- **Membership** This is the list of mail-enabled objects (mailbox, groups, public folders, and site mailboxes) that will receive messages sent to the group. Optionally, the managers of the group can be added automatically as members, so accounts such as Administrator join the group membership if you leave them as a group owner. This is a bad idea because administrative accounts can receive a lot of email they are neither interested in nor should receive. The best idea is to have administrators just create a group and then immediately hand the group over to its owners to populate the group membership. In the right-side screen in Figure 6-2, you see that the group ownership has been changed to reflect the group's real owners and that the group membership has been initially populated with other mailboxes and groups.

- **Membership approval** This is a set of properties that controls whether users can join or leave the group without administrator intervention, as follows:
  - The MemberJoinRestriction property controls how an end user can join a group.
  - The MemberDepartRestriction property controls how an end user can leave a group.

Note

Between the MemberJoinRestriction and MemberDepartRestriction properties, you can define exactly how users can control their membership of a group. The available values for these properties are:

- **Open** **Users can join or leave the group as they wish.**
- **ApprovalRequired** **Users can request to join the group, but the final decision is made by one of the group owners. If a user attempts to join a group for which approval is required, Exchange sends a request for approval to the group owners. If one of the group owners approves the request (Figure 6-3), Exchange adds the user to group membership.**
- **Closed** **Only a group member can update group membership.**

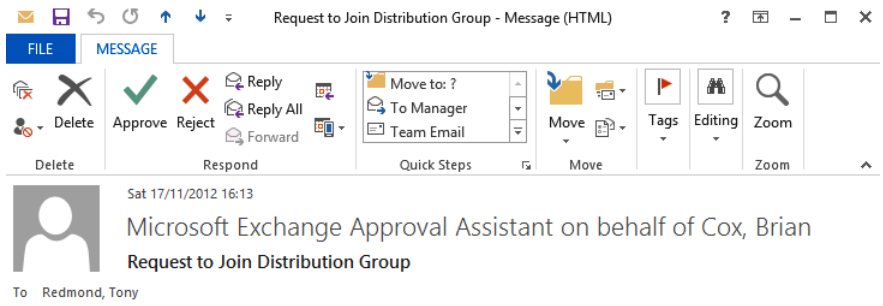


Figure 6-3 Viewing a membership request to join a group

Creating a group is only the start of the group's maintenance. The group is immediately usable in that it appears in the GAL, and users can address it. However, many properties can be tweaked to configure the group just right. If you edit the properties of the newly created group, EAC exposes a screen divided into the various properties that can be set. The sections are:

- **General** Set properties such as the name of the group, its description, and its alias. In addition, you can define whether the group appears in the GAL. People can still send email to a hidden group by using the group's SMTP address.

- **Ownership** Maintain the list of group owners. Note that only an owner can delete a group, so if you remove Administrator from the list of group owners, no one who uses EAC to manage Exchange with this account will be able to delete the group if the need arises. An administrator can add himself to the group ownership and then delete the group, except in the case of a security-enabled group, whose membership can be updated by an owner only.

- **Membership** Maintain the list of group members. In addition to editing the group's properties to update its membership, EAC includes a shortcut that's useful when you have only a single new member to add to the group. Move to the rightmost column in the list of groups, and a pencil symbol is displayed. Click the pencil, and EAC exposes a dialog box (Figure 6-4) to browse for the new member. (Strangely, EAC doesn't allow you to type the name or email address of the new member directly into

the dialog box.) After you've browsed the GAL and found the right person, click Save to add her to the group.

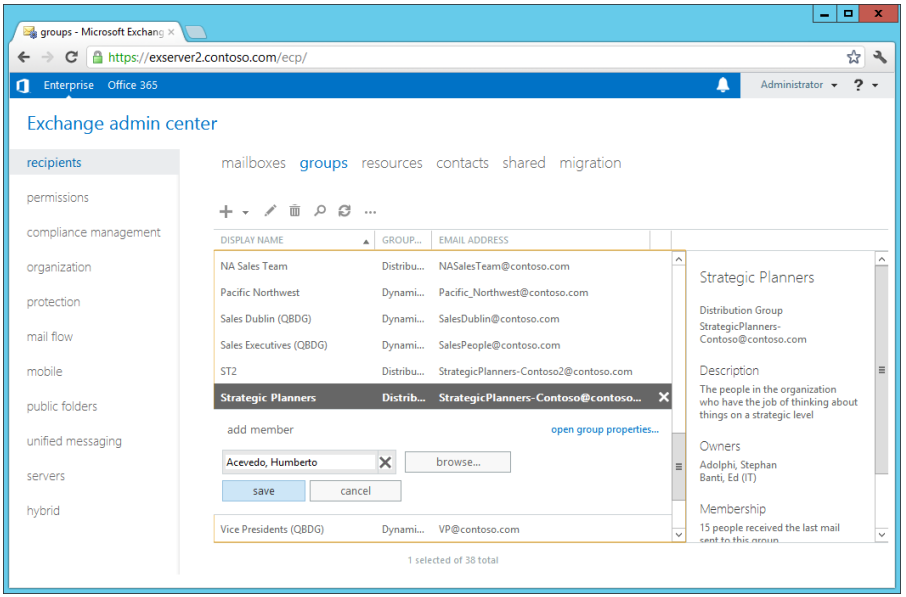


Figure 6-4 Shortcut to add a new member to a group

- **Membership approval** Control whether users can join or leave the group without ownership approval or whether applications to join the group will be moderated by the owners.

- **Delivery management** Control who can send messages to the group. The default is to accept messages from people inside the organization only; if someone outside the organization uses the group's SMTP address to send a message, Exchange will decline it and generate a nondelivery report to the sender. This step prevents the potential for spam sent to large internal groups. You can define that a group may accept messages from external correspondents. However, you can also limit the people who can send messages to the group, including external mail-enabled contacts. Typically, because it's easier to manage the membership of a group than individual entries, if a group is limited for delivery, another group controls who is allowed to send messages to the group.

- **Message approval** Define any controls Exchange applies when a message is sent to the group. You can instruct Exchange to send all messages that are addressed to the group to a moderator (either some nominated individuals or the group owners) who

will review the content and decide whether to accept the message for delivery to the group members. You can also decide whether to inform people whose messages are declined when this happens or to create a black hole so that the senders of declined messages never know that their message was declined.

- **Email options** Manage the email addresses for the group. By default, Exchange applies an email address policy to create a suitable email address for the group. You can exclude the group from the policy and define as many email addresses as you like, including those used for routing to older or foreign email systems.
- **MailTip** Create a customized MailTip that clients that support MailTips can display (Outlook 2010 and later plus Outlook Web App; see Chapter 7, "Addressing Exchange," for more information) such as, "Messages sent to the Strategic Planners group will be moderated and might be declined." MailTips can be up to 175 characters long.
- **Group delegation** Manage the Send As and Send On Behalf Of permissions for the group. You might want to have someone nominated to communicate on behalf of the group and have any messages sent for the group go out under the group's email address rather than individuals'.

INSIDE OUT **Using group metrics**

After a group has been in use for a little while, EAC displays two informational pieces of data: how many members are in the group (information calculated when a Mailbox server generates group metrics data) and how many recipients received a message the last time someone sent email to the group.

Group owners

When a group is created, Exchange assigns the account that was used to create it as the group's owner. This really doesn't matter if group membership is always maintained by administrators, but it can be useful to assign the ability to manage groups to individual users. Outlook was the only tool for this purpose that was available in previous versions of Exchange. If you own a group, you can use Outlook to select the group from the GAL and update its membership. This wasn't always possible with previous versions of Exchange, especially when the group was owned by one domain and your account was in another.

As discussed later in this section, users can create and manage groups through the Options section of Outlook Web App. You can limit users' ability to create groups while preserving the option to allow them to maintain groups they already own. If you decide that allowing users to maintain groups is the right approach, you might have to review the current ownership of groups to allocate the correct ownership because users won't be able to update a group unless they are listed as an owner. Remember that only an owner can delete a security-enabled group, so you have to make a decision whether to retain an administrative account in the group ownership to allow administrators to maintain those groups if the need arises (for instance, to transfer the group ownership to another user if someone leave the company).

Group ownership is maintained by selecting a group, viewing its properties, and choosing the Ownership section (Figure 6-5). You can add individual mailboxes or security groups to the ownership list for a group. These objects are security principals in that they can be authenticated by Windows. Normal distribution groups and dynamic distribution groups do not hold security principals and therefore cannot be used as group owners. Figure 6-5 shows that two users plus users who belong to the Exchange Administrators group manage the Email Interest List group.

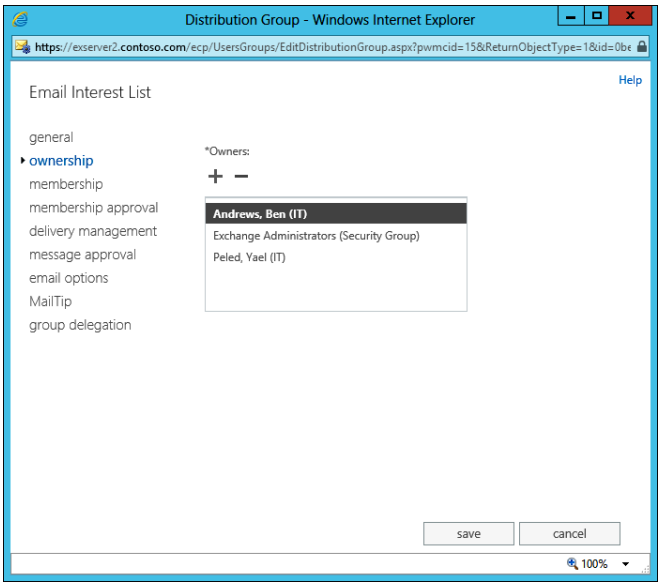


Figure 6-5 Amending the ownership of a group

You can also use the Set-DistributionGroup cmdlet to update the management list for a group. The input to the ManagedBy parameter is a comma-separated list of identifiers to the mailboxes of members you want to be group managers. The identifiers can be email

addresses, names, aliases, or even distinguished names—whatever format pleases you, as long as Exchange can identify the intended group owner uniquely. Although you can assign many managers to a group, it's usually best to limit the set to between two and six; more than this number creates the potential for confusion if several managers attempt to update the group membership at one time. The BypassSecurityGroupManagerCheck switch tells Exchange that it's okay to add these users as group owners.

```
Set-DistributionGroup –Identity 'Email Interest List'
-BypassSecurityGroupManagerCheck -ManagedBy 'Andrews, Ben (IT)', 'Peled, Yael (IT)'
```

You can use a command like this to check that the right owners have been assigned to the group:

```
Get-DistributionGroup –Identity 'Email Interest List'| Format-List Name, ManagedBy
```

Group expansion

Exchange expands the membership of groups included in message headers when the messages pass through the transport system. Exchange 2013 doesn't have a hub transport role, and the vast majority of the processing required to send messages, including group expansion, occurs on the Mailbox server. In previous versions of Exchange, it was common practice to assign the responsibility for group expansion to specific servers, and Exchange would route any message addressed by the group to the server on which the group is processed. When a group is expanded, the Mailbox server creates a list of the individual group members in memory and uses the list to send the message along the most efficient route. The expanded membership is not added to the message header because this would increase the size of the message. Each address adds between 1 KB and 2 KB to the size of a header, so it doesn't make much sense to take on this overhead, especially when some groups could have thousands of members.

Exchange 2013 takes a different approach and allows any Mailbox server to expand the group. The ExpansionServer property of the group is left blank, and EAC does not expose any user interface in group properties to allow an administrator or group owner to change the expansion server. Assigning a specific expansion server to a group has a downside in that Exchange cannot expand the group if that server is unavailable for some reason, thus creating the potential for a single point of failure within the messaging environment. It is therefore best to leave any Mailbox server able to expand a group. However, if you find that a need exists to direct a group to a specific Mailbox server to have its membership expanded there, you can still use EMS to set the necessary value. For example:

```
Set-DistributionGroup –Identity "Email Interest List" –ExpansionServer
'Exserver10.contoso.com'
```

If you're upgrading from a previous version of Exchange, it's a good idea to review groups that include specific expansion servers in their properties and then update these to allow any Mailbox server to do the job. Use the following command to discover groups that have a specific expansion server set together with their current owners:

```
Get-DistributionGroup | Where-Object {$_.ExpansionServer -ne ""} | Format-Table Name,
ManagedBy
```

Then, if you want to reset the expansion server property, amend the command to add a Set-DistributionGroup command:

```
Get-DistributionGroup | Where-Object {$_.ExpansionServer -ne ""} |
Set-DistributionGroup -ExpansionServer $null
```

Remember that expansion servers can also be defined for dynamic distribution groups, so you can perform the same check using the Get-DynamicDistributionGroup cmdlet to locate any of the groups that have an expansion server in place.

Tip

Keeping just the group name in the header enables recipients to respond to the message and be sure that Exchange will deliver it to the current membership and avoids problems with replies going to nonexistent addresses.

Protected groups

A protected group is one that is restricted in terms of the users who can send messages. As has been discussed, you can use moderation to protect groups, but you can also restrict groups in terms of from whom they are willing to accept messages. Figure 6-6 shows the basic approach. Select the group and edit its properties. Open the Delivery Management section. You can then select the users who are allowed to send messages to the group. EAC doesn't support the ability to list users who are explicitly blocked.

Note that a group member or even one of the group owners cannot send mail to the group if he is not listed in the set of people who are permitted to send messages to the group.

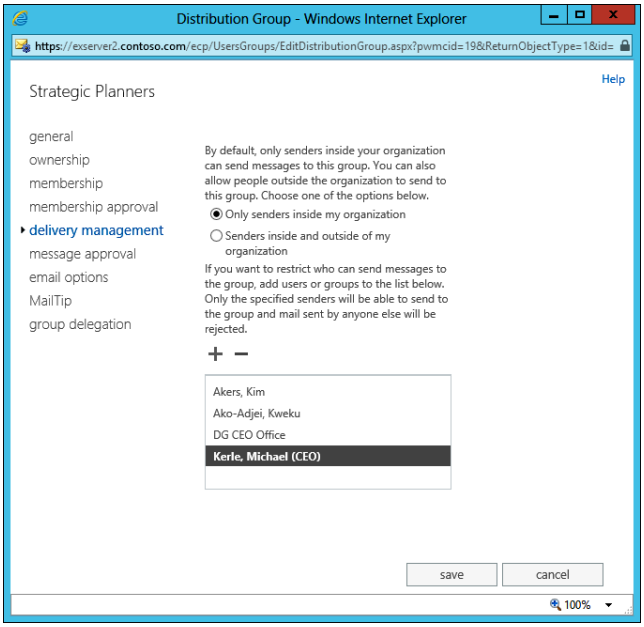


Figure 6-6 Limiting the mailboxes and groups that can send to a group

By looking at the properties of the group shown in Figure 6-6 through EMS, you can see the relevant controls:

```
Get-DistributionGroup -Identity "Strategic Planners" | Format-List *Accept*"
AcceptMessagesOnlyFrom                 : {contoso.com/Exchange users/Michael Kerle,
contoso.com/Exchange users/Kweku Ako-Adjei,}
AcceptMessagesOnlyFromDLMembers        : {contoso.com/Exchange groups/CEO Office}
AcceptMessagesOnlyFromSendersOrMembers : {contoso.com/Exchange users/Michael Kerle,
contoso.com/Exchange users/Kweku Ako-Adjei,, contoso.com/Exchange groups/CEO Office}
```

Three properties comprise the controls that are in place. AcceptMessagesOnlyFrom contains the mailboxes and mail-enabled users who are allowed to send messages to the group. AcceptMessagesOnlyFromDLMembers contains the names of groups whose members are allowed to send to this group, whereas AcceptMessagesOnlyFromSendersOrMembers contains the full list of the authorized senders from whom the group will accept messages. You can manipulate these properties by using the Set-DistributionGroup cmdlet.

```
Set-DistributionGroup -Identity 'Sales Planners'-BypassSecurityGroupManagerCheck
-AcceptMessagesOnlyFromSendersOrMembers 'Ruth, Andy', 'Akers, Kim', 'Sales Planners'
```

It's sufficient just to update the AcceptMessagesOnlyFromSendersOrMembers property because Exchange populates the value of the AcceptMessagesOnlyFrom and AcceptMessagesOnlyFromDLMembers properties based on your input. If you have only

a group or individual sender to add, you can do so by updating the relevant property. However, remember that:

- Adding an individual sender to the AcceptMessagesOnlyFrom property overwrites the existing list of individuals who can send messages to the group. Thus, unless you want to replace the existing senders with another set, you have to append the new sender to the existing list.
- If you update the AcceptMessagesOnlyFrom property, Exchange won't do anything to the AcceptMessagesOnlyFromDLMembers property. If you want to change the list of groups that are allowed to send to a group, you have to update the AcceptMessagesOnlyFromDLMembers property.

Because of the potential for confusion in dealing with three separate but interlinked properties that control delivery management, it's best to use EAC to update the list of senders who are permitted to send messages to a group. EAC populates the current list, sorts out whatever groups and individual senders you add to the list, and makes sure that the right entries are written back into the three properties afterward.

Going back to the EMS example, notice that I made sure to allow the members of the group to send messages to the group. Although it is conceivable that you might set up a security group to protect some resource and prevent the group members from sending messages to the group, in most cases it is desirable to allow the members to email the group.

After delivery management controls are in place for a group, anyone who isn't on the list who attempts to send messages to the protected group will receive a message delivery report (DSN error code 5.7.1) similar to that shown in Figure 6-7. Anyone using a client that supports MailTips will also see a notice telling her that she isn't allowed to send messages to the group—but she can go ahead if she wants, and Exchange will duly block her email. Hopefully, the text included in the message delivery report is clear enough so that anyone receiving such a response will understand the reason delivery failed and won't bother the help desk with a request to send email to the protected group.

The same error is displayed if an external user attempts to send messages to a distribution group that requires sender authentication. By default, Exchange groups require sender authentication; the sender must be known to the organization before the transport system will route incoming messages to the group. If you want to allow external users (perhaps from another Exchange organization in your company) to send messages to a group, you can turn off the authentication requirement by selecting Senders Inside Or Outside My Organization under delivery management, or by using an EMS command similar to this:

```
Set-DistributionGroup -Identity 'Email Interest List'
-RequireSenderAuthenticationEnabled $False
```

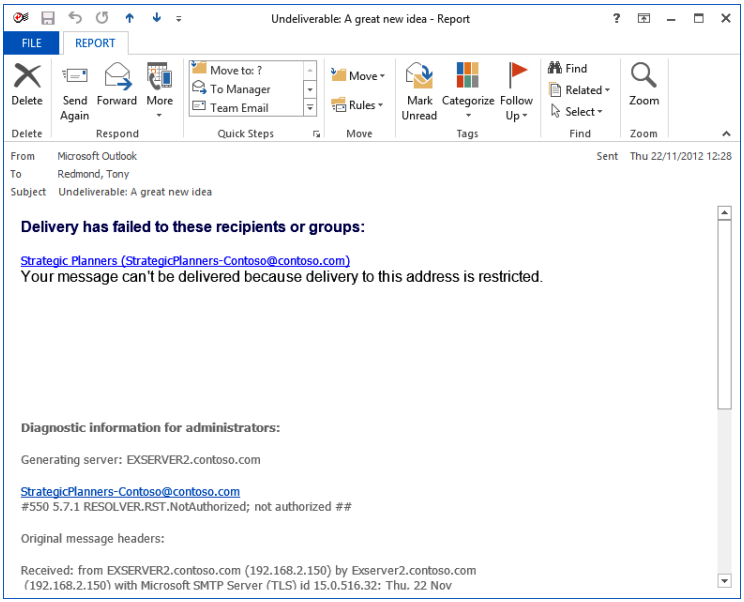


Figure 6-7 Message delivery report for a message sent to a restricted group

Viewing group members

The next question that arises after you've populated a group is how to be sure that the right members have been added. It's easy to review the membership of a small group by looking at the group properties with Outlook or Outlook Web App, but this approach fails after you have more than 50 members because the client interface is not suitable for browsing large groups. It would be nice to find a button that generates a formatted listing of group members, but no client includes this feature and neither does EAC. If you need to create a list of group members, you have to do it with EMS. Here's a one-liner to fetch the membership of a group, select two properties, and pipe the output to a text file:

```
Get-DistributionGroupMember –Identity 'Sales Executives'| Select-Object Name,
DisplayName > C:\TEMP\Sales.txt
```

A variation on the theme would be to pipe the output to a comma-separated value (CSV) file with the Export-CSV cmdlet that you can then manipulate with Microsoft Excel to polish the report.

For example:

```
Get-DistributionGroupMember –Identity 'Sales Executives'| Export-CSV
"C:\TEMP\Sales.CSV" -NoTypeInformation
```

Sometimes you want to find out quickly how many members are in a distribution group. You can do this as follows:

```
(Get-DistributionGroupMember -Identity ‘Sales Executives’).Count
```

You cannot fetch the membership of a dynamic distribution group in the same way because EMS does not include a cmdlet to expose the membership of a dynamic group. However, it’s easy to extract the recipient filter that underpins a dynamic group and use that to extract the membership. In this case, export to a CSV file. You should always use the Select-Object cmdlet when you want to export to CSV because the Format-Table cmdlet generates some interesting and totally useless information when it’s used in conjunction with Export-CSV and Exchange data.

```
$Filter = (Get-DynamicDistributionGroup –Identity "Vice Presidents (QBDG"")
.RecipientFilter
Get-Recipient –RecipientPreviewFilter $Filter | Select-Object Name, DisplayName,
Department, City | Export-CSV c:\temp\VPList.CSV
```

The resulting data will look something like this:

```
"Name","DisplayName","Department","City"
"Tony Redmond","Redmond, Tony (IT)","IT","Dublin"
"Kim Akers","Akers, Kim","Marketing","Palo Alto"
"Eoin P. Redmond","Redmond, Eoin P","Sales","Dublin"
"
```

Some companies don’t like the way Outlook reveals group membership so easily. They want the ability to stop users from seeing group members either through the GAL or by expanding the membership of a group after adding it to the header of a message. Common examples of group membership lists administrators are asked to conceal include confidential senior management teams, external advisors, and members of board committees.

It is possible to set access control lists (ACLs) on groups to block access to all but selected users. However, this arrangement is administratively difficult to control for all but a few groups. A better solution is to create a dummy distribution group that points to the real distribution group, which is marked as hidden from the GAL. In Figure 6-8, you see a group called Secret Board Committee that is visible to users through the GAL. If users look at group membership, they see nothing because the only member of the group (the real committee) is hidden. Ideally, the secret group is also well protected against random messages by setting the appropriate message delivery flags so that only the people who should be able to communicate with the group are allowed.

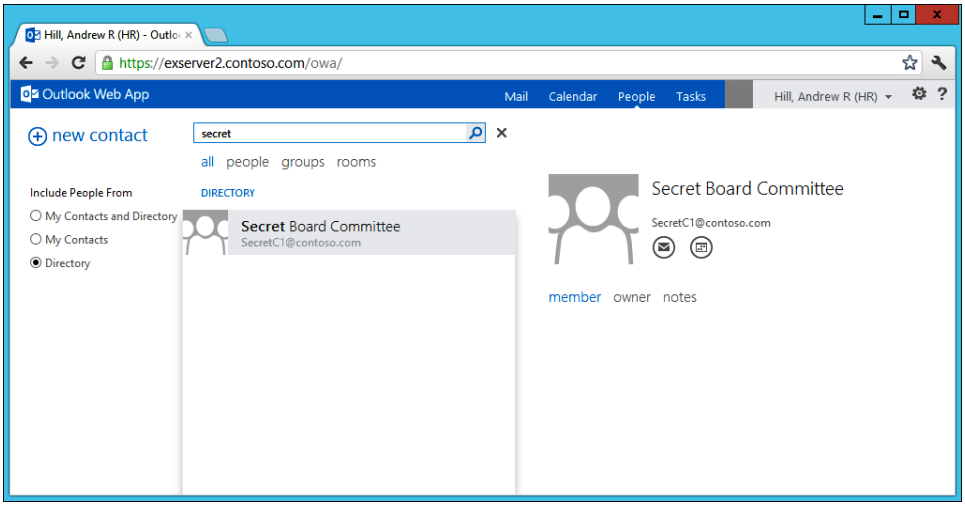


Figure 6-8 Stopping users from viewing group membership

Using a dynamic distribution group is an even better solution because no client ever exposes the membership of a dynamic group. You can exploit the 15 custom attributes available for mail-enabled objects to assign values that are then used by the filters for dynamic groups. Thus, you could create mail-enabled contacts for the external advisors that you want to hide and put EXADV in one of the custom attributes; the board committees might use another attribute, and so on. This approach has the great value of being easily implemented through the existing management tools (EAC and EMS) and not depending on some magic such as manipulation of ACLs that could be affected by a change in a future version of Windows or Exchange.

INSIDE OUT An additional control through Outlook 2010 and Outlook 2013

Outlook 2010 and Outlook 2013 impose an additional control over moderated groups (those that require approval before messages can be sent to them). If you add a moderated group to a message header, right-click, and select Expand Group, Outlook will refuse to insert the addresses of individual members of the group into the message header.

Security groups

The default type for a new group is a mail-enabled universal distribution group. If you need a group to be a mail-enabled security group (one that holds a security principal), you need to select it to be a security group when you create it through EAC by clicking the down arrow beside the + (plus) sign and selecting Security Group from the drop-down list.

The initial set of properties (name, alias, and so on) is similar to a regular distribution group. However, security groups don't allow users to join or leave without owner permission. This restriction is quite logical. A security group holds a security principal that permits access to some kind of resource. It wouldn't be a very secure situation if users could gain access to a security principal without some form of oversight. Likewise, because security groups represent a security principal, it's important for you to assign ownership of the group to the right people who can then control who else is allowed to be part of the group.

Users can create security-enabled groups through Outlook Web App only if they hold the Organization Management role. (Users who hold the Recipient Management role can create security groups through EAC; Microsoft might address this disconnect in role-based access control [RBAC] roles in a future release.) Figure 6-9 shows the version of the New Distribution Group dialog box Outlook Web App presents. You can see the option to Make This Group A Security Group, which is dimmed for users who do not hold the Organization Management role.

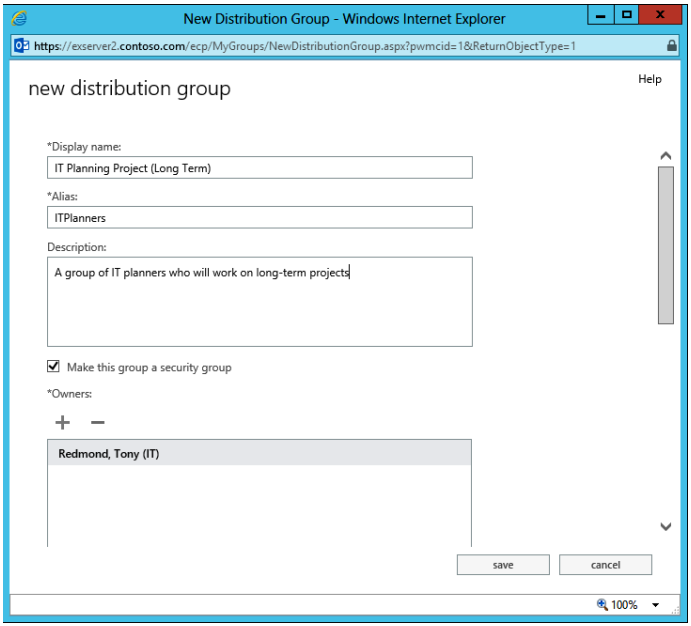


Figure 6-9 Creating a security-enabled group through Outlook Web App

Because it's not supported by the Set-DistributionGroup cmdlet, you can't use EAC or EMS to modify the properties of a mail-enabled distribution group to make it into a mail-enabled security group after it is created. However, you can change a group's properties through the Active Directory Users and Computers console to make it a mail-enabled security group. If you use Active Directory Users and Computers to change a group to become a mail-enabled security group, it updates the Active Directory properties as follows:

- GroupType is changed from Universal to Universal, SecurityEnabled.
- RecipientType is changed from MailUniversalDistributionGroup to MailUniversalSecurityGroup.
- RecipientTypeDetails is changed from MailUniversalDistributionGroup to MailUniversalSecurityGroup.

However, the Exchange-specific properties that control whether users can join or leave the group are not changed. This can lead to some apparent inconsistencies. For example, if you upgrade a group that is open to joining and leaving, the MemberJoinRestriction and MemberDepartRestriction properties will still show as Open, and you might conclude that it is possible for users to get around the normal restriction that only group owners are allowed to update membership. In fact, even though the membership seems to be open, Exchange won't allow users to join or leave the group and signals an error if they try to do so.

Tracking group usage

Groups are very easy to create, are a powerful method to address lots of recipients at one time, and are reasonably easy to maintain. The only downside is that groups tend to linger in Active Directory and clutter up the GAL well past their best-by date. It's common to create groups to assist communication for an event or some other purpose. The group is used for many messages in the time up to the event. Afterward, it sits in the GAL waiting to be used again and stays there until an administrator concludes that the group is no longer used and puts it out of its misery. Two questions arise from this scenario. The first is how to know whether groups are being used. The second is what to do for groups that are no longer used.

Exchange doesn't include functionality to analyze the usage of groups. There are third-party products that produce statistics from an Exchange server, but it's also possible to generate some rudimentary data about groups from the message-tracking logs. Every time a Mailbox server expands group membership, Exchange logs an event in the message-tracking log. The event also includes the SMTP primary address of the group that was expanded. You can search for the Expand records and generate counts of how often groups are used. For example, this code scans the message-tracking log for the events in which you are

interested, uses the Group-Object cmdlet to sort the events by the SMTP primary address, and finally outputs a table showing the number of times each group was used.

```
Get-MessageTrackingLog -Event Expand | Group-Object RelatedRecipientAddress |
Format-Table Name, Count -AutoSize
```

```
Name                                Count
--------                            -------
Sales@contoso.com                   12
Marketing@contoso.com               11
Management@contoso.com              9
IT@contoso.com                      1
```

There are many ways to build on this code. For example, you can check for usage in a particular date range or output data in a form you can collect from all Mailbox servers over time to provide an organization-wide view.

Now that you know how to identify the usage of groups, you can review the groups in the GAL and determine the groups that no one ever uses. If a group hasn't been used in a couple of months, it's a good candidate to be removed—or at least hidden—from the GAL for a month or so to see whether anyone notices. If no one does, you can safely delete the group.

Allowing users to manage groups

For many companies, giving users the ability to know what groups they belong to, to know what groups are available to join, and to join and leave groups without administrator intervention is the preferred way of working. The advantage here is that users do the work to manage groups and so relieve help desks or other administrative staff of maintaining group membership and so on. The downside, especially when users are allowed to create new groups, is that you might accumulate a profusion of groups within the organization, something that can cause confusion if, for instance, a number of groups share similar names but have different membership. For example, it's difficult for a user who wants to send some information about a new sales campaign to the sales teams in the company if she sees six groups that might be appropriate:

- All Sales
- Sales Executives
- Sales Teams
- Sales Group

- Sales Department
- Sales Campaign

Other companies prefer to keep groups under strict control and prefer that the only interaction a user has with a group is when she receives email sent to the group. Exchange 2013 enables you to implement both modes of working by using role-based access control to determine the options presented to users in the Options section of Outlook Web App.

With Outlook Web App, users can do the following:

- View the groups to which they belong
- Update the membership of the groups of which they are a registered manager
- Delete groups they own
- Create new groups
- Join or leave open groups someone else has created

Planning for user-maintained groups

Before you allow users to create groups, consider the consequences that might flow from this decision.

Users don't typically plan how best to create groups and tend to create groups without checking to see whether an appropriate group already exists. They are unlikely to follow any naming conventions. You could therefore have duplicated groups scattered throughout the GAL. This is a good reason to implement a group naming policy for user-created groups.

In the same light, apart from occasionally updating group membership, users are unlikely to maintain groups. Orphan groups will occur after users leave the organization. The result is the accumulation of many outdated and unwanted groups that linger in the GAL until an administrator deletes them.

Users can create mail-enabled universal groups; they cannot create mail-enabled universal security groups unless their account holds the Organization Management role. If a user needs a mail-enabled universal security group, she can create a group as normal and then ask an administrator to upgrade it to become a universal security group.

Consider the appropriate location in Active Directory for user-created groups. For more information about how to configure the default organizational unit (OU) for group creation, refer to the "Defining a default group location" and "Creating a group naming policy" sections later in this chapter.

These issues escalate in importance as the organization grows. Allowing users to create their own groups in a small organization, in which it's easy to keep track of what's happening in the GAL, is one thing; allowing 100,000 users to create groups and accumulating 250,000 extra groups in the GAL is quite another. Users will find it difficult to find recipients in the GAL, and Offline Address Book (OAB) file downloads will become increasingly large. Even worse, the thought of cleaning up tens of thousands of unwanted or unused groups on a regular basis is numbing.

Defining a default group location

If you allow people free rein to create groups, it's likely that a wide variety of names will be used for groups—anything from friendly names such as "Sandy's best friends" to the more algorithmic "ABC-100-Group-A." It's important for groups to have appropriate names that are easy for users to find when they look for the right group to which to send a message. You can't do much about the intended use of a group because the way they are used to share information varies from groups that reflect organizational structure (the members of a department) or users at the same level (all the VPs in the company) to the banal but much used (the company softball team). However, you can make sure that the groups follow a naming convention that enables users to find them in a logical place in the GAL.

Exchange 2013 includes the ability for an administrator to control two important aspects of group creation:

- Where in Active Directory new groups are created. This is set by running the Set-OrganizationConfig cmdlet (see the following).
- The naming policy applied to new groups when created by users or through EMS. EAC is used to create the group naming policy, but EAC will then ignore the policy because Exchange believes that an administrator knows what he's doing and is capable of deciding whether to follow the policy (if he can remember that a policy exists).

If you create a new group without setting a default location, the group is automatically placed in the Users OU. This OU has the advantage of being guaranteed to be present because it's one of the default OUs created when you install Active Directory. Few administrators delete OUs without good reason, but if you do remove the Users OU, users won't be able to create new groups with Outlook Web App. No restrictions exist on OU location when you create new groups through EAC or EMS because you can specify exactly which OU you want to use.

INSIDE OUT **You can define where a new group is located**

Administrators asked for the ability to control where new groups are created. You can define the location for all new groups created by users within the organization by updating the organization's configuration. To define where new groups are placed, use a command such as this:

```
Set-OrganizationConfig -DistributionGroupDefaultOU 'contoso.com/Exchange
Groups'
```

The specified OU must exist before you run this command. Exchange continues to create new groups in the default Users OU if you don't define another location. Existing groups are not automatically moved to the specified location after you create a naming policy, and EAC will not respect the policy on the basis that administrators should have the ability to place a new group in any OU they deem appropriate. If you find that groups have been created in less than optimal locations, you can use the Active Directory Users and Computers console to move the groups into the OU defined by policy. Make sure that moving the groups won't affect anything else before you do it; some applications might depend on finding groups they use stored in a particular OU.

You can only create a single OU for the organization, so this is very much a one-size-fits-all solution to the problem that works well for small deployments but isn't flexible enough for large organizations, which might want to distribute groups across a number of OUs. Stuffing thousands of groups into one OU is not always the best approach, but it's understandable that Microsoft took this approach because no one really wants to expose the Active Directory OU structure to users through EAC to allow them to select where a new group should be created. The upshot is that you might have to move groups around to put them into more appropriate OUs after they are created by users. Relocating a group to another OU does not affect the ability of group owners to manage it or other users to see it in the GAL.

Creating a group naming policy

No naming policy can completely prevent people from giving silly or inappropriate names to new groups. It would be unreasonable to expect such an intelligent policy as parsing group names to determine their worth because that would require a good deal of artificial intelligence and knowledge about company culture. One company's silly name is another company's code name for a breakthrough product. The intention behind the ability to create a group naming policy is to do no more than allow new groups to be clearly identified in the GAL through the application of a suitable prefix or suffix.

The naming policy applies to groups created by "normal" users through Outlook Web App; it is not applied to groups created by administrators through EAC or Outlook Web App. Exchange also applies the policy if an administrator creates a group by using EMS. However, if necessary, the policy can be ignored by specifying the IgnoreNamingPolicy parameter. For example:

```
New-DistributionGroup -Name 'Senior Executives' -Type Security
-IgnoreNamingPolicy -SAMAccountName 'SeniorExec' -OrganizationUnit
'contoso.com/Groups'
```

By default, Exchange 2013 does not allow users to create groups, and Outlook Web App therefore does not display the option to them to do so. If you are brave enough to allow users to create new groups, you must either amend the Default Role Assignment Policy and select MyDistributionGroups (Figure 6-10) or copy the Default Role Assignment Policy to create a new user role assignment policy that permits users to create groups and assign the new policy to the users you want to allow to create groups. This is the recommended option because it avoids making a change that potentially affects many users and allows for greater granularity over the control of group creation.

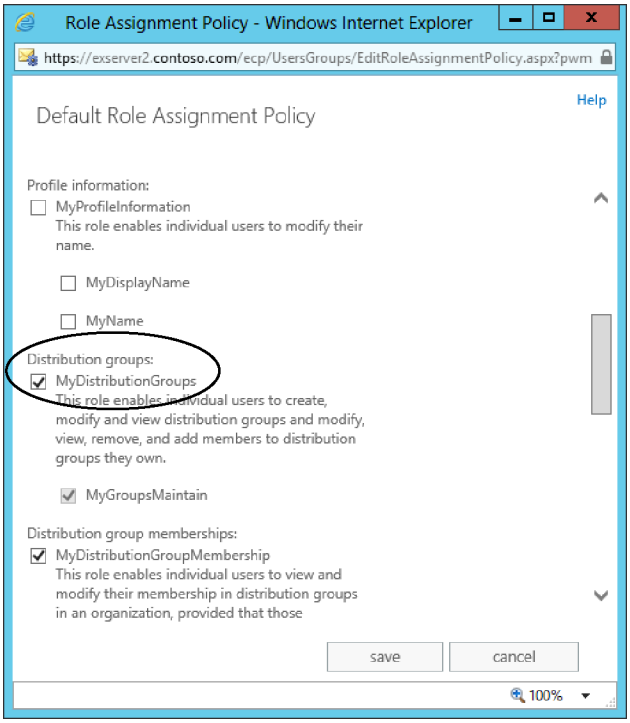


Figure 6-10 Updating the Default Role Assignment Policy to allow users to create groups

Note

A single group naming policy applies across the organization. You cannot have granular naming policies based on business structure or other filters. Remember that groups created by administrators are not governed by the group naming policy because Exchange applies the policy to user-created groups only.

In addition, there is no retrospective application of the naming policy to existing groups. If you want to make existing groups conform to the new standard, you have to update their display names by using EAC or EMS. Some scripting to identify all the offending groups and update their display names might come in handy here!

To create a group naming policy for the organization, open EAC, navigate to Recipients, and then choose Groups. The option to create the naming policy is under the list of groups. Details of the naming policy if it is already defined will be listed here. Click the ellipsis (...) and then choose Configure Group Naming Policy, and you are presented with a screen similar to that shown in the left side of Figure 6-11, where you can define which prefix or suffix should be applied to new groups. The right side shows how you can enter any words users are not allowed to include in group names. These words might include inappropriate words that are offensive or not suitable for business purposes.

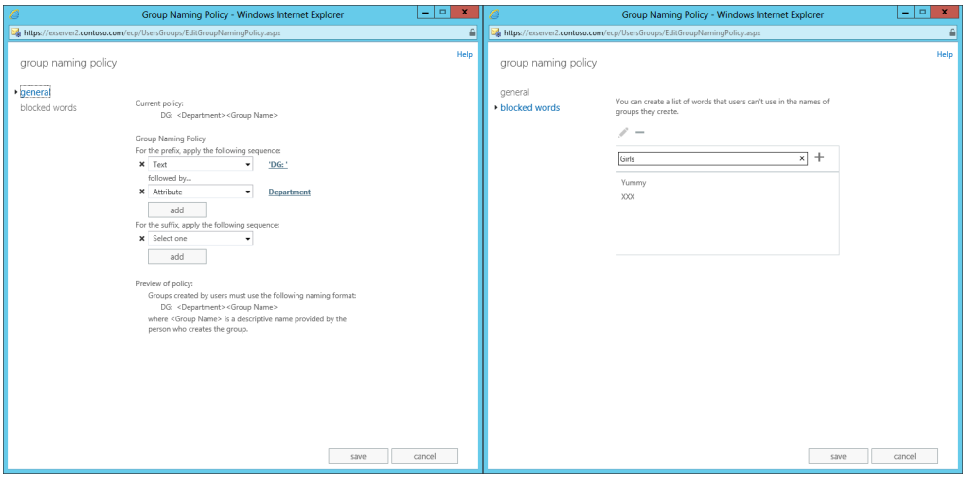


Figure 6-11 Defining a group naming policy for the organization

When you define the prefix or suffix, you can instruct Exchange to use a text string, an Active Directory attribute, or a mixture of both combined in whatever order you choose. In this context, the Active Directory attribute is determined from the account of the user

who creates the new group, and Exchange won't include the value in the group name if the attribute is not populated.

Prefixes are used most often to identify groups simply because it means that all groups are then listed together in the GAL. Figure 6-11 specifies that a prefix is created from two parts: a text string (DG:) followed an Active Directory attribute (Department). Thus, if the user specifies that the group name is Dublin Users and the value of his Department attribute is *Admin*, the resulting group name is DG: Admin Dublin Users. If the Department attribute is blank, the group name is DG: Dublin Users. EAC informs users about the effect of the naming policy, after they create new groups, with a pop-up message to tell them what the real name of the group will be.

When you have selected the right combination of text and Active Directory attributes to form group names, click Save to write the new policy into the organization settings. Exchange stores the group naming policy in the DistributionGroupNamingPolicy property of the organization configuration. You can access the properties that control how users create groups with the Get-OrganizationConfig cmdlet. For example:

```
Get-OrganizationConfig | Select Distr*
```

```
DistributionGroupDefaultOU  DistributionGroupNameBlockedWordsList  DistributionGroupNamingPolicy
--------------------------  ----------------------------------  ----------------------------
contoso.com/Exchange Users  {New, XXX, Chefs}                     DG <Department><GroupName>
```

Tip

It's possible to update the group naming policy with the Set-OrganizationConfig cmdlet, but it's easier and you are less likely to make a mistake if you update the policy with EAC.

From a computer point of view, group names that are generated according to a policy are useful because they ensure that groups are all arranged neatly in the GAL. From a human perspective, an administrator might still need to review and possibly amend group names slightly to make them more aesthetically pleasing. Of course, your users might be fully prepared to accept computer-friendly rather than user-friendly group names, and then you won't have to make any changes.

How users maintain groups

As explained in the "Creating a group naming policy" section earlier in this chapter, the default for Exchange 2013 is to enable users to view the groups to which they belong.

If you want to allow users to create new groups, you have to amend the Default Role Assignment Policy to set the check box for the MyDistributionGroups option. When you change the policy in this manner, Outlook Web App displays the necessary user interface to enable users to create new groups the next time they start the client. You know that the capability for users to create groups exists when the + (plus) sign is listed in the set of icons above Distribution Groups I Own (Figure 6-12). Otherwise, Outlook Web App displays only a list of the groups the user owns. The user can still edit the properties of the groups, including their membership, or even delete groups. She just can't create new groups. However, an administrator can create a new group on behalf of the user and then set the group's ownership to include the user.

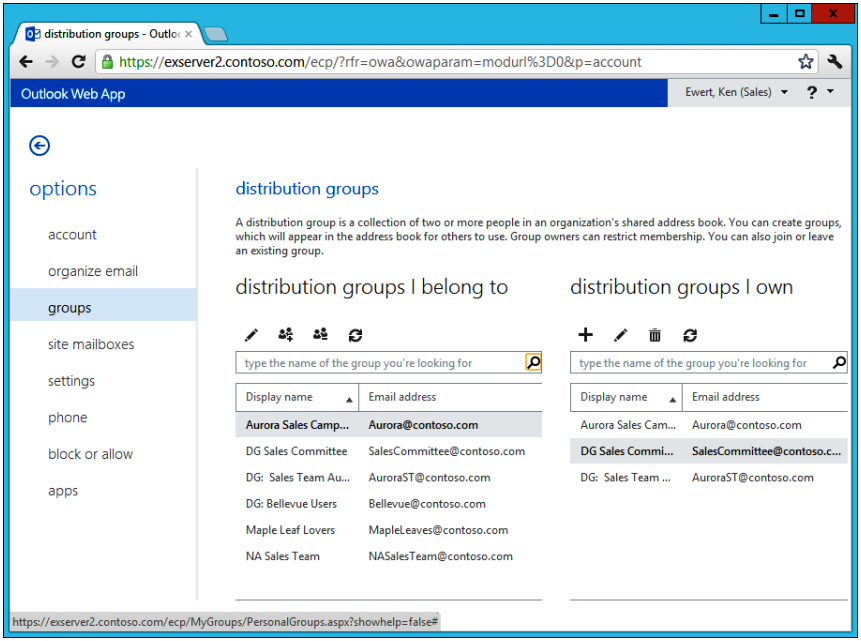


Figure 6-12 The user view of distribution groups presented by Outlook Web App

Depending on the membership approval rules set for the group, a user might be able to join a group or leave a group to which he already belongs. The search option enables users to browse the set of available groups in the GAL to see whether there are any groups they wish to join, even if the groups in which they are interested won't allow them to join because membership is closed or otherwise controlled. Obviously, groups that are hidden from the GAL are not shown in this list.

If a suitable group is found, the user can double-click its name to view the group's properties and then click Join to request membership of the group. Figure 6-13 shows the properties of a group. In this case, you can see that requests to join the group are automatically

accepted; anyone can join without further approval. Behind the scenes, this means that the group's MemberJoinRestriction property is set to Open. You can edit the properties of a distribution group with EAC to set these properties (on the Membership Approval tab) or use the Set-DistributionGroup cmdlet. For example:

```
Set-DistributionGroup –Identity 'NA Sales Team' –MemberJoinRestriction
'ApprovalRequired' –MemberDepartRestriction 'Open'
```

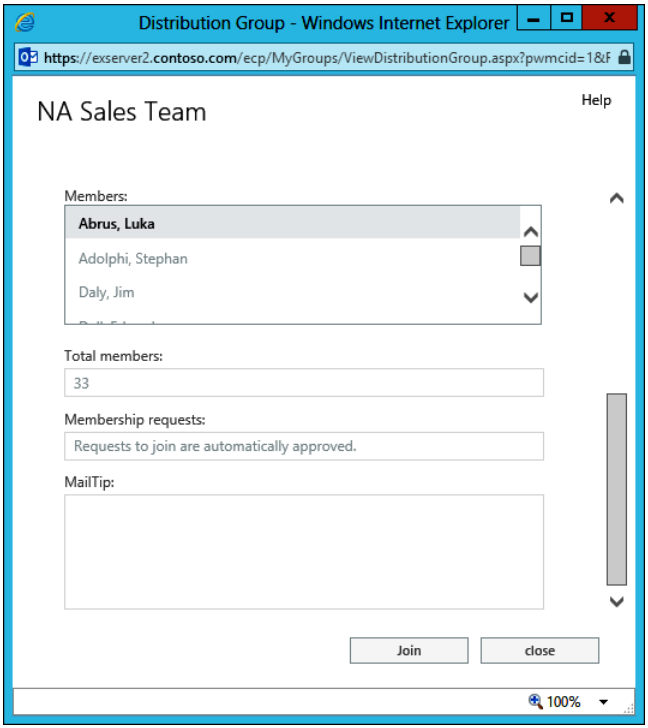


Figure 6-13 Viewing group properties

Joining an open group makes a user's membership effective immediately, and she will begin to receive any messages addressed to the group. It's obviously important to pay attention to the membership rules for any sensitive group. When you create a new group with EAC, the default is to leave both joining and leaving as Open. It's easy to miss this because the options are at the bottom of the screen used to enter the properties of the new group, and if you don't scroll down to the bottom or review the group properties either, you might leave a group open for all to join. Sometimes this wouldn't be a problem, but it could be for sensitive groups that relate to topics such as HR, compensation, employee relations, corporate strategy, and so on. For this reason, it's a good idea to ask group owners to review the membership of their groups regularly by viewing group

properties through Outlook Web App so that the right people have access to information through the group.

You can also use EMS to scan for groups that have open membership and decide whether any of these need to be restricted. For example, this command looks for any group that allows users to join without permission and outputs a simple list of the group name and owners. A review of the list will identify groups that might need to be restricted!

```
Get-DistributionGroup | Where {$_.MemberJoinRestriction -eq 'Open'} |
Format-List Name, ManagedBy
```

```
Name                        ManagedBy
--------                    ---------
IT Users                    {contoso.com/Users/Administrator}
Sales Committee             {contoso.com/Exchange users/Ken Ewert}
DG Marketing                {contoso.com/Users/Administrator}
Contracts staff             {contoso.com/Users/Administrator}
NA Sales Team               {contoso.com/Exchange users/David Pelton}
```

Restricting users to looking but not touching

The Default Role Assignment Policy is not granular enough to block users from creating new groups while allowing them to manage groups they own. However, you can create a new customized management role and add it to the Default Role Assignment Policy to replace the existing role assignment that allows users to create their own groups. The new management role is based on the old role, but you remove the cmdlet that allows users to create new groups so that all that remains are the cmdlets that allow users to update the existing groups they own. The procedure to create the new management role and add it to the Default Role Assignment Policy is as follows.

In the Permissions section of EAC, choose User Roles. Create a new user role policy and include all the role assignments that allow users access to different functionality, including MyDistributionGroups. You create a new user role policy rather than amending the default user role policy to provide the flexibility of assigning the ability to create new groups to some users but not others. In this example, you call the new user policy Restricted Group Management, but you can give the policy whatever name seems sensible to you.

1. Define a new customized management role based on the existing MyDistributionGroups role.

   ```
   New-ManagementRole -Name MyGroupsMaintain -Parent MyDistributionGroups
   ```

2. Remove the New-DistributionGroup cmdlet from the new customized management role you just created. The other cmdlets that interact with distribution groups such as

Add-DistributionGroupMember (to add a recipient to a group) are maintained in the role.

```
Remove-ManagementRoleEntry MyGroupsMaintain\New-DistributionGroup
-Confirm:$False
```

3. Remove the standard MyDistributionGroups role from the user role policy you created. The first command captures a pointer of the unwanted role assignment into a variable, and the second removes the role assignment from the user policy.

```
$OldAssignment = Get-ManagementRoleAssignment -RoleAssignee 'Restricted Group
Management' -Role 'MyDistributionGroups'
Remove-ManagementRoleAssignment $OldAssignment -Confirm:$False
```

4. Assign the new customized management role to the new user role policy you created. Essentially, your new role takes the place of the standard MyDistributionGroups role. The only difference between the two is that your new role doesn't allow users to access the New-DistributionGroup cmdlet, so they will never be able to create new groups.

```
New-ManagementRoleAssignment -Name 'MyGroupsMaintain-Restricted Group
Management' -Role MyGroupsMaintain -Policy 'Restricted Group Management'
```

5. Validate that the new role has been assigned to the user role policy. You'll notice that the MyDistributionGroups role assignment is no longer included in the policy.

```
Get-ManagementRoleAssignment -RoleAssignee 'Restricted Group Management' |
Select Name, Role
```

```
Name                                                        Role
----                                                        ----
MyContactInformation-Restricted Group management            MyContactInformation
MyProfileInformation-Restricted Group management            MyProfileInformation
MyDistributionGroupMembership-Restricted Group management   MyDistributionGroupMembership
My Custom Apps-Restricted Group management                  My Custom Apps
My Marketplace Apps-Restricted Group management             My Marketplace Apps
MyRetentionPolicies-Restricted Group management             MyRetentionPolicies
MyTextMessaging-Restricted Group management                 MyTextMessaging
MyVoiceMail-Restricted Group management                     MyVoiceMail
MyDiagnostics-Restricted Group management                   MyDiagnostics
MyTeamMailboxes-Restricted Group management                 MyTeamMailboxes
MyBaseOptions-Restricted Group management                   MyBaseOptions
MyGroupsMaintain-Default Role Assignment Policy             MyGroupsMaintain
```

6. Assign the new user role policy to the mailboxes for which you want to restrict the creation of new groups. Again, the only difference between this user role policy and the default provided by Exchange is that you have removed the role assignment that tells Outlook Web App to display the Create New Group icon associated with the New-DistributionGroup cmdlet:

```
Set-Mailbox -Identity 'JSmith' -RoleAssignmentPolicy 'Restricted Group
Management'
```

After you have verified that everything is in place, you can open Outlook Web App to see whether the change that has been made to the user role policy is effective. Figure 6-14 illustrates what you expect. The user can see all the groups he owns and can manage group membership, update group details, and even delete a group. However, if you compare this screen to the equivalent before the change was made (Figure 6-12), you'll see that the New option is not available, so the user cannot create a new group.

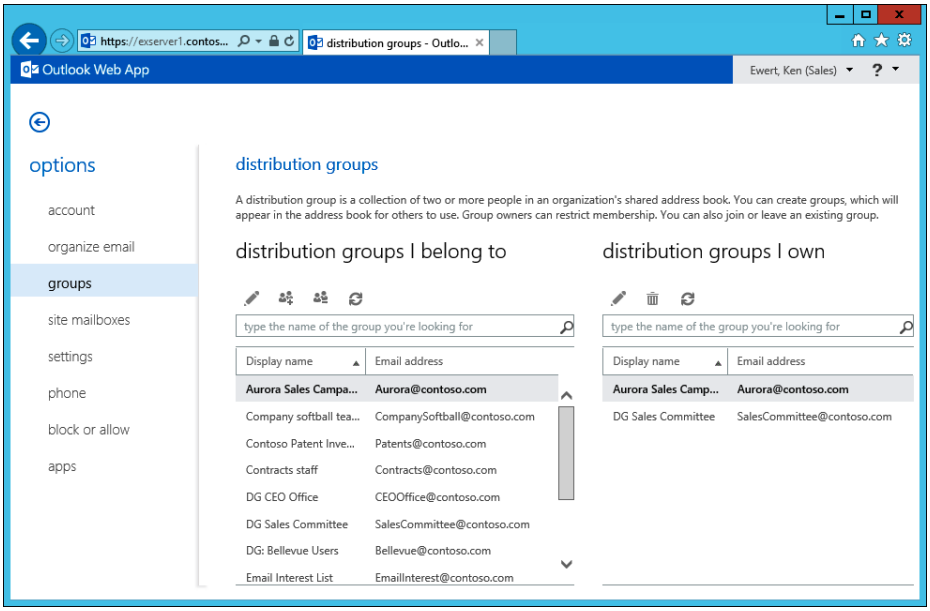


Figure 6-14 The effect of the customized role assignment policy

You might also want to remove the ability to delete groups, just in case a user makes a horrible mistake and gives you the opportunity to reconstruct a large distribution group. In this scenario, you can just remove the Remove-DistributionGroup cmdlet in the same way you removed the New-DistributionGroup cmdlet. (See step 2 in the process described earlier in this section.)

If you prefer not to grapple with the finer details of roles, assignments, and policies, Microsoft has published a script to do the work for you on the Exchange blog at *http://msexchangeteam.com/*. (Search for a reference to ManageGroupManagementRole.ps1.) You can grab the script and run it to make the necessary assignments.

Room lists

Managing or finding a suitable conference room for a meeting can be quite a hassle at times. Exchange attempts to ease the pain by introducing the concept of a room list. This is a distribution group whose membership is composed entirely of room mailboxes and is marked with a special property so that clients that currently support room lists can recognize these lists and incorporate them into the creation of new meetings. The intention is that you would create room lists to help users navigate through the full collection of rooms available within a company. This is more important in large companies that might have hundreds of rooms spread across multiple buildings in multiple locations. In such a scenario, you might create a room list for each building the company occupies. The feature is obviously less interesting for smaller companies that might have only one or two rooms to manage.

You can create a room list group in two ways:

- Create a distribution group as normal with EAC and then mark it as a room list with EMS.
- Create a distribution group with EMS and mark it as a room list when it is created.

To create a new room list group with EMS, you use the New-DistributionGroup cmdlet as normal and include the RoomList parameter. For example:

```
New-DistributionGroup -Name 'HQRooms' -OrganizationalUnit 'contoso.com/Groups'
-RoomList -DisplayName 'HQ Rooms'
```

To mark an existing group (one created with EAC), you use the Set-DistributionGroup cmdlet:

```
Set-DistributionGroup -Identity 'Building 2 Rooms' -RoomList
```

After you have created the room list groups, you can populate their membership with room mailboxes. If you attempt to add anything but a room mailbox to a room list, you get an error. To add new room mailboxes to the group, use the Add-DistributionGroupMember cmdlet:

```
Add-DistributionGroupMember -Identity 'Building 34' -Member 'Room 750'
```

EAC does not normally display room lists, so maintenance of their membership has to be done using EMS. In fact, the only time you will ever see room lists in EAC is if the group is in the cached data set when it is marked as a room list. However, even if it displays the room list, EAC won't let you add anything but a room mailbox to the group and will signal an error if you try. After it refreshes its cache (either when an administrator refreshes it manually or when the console is reloaded), the room list will be suppressed and won't appear in the list of groups.

Outlook 2010, Outlook 2013, and Outlook Web App can use room list groups when users set up meetings. In Outlook, the room list finder enables users to select a particular room list group (for instance, the room list group that contains the rooms for the building in which they want to hold the meeting). Outlook then loads the room mailboxes from the group and retrieves the free or busy data for the rooms to show the user when the rooms are free. The user can then select the most appropriate room and add it to the meeting. This feature does not override any restrictions the Resource Booking Attendant imposes (see the "Processing meeting requests according to policy" section in Chapter 5, "Mailbox management," for more information); it simply provides an easier way for users to locate the desired conference room for their meetings.

Figure 6-15 illustrates a room list group in action when scheduling a meeting by using Outlook Web App. You can see that the HQ Rooms list has been selected in the overlay screen. This screen is shown if you click Add A Room when creating a meeting, at which time you can select any available room list. After a room list is selected, Outlook Web App shows the available rooms with their current status (all free in this case), and the user can then select the room she'd like to reserve. The selected room is then added as a resource to the meeting.

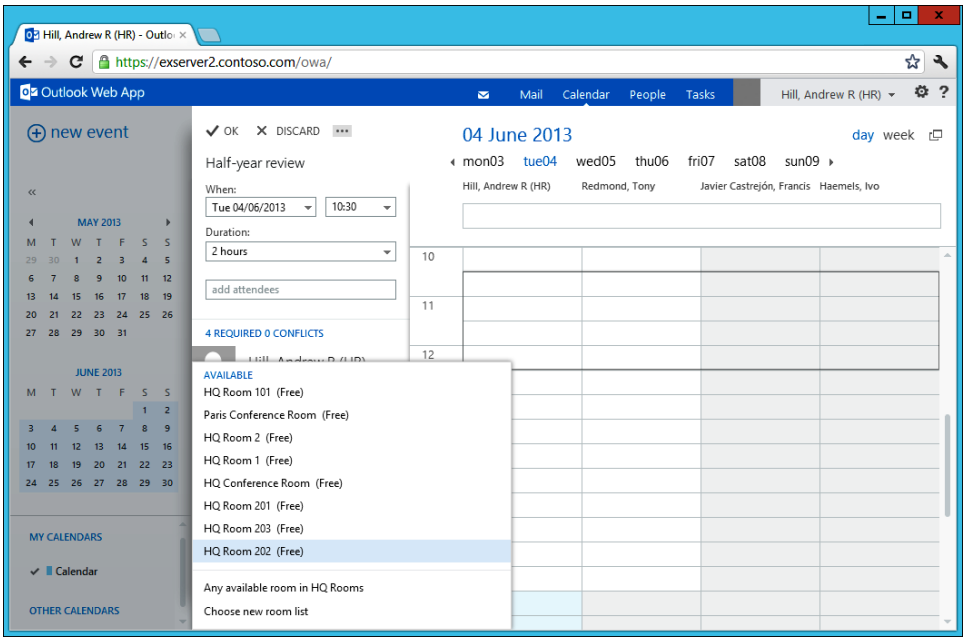


Figure 6-15 Using a room list with Outlook Web App

Dynamic distribution groups

Keeping track of people is generally difficult in large companies, which is why they invest so much money in HR systems. The same challenge exists for messaging administrators and help desk personnel, who can spend a lot of time maintaining the membership of groups as new people join the company, others leave, and others take up new roles and responsibilities. And as has just been discussed, populating and maintaining the membership of large distribution groups can be a nuisance for Exchange administrators.

Dynamic distribution groups offer a solution to this problem by allowing groups to be based on a query that Exchange can execute against Active Directory to determine the latest membership every time a message is sent to a group. The success and failure of dynamic distribution groups depends entirely on the accuracy of the data held in Active Directory. Success is defined as messages being delivered to the right people all the time, and you can expect success if the properties of mail-enabled objects are maintained in a consistent manner. However, if the data are inconsistent or not maintained, you can expect some interesting results when a Mailbox server expands the group to resolve the addresses for a message.

It's also fair to say that dynamic distribution groups generate a higher load on the transport system than normal groups. Instead of reading group membership when a message is routed, the transport system has additional processing to execute the query to determine group membership before a message can be routed. Although most queries are simple and execute quickly, especially on the high-powered server technology in use today, it's also possible to create queries that return thousands of recipients such as "All users in the United States," which will impose some strain as messages are processed. Even so, it's unlikely these groups will be used more than a few times per day, and this should not impose too high a performance penalty on any server.

From a user perspective, the only difference between a normal group and a dynamic group is that you cannot view the membership list of a dynamic group by looking at the group properties through the GAL. Dynamic groups don't hold security principals, so they cannot be used to secure objects. Apart from these minor differences, dynamic groups appear very similar to normal groups.

OPATH queries

Behind the scenes, every time a Mailbox server processes a message addressed to a dynamic distribution group, it uses the query stored in the group's properties to search Active Directory and expand the group membership into a list of addresses that match the query. Queries are stored in OPATH format.

A dynamic distribution group stores its query in the RecipientFilter (known as MSExchQueryFilter in Active Directory) property, where you'll find something like this:

```
((RecipientType -eq 'UserMailbox' -and Department -eq 'Sales') -and -not (Name -like
'SystemMailbox{*'))
```

This query looks for all mailboxes that have Sales as the value in their Department property. The last part of the query excludes system mailboxes. It is obvious that this query will work only if the value of the Department property is maintained. The query won't find mailboxes that belong to the department if the value is left blank or not spelled correctly. In other words, don't expect dynamic distribution lists to understand magically what Active Directory should contain. These queries operate on precise data.

Tip

Queries used for dynamic distribution groups are scoped to establish which Active Directory objects should be included when the query is evaluated. The scope is stored in the RecipientContainer property for the query. If you don't set a Recipient Container rule for the group, by default, EAC sets this property to the OU in which the new dynamic distribution group is created. This is very probably not the correct value for the group's scope unless you're in the habit of creating groups in the root container of Active Directory and intend for the group to be able to locate objects stored throughout the forest. It's important to establish the correct scope for a dynamic group because this has a dramatic effect on the objects the query returns and therefore who will receive messages addressed to the group. If the scope is set incorrectly, you might have no objects being found, and any message will go into a black hole. However, if the scope is set too broadly, any message could be delivered to tens of thousands of mailboxes when you really intended it to go to just a few.

If you're moving from Exchange Server 2003 to Exchange 2010 as an interim stage en route to Exchange 2013, you might have some of the older LDAP-based, query-based distribution groups. You can upgrade these groups to become dynamic distribution groups with the Set-DynamicDistributionGroup cmdlet by specifying the ForceUpgrade parameter. This upgrades the version of the group, makes it manageable by Exchange 2010, and translates the old LDAP query into OPATH format. Both LDAP and OPATH format queries are kept in the group properties to ensure backward compatibility and that Exchange 2003 servers understand the query. For example:

```
Set-DynamicDistributionGroup -Identity 'Old Exchange 2003 Users' -ForceUpgrade
```

It is possible that the cmdlet will be unable to translate the LDAP query into OPATH, especially if you built a highly customized LDAP query to use with Exchange Server 2003. In this

instance, Exchange flags an error, and you will have to create a new dynamic distribution group to use with Exchange 2010 and replace the older group. After the group is created in the new format, it is acceptable to Exchange 2013.

Creating new dynamic distribution groups

Before starting to create a new dynamic distribution group, take the time to discover three key characteristics of the new group:

- **The scope for the query** Active Directory can only return recipients based on the scope of the query you use. The scope is established by an OU and its children within a single domain; it cannot span multiple OUs. If the recipients you want to find are in multiple OUs (or domains), you must create separate dynamic distribution groups for each OU and then combine the groups into one supergroup. Remember that EAC uses the OU in which you create the new group as its default scope and that you might need to adjust the RecipientContainer property of the group afterward. You can do this with EMS only.
- **The recipients** You can include all types of mail-enabled recipients or just a particular type, such as recipients that have Exchange mailboxes.
- **Rules for selection** Precoded queries enable you to filter against the following Active Directory properties:
  - State or Province (StateOrProvince)
  - Department
  - Company
  - Recipient Container (to set the scope for the objects found by the query)
  - Custom Attributes (CustomAttribute1 through CustomAttribute15)

Dynamic distribution groups are created through the Groups section of EAC. To begin, click the down arrow beside the + (plus) sign and select Dynamic Distribution Group. Just like a regular group, you have to provide the name and alias for the group, the name of its owner, and the OU in which it will be stored. The bottom part of the screen (Figure 6-16) reveals some detail of the query that underpins the new group.

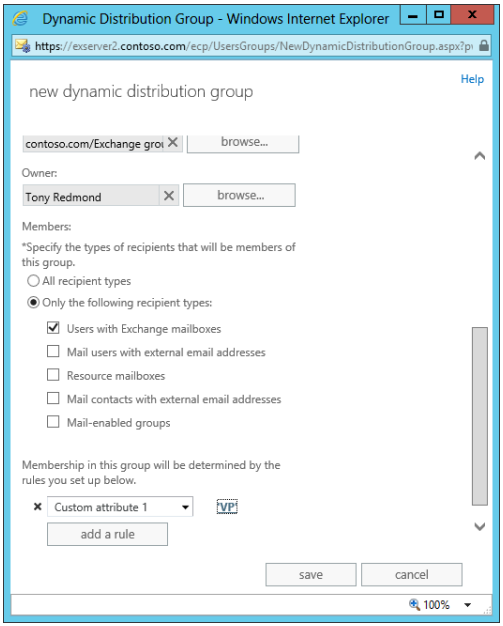


Figure 6-16 Creating a dynamic distribution group

As explained earlier, the query is broken up into the types of recipients and the rules used to filter the objects in Active Directory. In this instance, you've selected that only mailboxes are included in the group and have begun to input the first rule, which is based on Custom Attribute 1. The set of 15 custom attributes supported by Exchange enables a lot of flexibility in selecting recipients, provided the attributes are populated correctly.

Figure 6-17 shows how to build a rule. In this instance, your example group is intended to address all users who hold the Vice President title. Exchange doesn't allow precoded queries against the title attribute and, anyway, user titles are often free-format and so vary widely; your query would have to accommodate titles such as Vice President (Sales) and VP-APJ. It's much easier and usually more accurate to base queries on data that administrators maintain and follow a tight format. In this case, you want to look for mailboxes that have VP stored in Custom Attribute 1, so that's what you enter.

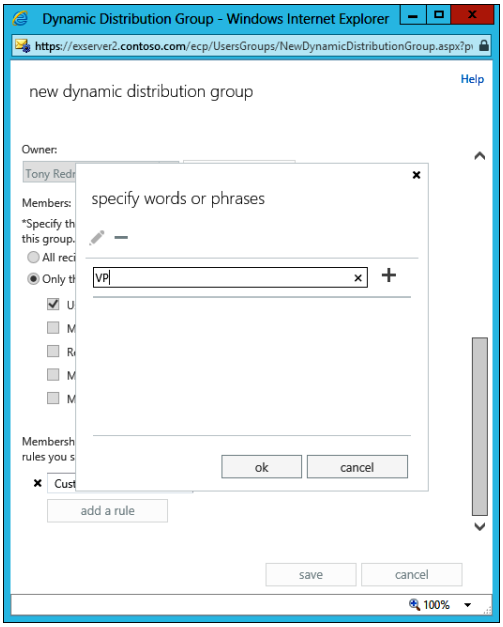


Figure 6-17 Specifying the query for a dynamic distribution group

You can specify multiple values for the properties used as rules in dynamic distribution groups. For example, you can look for users in the Sales and Marketing departments that work in the Coho or Coho Winery company. When a dynamic group is created, Exchange stores the query in both OPATH and LDAP format. You can see these queries with the Get-DynamicDistributionGroup cmdlet. For example:

```
Get-DynamicDistributionGroup -Identity "Sales Professionals" | Format-List *Filter*
```

Both precoded and custom OPATH queries can be used for dynamic distribution groups. Dynamic groups created with EAC use precoded queries, so these are the most common type you will encounter. They are also the most limited because they are based on restricted criteria. The advantages of precoded queries are that Exchange can build the query quickly and that you can have a high degree of confidence that the query will operate properly when you create groups with EAC. Behind the scenes, EAC builds the OPATH query for you with the criteria provided for the group. Because they are based on a limited set of criteria, precoded queries are tuned for performance.

Custom queries are the most flexible because you essentially roll your own OPATH query and add it to the group through EMS, a topic examined later in this section. However, custom queries are harder to build and maintain and usually run a little slower than precoded queries.

Note

To make a dynamic distribution group work, you have to provide values for Exchange to check against. This can become frustrating when you want to create geographic-based groups, such as one based on everyone who is located in Ireland. It's a little irritating that you can't use the Country property for this purpose because it seems likely that this property would be one of the more commonly used. Microsoft didn't select it for inclusion in the set of properties supported for dynamic group rules, so the most practical approach is to use the State or Province property instead. Of course, for the group to be successful, the State or Province property must be populated with the values for which you are checking.

Validating query results

EAC is less user-friendly when creating new dynamic distribution groups than the wizard used in Exchange 2007 and Exchange 2010. Apart from not showing the full list of available mailbox properties that could be available to use in queries, it lacks the Preview function that was so valuable because it enables you to test the effectiveness of the filter you've just created, checks that you've specified the correct parameters for the query, and verifies that Active Directory is populated with sufficient information to make the query useful. No equivalent to the Get-DistributionGroupMember cmdlet is available to list the membership of dynamic distribution groups. The only way to check a recipient filter with Exchange 2013 is to execute it as a filter for the Get-Recipient cmdlet in EMS, which is definitely not as user-friendly as being able to click a Preview button.

You can use the following commands to test a query with EMS. These extract the recipient filter and the recipient container (scope) from the group and then use these values as input to the Get-Recipient cmdlet. Before running any test, it's wise to know what represents success—in this case, the set of Active Directory objects you expect the query to return. If you expect EMS to return a lot of data, it is more convenient to pipe the output of the Get-Recipient command to a text file rather than attempting to check information on screen.

```
$Filter = (Get-DynamicDistributionGroup -Identity "Sales Professionals")
.RecipientFilter

$Scope = (Get-DynamicDistributionGroup -Identity "Sales Professionals")
.RecipientContainer

Get-Recipient -RecipientPreviewFilter $Filter -OrganizationUnit $Scope
```

The Get-Recipient command might not result in the expected set of users. If it doesn't, an incorrect scope is the likely cause and is probably due to a failure to specify a value for the Recipient Container rule in the query. Before adjusting the scope, you can run a modified

version of the same command to check what Active Directory returns if the scope is unlimited. This checks that the properties the query uses are actually populated!

```
Get-Recipient -RecipientPreviewFilter $Filter
```

If Active Directory returns a different set of results, you know that the scope or query is incorrect. However, if Active Directory returns nothing at all, the immediate suspicion is that the query is being resolved against an OU that does not contain the objects you want to locate. Of course, the data could be inaccurate also, and a wide range of different results might be obtained if properties are not populated correctly or if the objects you want to find are scattered in different OUs. The only way to debug a recalcitrant query is to be persistent and know what data you want to find.

If you discover that you need to adjust the scope for the query, you can do this with the Set-DynamicDistributionGroup cmdlet. For example, the following command sets the scope to the entire contoso.com forest:

```
Set-DynamicDistributionGroup -Identity "Sales Professionals" -RecipientContainer
"contoso.com"
```

After making any change to a query's scope, you should validate that the new scope returns the correct results by comparing them to the expected data.

Tip

It's a good idea to check the list of recipients found by the filter carefully to determine that the right recipients are present and that the list doesn't include a recipient that shouldn't be there. After bad or missing Active Directory data, the most common problem in finding recipients by using filters is an incorrect scope. Remember, the filter Exchange generates will start at a certain OU in Active Directory and find objects in that point and child OU. It won't find objects elsewhere.

Creating dynamic groups with EMS

The basics of creating a new dynamic distribution group with EMS are very straightforward. The New-DynamicDistributionGroup cmdlet is run to create the new group and specify the various parameters, including the query. Normally, it is best to create any group you can through EAC and go to EMS only when you need to create a group that uses a custom filter, which will be covered later in this section. In the meantime, a basic dynamic group might be created as follows:

```
New-DynamicDistributionGroup -Name "Contoso Ireland" -DisplayName "Contoso Ireland
Employees (QBDG)" -Alias "ContosoIRL"
-OrganizationalUnit "contoso.com/Groups" -RecipientContainer "contoso.com"
```

```
-IncludedRecipients "MailboxUsers" -ConditionalStateOrProvince "Ireland",
"Dublin" -ConditionalCompany "Contoso", "Contoso Ireland"
-ConditionalCustomAttribute10 "Employee"
```

Separately, these elements are:

- **Name** The name of the group as used in Active Directory. Users never see this name.
- **DisplayName** The name of the group as shown in the GAL. Because users do see this name, it is wise to make it indicative of how the group is intended to be used. In this instance, the (QBDG) suffix is simply to tell users that this group is based on a query, so they know they should not bother to look up its membership through the GAL.
- **Alias** A unique name that can be used to work with the group.
- **OrganizationalUnit** The Active Directory OU in which the group is to be created. The OU must exist or the command will fail.
- **RecipientContainer** The scope of the query. In this instance, the scope is set to contoso.com; all objects in the contoso.com forest fall within the scope of the query.
- **IncludedRecipients** The mail-enabled objects the query should locate. In this case, you just want to address user mailboxes, so that's what you specify.
- **ConditionalStateOrProvince** A special parameter that accepts a value that will be evaluated against the StateOrProvince attribute of Active Directory objects.
- **ConditionalCompany** A special parameter that accepts a value that will be evaluated against the Company attribute of Active Directory objects.
- **ConditionalCustomAttribute10** One of 15 similar parameters that can be used to search against the 15 custom attributes Exchange supports. In this instance, you want to search against CustomAttribute10.

The next section discusses the use of the conditional parameters.

The full meaning of the query in this group is:

- Search all of Contoso.com.
- Find user mailboxes that satisfy the following criteria:
  - The value of their StateOrProvince attribute is either *Ireland* or *Dublin*.
  - The value of their Company attribute is *Contoso*.
  - The value of their CustomAttribute10 attribute is *Employee*.

Or, in plain English, find all mailboxes belonging to Contoso employees who work in Ireland.

Largely because the maintenance of membership is not required, group ownership is not as important for dynamic groups as it is for regular groups. However, it is still possible to nominate owners for a dynamic group and to apply the same moderation controls as you can use with normal groups. These values are set after the group is created. For example:

```
Set-DynamicDistributionGroup –Identity "Contoso Ireland" –ManagedBy "Tony Redmond"
–ModeratedEnabled $True
–ModeratedBy "Tony Redmond" –MailTip "Messages sent to the Contoso Ireland group need
to be approved before distribution"
```

Now that you know the basics of how to create a dynamic group with EMS, we can move on to some more interesting queries.

Using custom filters

A custom filter is one that is built without using the precoded parameters. You cannot create a dynamic distribution group that is based on a custom filter through EAC. Instead, you use the New-DynamicDistributionGroup cmdlet in EMS to create the new group and specify the OPATH query you want to use. For example, here is the one-line command to create a group that contains a recipient filter that includes anyone who works for the Contoso company:

```
New-DynamicDistributionGroup -Alias 'AllContoso' -DisplayName 'Contoso Company Users
(QBDG)'
–IncludedRecipients 'MailboxUsers' -RecipientContainer 'contoso.com' –Name 'All
Contoso'
-OrganizationalUnit 'contoso.com/Groups'
```

In fact, this is not a very complex filter, and EAC could easily have generated it. Note that the RecipientContainer parameter specifies the scope of the objects that form part of the dynamic group based on their place within Active Directory. In this case, you specify the root of the contoso.com domain; any object under this point is included. After creating the new dynamic distribution group, you can determine whether the filter is precoded or custom with the Get-DynamicDistributionGroup command:

```
Get-DynamicDistributionGroup –Identity 'AllContoso' | Format-List DisplayName,
RecipientFilterType, RecipientFilter, LDAPRecipientFilter
```

```
DisplayName              : Contoso Company Users
RecipientFilterType      : Precanned
RecipientFilter          : ((RecipientType -eq 'UserMailbox') -and (-not(Name -like
     'SystemMailbox{*')) -and (-not(Name -like 'CAS_{*')) -and (-not(RecipientTypeDetailsValue
```

```
     -eq 'MailboxPlan')) -and  (-not(RecipientTypeDetailsValue -eq 'DiscoveryMailbox')) -and
     (-not(RecipientTypeDetailsValue -eq 'PublicFolderMailbox')) -and
     (-not(RecipientTypeDetailsValue -eq 'ArbitrationMailbox')))
LdapRecipientFilter      : (&(!(!(objectClass=user)))(objectCategory=person)
     (mailNickname=*)(msExchHomeServerName=*)(!(name=SystemMailbox{*))
     (!(name=CAS_{*))(!(msExchRecipientTypeDetails=16777216))
     (!(msExchRecipientTypeDetails=536870912))
     (!(msExchRecipientTypeDetails=68719476736))(!(msExchRecipientTypeDetails=8388608)))
```

Even though this dynamic distribution group was created through EMS, it looks like a precoded query, very similar to the filters EAC generates. The reason is that Exchange optimizes filters as much as possible when they are created. When Exchange examined the query specified through EMS, it decided that this filter is really a precoded filter because it does not use any properties outside the set of conditional properties on which precoded filters are based. As you will recall from the earlier discussion, the conditional properties that are available for filtering when you create dynamic distribution groups are as follows:

- ConditionalDepartment
- ConditionalCompany
- ConditionalStateOrProvince
- The 15 custom attributes (*ConditionalCustomAttribute1 to ConditionalCustomAttribute15*)

Why Exchange provides special conditional properties

The reason this set of special conditional properties exists is very simple. A dynamic distribution group is a recipient object. You can set properties for recipient objects, including the properties you might want to use when you create a filter. The potential for confusion when setting properties of the group itself and the properties used by the filter that generates the group membership is apparent, so the Exchange developers added the special filtering properties to make it obvious when you want to set a property of the group and when you want to set the value of a property used in a filter.

If the set of special filter properties doesn't suffice for the filter you want to create, you can create a custom filter by using other properties of recipient objects and store it in the RecipientFilter property. Note that if you create a custom filter, you cannot combine it with

a precoded filter. The rules of thumb for building filters for dynamic distribution groups are therefore as follows:

- Precoded filters are easier to build, but you are limited to a restricted set of special conditional properties. For example:

  ```
  New-DynamicDistributionGroup –Identity 'ContosoUK' –ConditionalStateOrProvince
  'England', 'Wales', 'Scotland'
  –ConditionalCompany 'Contoso', 'Contoso UK'  -ConditionalCustomAttribute10
  'Employee' –DisplayName 'Contoso UK (QBDG)' –Name 'Contoso UK'
  -OrganizationalUnit 'contoso.com/Groups'   -RecipientContainer 'contoso.com'
  –IncludedRecipients 'MailboxUsers'
  ```

  In this example, the filter Exchange uses to build the dynamic group is "Any account from England, Wales, or Scotland that belongs to the company called Contoso or Contoso UK and that has the value *Employee* in Custom Attribute 10." Any filter that has to check multiple values for a property is evidence of a certain lack of standards in the population of mailbox data. Remember the caveat that although dynamic distribution groups are great ways to address changing populations, they can be slow for the transport service to resolve against Active Directory if more than a couple of hundred addresses result or you add many conditions. This is when simpler is better.

- As shown in the previous example, you can restrict a dynamic distribution group to send messages to a certain set of recipient types by specifying the –IncludedRecipients parameter. You can then select from one or more of the following values: AllRecipients, MailboxUsers, Resources, MailContacts, MailGroups, MailUsers, or even None (which would be a kind of non-op).

  If a precoded filter doesn't suffice, you can create a filter query based on a broader range of recipient properties and store it in the –RecipientFilter property. You can combine properties to create a filter that is as complex as you like. For example:

  ```
  -RecipientFilter {(Company –eq 'Contoso') –and (Office –eq 'Dublin') –and
  (Department -eq 'Sales') –and (CustomAttribute10 –eq 'Employee')}
  ```

- You cannot combine precoded and custom filters in a dynamic distribution group.

For more information about the properties that are supported for RecipientFilter, see *http://technet.microsoft.com/en-us/library/bb738157.aspx*.

A custom filter to address mailboxes in a database

To illustrate the point, let's create a dynamic distribution group that is useful to administrators. In older versions of Exchange, it was common to manage users on a server basis, so it's a good idea to have a dynamic group that sends messages to everyone whose mailbox is on a specific server so that you can advise them about important issues such as server outages for planned maintenance. The command to create a new dynamic distribution group

with a recipient filter that includes all the mailboxes on an Exchange server is shown next. You have to use a custom filter in this case because ServerName is not one of the set of special properties supported by a precoded filter.

```
New-DynamicDistributionGroup -Name 'Mailboxes on ExServer1' -Alias MbxSvr1
-RecipientContainer 'contoso.com'
-OrganizationalUnit Groups -RecipientFilter {ServerName -eq 'ExServer1'}
–IncludedRecipients 'MailboxUsers'
```

This code works superbly for Exchange 2007, but you'll have spotted that you cannot predict which Exchange 2010 or Exchange 2013 mailbox server hosts the active copy of a mailbox because the connection between database and server is broken when Database Availability Groups (DAGs) were introduced. The dynamic group created with the preceding code still works on Exchange 2010 and Exchange 2013, but there is no guarantee that you can communicate with the desired target audience if the database that holds its mailboxes has moved to a different server.

Therefore, you need to create a dynamic group that addresses messages to all the mailboxes in a database rather than one that addresses all the mailboxes on a server. This is slightly more difficult because Exchange requires you to pass the full distinguished name of the target mailbox database as part of the filter rather than a simple display name. The logic is probably that the fully distinguished name ensures that you always use the right database.

The first thing you need to do is get the distinguished name of the database. This is easily done with:

```
$DB = (Get-MailboxDatabase –Identity 'VIP Mailboxes').DistinguishedName
```

As you can see, the distinguished name is put in a variable to make it easier to reuse. Of course, you can just take down the full name and retype it into the command to create the new dynamic group. This will also work, but you have to be very careful to type in the exact name without any additional spaces, which can be difficult with such a large and complex name:

```
CN=VIP Mailboxes,CN=Databases,CN=Exchange Administrative Group (FYDIBOHF23SPDLT),CN
=Administrative Groups,CN=contoso,CN=Microsoft Exchange,CN=Services,CN=Configuration,
DC=contoso,DC=com
```

After you have the distinguished name safely stored, you can create the new dynamic group to address mailboxes in the VIP Mailboxes database as follows:

```
New-DynamicDistributionGroup –Name 'VIP Mailboxes' –Alias 'VIPList'   -DisplayName
"VIP Mailboxes (QBDG)"
–RecipientContainer 'contoso.com' –OrganizationalUnit 'Exchange Groups'
–RecipientFilter "Database –eq '$DB' –and RecipientTypeDetails –eq 'UserMailbox'"
```

You can see that the filter used contains two conditions. The first is the name of the database as stored in the variable you created. The second specifies that you're interested only in user mailboxes because these are the only type to which anyone would ever want to send messages. If you omit the second condition, the list returned will include objects such as room and resource mailboxes. The resulting filter Exchange generates when it creates the new dynamic distribution group is as follows. Clearly, it's more complex than the command that was entered, but in fact it's simply because Exchange expanded the variable to the fully distinguished name for the database and included some additional conditions to filter out the objects you wouldn't want to address. Exchange automatically suppresses objects such as arbitration mailboxes when you specify the name of a database as a recipient filter.

```
((((Database -eq 'CN=VIP Mailboxes,CN=Databases,CN=Exchange Administrative Group
(FYDIBOHF23SPDLT),CN=Administrative Groups,CN=contoso,CN=Microsoft Exchange,CN
=Services,CN=Configuration,DC=contoso,DC=com') -and (RecipientTypeDetails -eq
'UserMailbox'))) -and (-not(Name -like 'SystemMailbox{*')) -and
(-not(Name -like 'CAS_{*')) -and (-not(RecipientTypeDetailsValue
-eq 'MailboxPlan')) -and (-not(RecipientTypeDetailsValue -eq 'DiscoveryMailbox'))
-and (-not(RecipientTypeDetailsValue -eq 'PublicFolderMailbox')) -and
(-not(RecipientTypeDetailsValue
-eq 'ArbitrationMailbox')))
```

Before you create the new group, you can test the effectiveness of the filter by passing it to the Get-Recipient cmdlet. The basic idea is that if the filter works with Get-Recipient, it will work as a custom recipient filter for a dynamic group. For example:

```
Get-Recipient –RecipientPreviewFilter "Database –eq '$DB' –and RecipientType –eq
'UserMailbox'"
```

To check that these are indeed the correct mailboxes, use the Get-Mailbox command:

```
Get-Mailbox –Database –Identity "VIP Mailboxes"
```

Hopefully, the sets of objects returned by the two commands match. If they do, take note of the results and then create the new dynamic group. You can then compare the original results against the set returned by the filter contained in the dynamic group by extracting the recipient filter from the group and using it with the Get-Recipient command. For example:

```
$Filter = (Get-DynamicDistributionGroup –Identity "VIP Mailboxes").RecipientFilter
Get-Recipient –RecipientPreviewFilter $Filter
```

If all has gone well, the sets of objects match, and you have a working dynamic group containing a custom filter that retrieves the mailboxes from the selected database.

INSIDE OUT **Changes to recipient types for custom filters**

You cannot use EAC to edit a custom recipient filter created for a dynamic distribution group, and if you attempt to do so, you run the risk of overwriting the custom query and rendering the group inoperative. The only way to edit a custom filter is to create a new recipient filter with EMS and update the group with the Set-DynamicDistributionGroup cmdlet.

Sharing

Sharing policies were introduced in Exchange 2010 to provide a mechanism for Exchange to control how users share data such as calendar free and busy information between organizations. Federation between two Exchange organizations requires either the on-site deployment of the Microsoft Federation Gateway, which acts as a trusted broker between the two organizations to ensure that data can pass between them in a safe and secure manner, or access to the Microsoft online version of the Microsoft Federation Gateway.

The Microsoft Federation Gateway is part of the Active Directory Federation Services role that can be installed on a Windows 2008 or Windows 2012 server. In most cases, it's easier to use the online variant unless you have compelling reasons to use your own internal gateway. In either case, you can use the Sharing option under the Organization section in EAC or the New-FederationTrust cmdlet to establish the necessary connection with another organization. Detailed steps describing how to use these options are provided in the Exchange team's blog post, *http://blogs.technet.com/b/exchange/archive/2012/10/30/managing-federated-sharing-with-the-eac.aspx*.

See *http://technet.microsoft.com/en-us/library/dd727938(WS.10).aspx* for more information about the deployment of Active Directory Federation Services.

Certificate management

X.509 certificates are important to Exchange because they underpin the Secure Sockets Layer (SSL)–based secure communications between clients and servers, including SMTP conversations (secured with Transport Layer Security [TLS]) to transfer email between servers. Certificates can also be used for federation between Exchange organizations. The

certificates authenticate that their holders are who they claim to be and are used to create the secure channels that transfer data. Exchange 2013 supports three kinds of certificates:

- Self-signed certificates A self-signed certificate is generated by the application that wishes to use it. It is not signed by a certificate authority (CA), and other applications or computers might not trust it because it doesn't have a certificate path that includes a trusted CA.
- Certificates issued by a Windows CA (Windows Certificate Services).
- Certificates issued by a commercial SSL vendor such as GoDaddy, Thawte, or VeriSign.

Table 6-1 summarizes the different protocols Exchange uses to communicate and the types of certificates that can be used to secure these connections. As you can see, third-party certificates are most useful and are an absolute requirement to secure external communications. Certificates are also used to secure communications between a reverse proxy server and Exchange if you deploy such a server in the perimeter network for external communications.

TABLE 6-1 Protocols and required certificates

Server role	Protocols requiring certificate	Required level of certificates
Mailbox	Outlook Web App Exchange Web Services Outlook Anywhere ActiveSync POP3 and IMAP4 Autodiscover	Third-party (recommended) or Windows public key infrastructure (PKI). Include the FQDN of the server and the Uniform Resource Locators (URLs) for applications such as Outlook Anywhere, Outlook Web App, and Office Communications Server.
Mailbox	Outlook Web App (HTTPS)	Self-signed
Unified Messaging	None	Self-signed. However, a CA-issued certificate is required to enable UM interoperability with Lync.
Transport (within mailbox and CAS)	SMTP over TLS	Third-party (recommended) or Windows PKI for external; self-signed for internal server-to -communications. Include the FQDN of the server and the domain name.
Edge transport	SMTP over TLS	Third-party (recommended) or Windows PKI. Include the FQDN of the server and the domain name.

Windows Certificate Services provides a public key infrastructure (PKI) to enable organizations to publish their own certificates. The certificates are more manageable because a complete Windows PKI is available to control the issuing, renewal, and revocation of certificates. Before they can be used to enable secure connections, these certificates also have to be installed in the trusted root certificate store of computers that are not part of Active Directory that need to communicate with Exchange. Windows PKI certificates are more manageable than self-signed certificates and are also free, so they are an acceptable solution for small to medium deployments.

From this discussion, you can see that the best solution for almost all deployments is to use a commercial certificate bought from a reputable third-party trusted CA that takes the responsibility to issue and ensure the validity of certificates. These certificates are more expensive, but they offer a major advantage in that the issuer's CA certificates are usually already installed in the trusted root certificate store of client computers, so you don't have to install certificates manually before devices can connect.

Exchange 2010 provided the first management interface for certificates. Before then, certificate management was intensely painful because it had to be done using Windows PowerShell. This isn't a problem if you're used to the nomenclature and parameters used with certificates, but can be quite off-putting if you don't work with certificates often. Thankfully, the user interface (UI) EAC presents makes the task pretty straightforward because you can view the certificates assigned to servers, create new certificates, and assign the certificates to Exchange services (Outlook Web App, ActiveSync, and so on). As shown in Figure 6-18, certificates are managed in the Servers section of EAC. In this case, you can see the three self-signed certificates that are automatically installed on a multirole server during installation. The selected certificate is assigned to the IMAP, POP, Internet Information Services (IIS), and SMTP protocols.

Clicking the selected certificate exposes its properties (Figure 6-19). Clicking Services at this point exposes the list of services with which you can associate a certificate. However, because this is one of the self-signed certificates Exchange creates for its own use, it's best to leave its properties intact; any change might affect something such as the ability of a Mailbox server to send messages by SMTP.

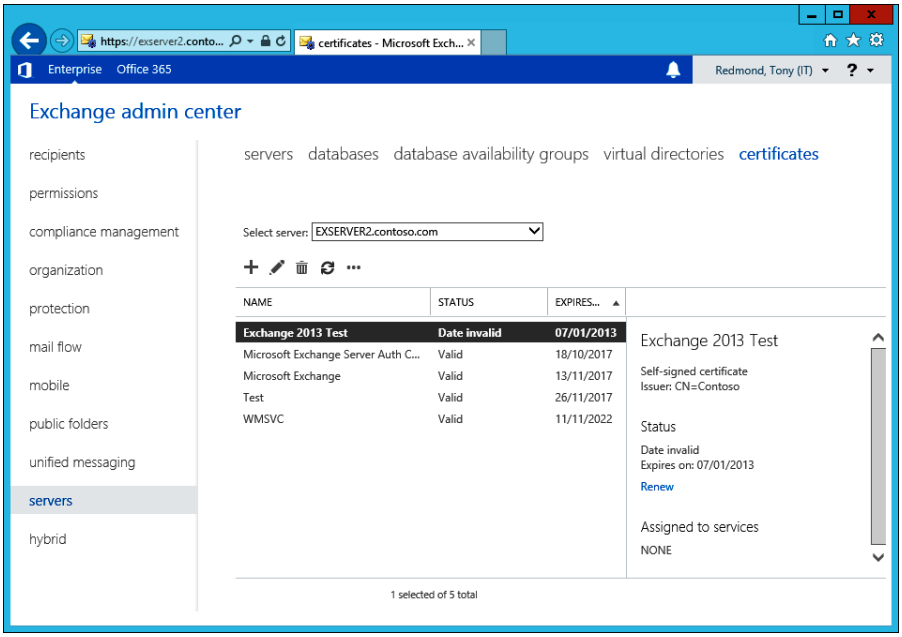


Figure 6-18 Viewing the certificates installed on a server

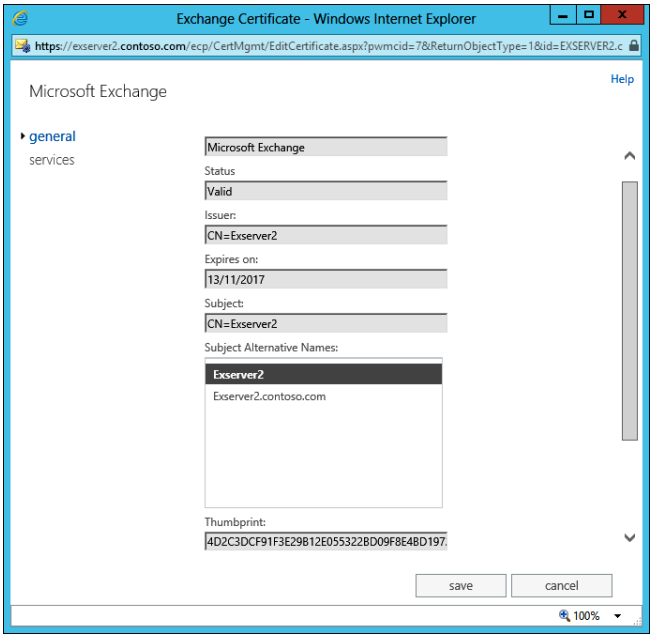


Figure 6-19 Viewing properties of a certificate

Several options are available to work with certificates. You can do any of the following:

- **Request a new Exchange certificate** This option enables you to create a request that is later sent to a Windows or commercial third-party CA that generates the necessary certificate to meet the requirements the wizard gathers. The certificate is brought into Exchange with the Complete Pending Request option, in which you give Exchange the name of the certificate file the CA generates so it can import the certificate.

- **Create a new self-signed certificate for use with Exchange** For example, because the self-signed certificates Exchange creates last for five years from the date the server was installed, you might have to create a new set of self-signed certificates when these certificates expire.

- **Import an Exchange certificate** Use this option when a company has general certificates that are used for more services than Exchange. You provide the name of the file containing the certificate and its password, and Exchange imports it. Afterward, the new certificate can be assigned to whatever services it is designed to support.

- **Export an Exchange certificate** This option enables you to export an Exchange certificate to a password-protected file.

If you want to use a commercial certificate to protect Exchange services, you must first create a request in the form of an REQ file (PKCS #10 format) that you send to the CA. The request provides the CA with all the information necessary to issue a new certificate. To create a new request, click the + (plus) sign to invoke a multiscreen dialog box in which you can enter the necessary details for a new certificate that can be used by the Exchange services. After all the details are entered, EAC generates a certificate request in the form of an encoded REQ file that you can then send to a CA. After the CA provides the new certificate in the form of a CER file, it can be imported into Exchange with the Import Exchange Certificate option and then assigned to the services that need to use it.

Planning for the deployment and use of certificates is a complex task that requires an understanding of how certificates are created and managed, the services that use certificates, what function the certificates service, the needs of other applications, and how to minimize the expenditure on commercial certificates by purchasing only certificates that cover multiple hostnames. Microsoft provides good information on these topics in TechNet (or *Microsoft Exchange 2013 Best Practices* [Microsoft Press, 2013]). A detailed discussion of how Exchange uses certificates with its various services and to integrate with other applications such as Lync Server can be found in Chapter 8 of *Exchange Server 2013 Inside Out: Connectivity, Clients, and UM by Paul Robichaux (Microsoft Press, 2013)*. You should understand this information and plan how to meet your requirements before deploying anything.

Mail flow and rules

Rules can be a confusing topic. Exchange owns and maintains one set of rules, and Outlook owns another set. The rules feature exposed in Outlook could be considered a superset of Exchange rules because the Outlook Rules Editor has traditionally been superior to and more functional than the version supplied with Exchange for use with Outlook Web App. However, the superiority only really exists in terms of the user interface to rules. Behind the scenes, Exchange performs the majority of the processing on incoming items and leaves Outlook to perform only the steps that have to be executed on a client.

INSIDE OUT Separate rules functionality

There's a good reason Exchange has separate rules functionality. Without it, rules processing would be unavailable to users who cannot use Outlook, but it is inconsistent and ineffective for two rule sets to exist. Exchange cannot manipulate a rule Outlook owns and vice versa. Another potential point of confusion is the difference in operation between client-side and server-side rules. Client-side rules imply that some action is required that can be performed only by a client, whereas server-side rules can be executed by the server independently of any client interaction. The outcome of server-side rule processing applies to all clients, and items have to pass through server-side rules before they can be processed by client-side rules. Junk mail processing is an example of server-side processing. Exchange cleans up the mail stream by removing unwanted spam before passing messages to clients. An example of a client-side rule is one that checks incoming messages for any item that has Important in the subject and moves it into a special folder within a .pst file. Another example is a rule that looks for messages and signals an alert whenever a message from a specific user arrives. My personal favorite is a rule that delays all outgoing messages (except those marked with high importance) by two minutes to allow the sender to rescue a message that has been incorrectly addressed, doesn't have the right attachment, or has simply been sent in a moment of great passion and contains some inappropriate words. None of these rules can be processed by the server because they require Outlook to perform an action.

Exchange and Outlook work together to process messages through all applicable rules. Exchange first executes all the rule processing that is possible on the server and then, if further client-side processing is necessary, it creates a special message in the Deferred Actions folder. These messages are called *deferred action messages* (DAMs) and basically tell Outlook that it has to complete processing for a message. Outlook reads and executes the DAMs as Exchange creates them. Any DAMs that are accumulated when Outlook is offline are cleared the next time the client initializes. You never see this activity because the folder

and the messages are hidden from any client view, although you can see them by using a low-level utility such as MFCMAPI.

In previous versions of Exchange, users can maintain their rules through Outlook or Outlook Web App, the difference being that Outlook Web App has access only to rules that run on the server, whereas Outlook can access both types. The vast majority of rules are server side; only those that need to perform some form of processing on the client are client side. These include actions such as:

- Moving it to a folder in a PST
- Printing it
- Flagging a message for action in a number of days
- Flagging a message with a colored flag
- Playing a sound
- Running a script
- Starting an application
- Displaying a desktop alert

Exchange 2010 allowed administrators to access rules that executed for a user mailbox for the first time. Access was available through EMS and ECP. Exchange 2013 allows access to user rules only through the *-InboxRule cmdlet set in EMS. For example, the code to create a trivial rule that executes inside Kim Aker's mailbox and looks for the word "Hello" in either the body of the text or the message subject and then sets the message status to Read before stopping any further processing of rules is as follows:

```
New-InboxRule 'Stop Hello spam' -Mailbox 'Kim.Akers@contoso.com'
-BodyContainsWords 'Hello'
-SubjectContainsWords 'Hello' -StopProcessingRules $True -MarkAsRead $True
```

A security check on administrators

Although it's possible for an administrator to set up a trivial rule like the one just shown for a user, administrators cannot create rules that move items from one folder to another, even within the same mailbox. Administrators should not know what folders exist in user mailboxes, and it would be inconsistent and a potential security breach if they could create a rule that moved items for a user, potentially without that user's knowledge. For example, it would be a problem if an administrator created a rule in the CEO's mailbox that refiled any new message including a key phrase into a folder in the 1administrator's mailbox.

You can certainly use EMS to create a rule for your mailbox to move messages into a folder after they arrive. For example, this rule looks for any message sent to a group (identified with its SMTP address) and moves it to a folder. The folder is identified using a scheme that ensures uniqueness if not ease of use. You can use the command:

```
Get-MailboxFolder -Recurse | Format-List Name, Identity
```

to see the list of folders in your mailbox.

```
New-InboxRule -Name "Exchange Discussion Group messages" -SentTo
'ExchangeDiscussions@contoso.com' -MoveToFolder "contoso.com/Users/Akers,
Kim:\Inbox\Exchange Discussions"
```

Rules are executed in priority order, and the most recently added rule always is assigned the highest priority. You can use Get-InboxRule to retrieve details of the rules set on a particular mailbox. The rules are listed in priority order. If necessary, you can run the Set-InboxRule cmdlet to change the priority of a rule, with 1 (one) being the highest priority. Exchange automatically adjusts the priority of the other rules to account for your change.

```
Get-InboxRule –Mailbox 'Kim.Akers@contoso.com' | Format-List
Set-InboxRule –Mailbox 'Kim.Akers@contoso.com' –Identity 'Most Important Rule'
–Priority 1
```

Delivery reports

Hearing from users that an email was never delivered is a pretty common experience for administrators. From time to time, it's true that problems cause Exchange to stop working, and messages don't get delivered, but in most situations, the problem that stops a message from getting to its final destination is simple user error. Typical reasons include messages being addressed incorrectly (sent to the wrong person), messages queued to be sent in the future, or messages in the Drafts folders waiting to be sent. You can also find information about received messages such as the membership of groups in the recipient list and whether a rule operated on a message (Figure 6-20).

Although it's satisfying for an administrator eventually to discover what the problem was, it can also take a lot of time. Delivery reports enable users to interrogate Exchange to discover what happened to a message they sent. If they can't figure things out, a user or an administrator can search for the specific message and generate a comprehensive delivery report that explains exactly what happened.

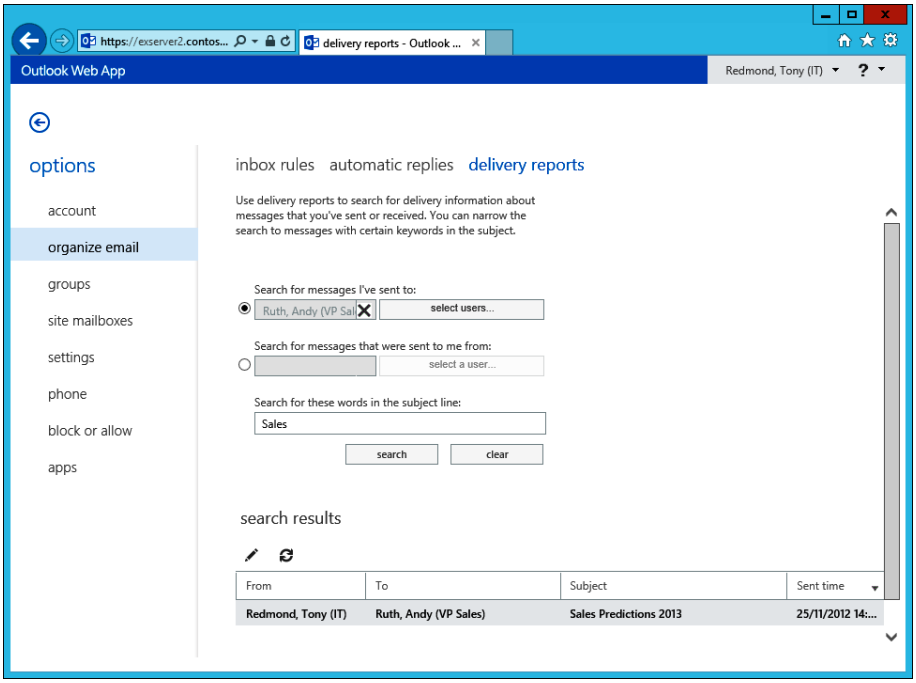


Figure 6-20 A user executing a delivery report search

Users can generate delivery reports for messages they send, provided they use one of the following:

- **Outlook Web App** Access delivery reports by opening the Organize Email section under Options (Figure 6-20).
- **Outlook 2010 or Outlook 2013** Access delivery reports in the backstage area for sent messages. (Click File and choose Open Delivery Report.) Because Outlook doesn't contain any user interface to display details of a delivery report, this option starts Outlook Web App to enable the user to access the delivery reports.

If they use another client to work with an Exchange mailbox, an administrator will have to generate a delivery report for them.

To begin a search, click Delivery Reports and then complete the basic parameters for the search. These are as follows:

- Search for messages the user has sent OR messages she has received
- AND words in the message subject line (optional)

After the search executes, Outlook Web App displays any matching messages it can find in the results pane below the search criteria. You can then select one of the messages to view details (Figure 6-21).

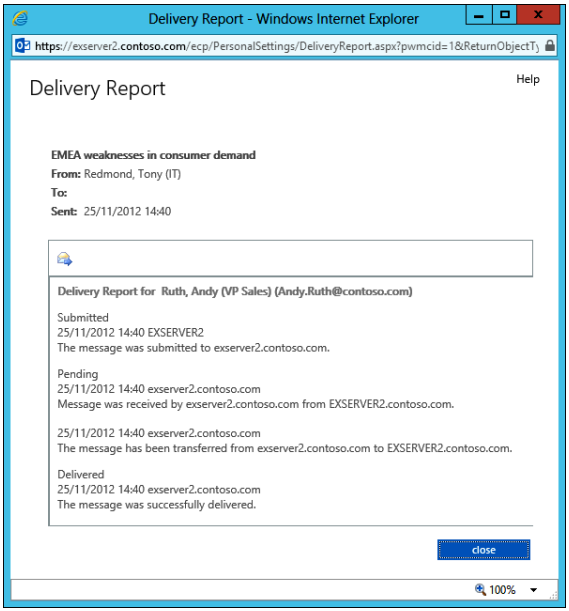


Figure 6-21 Viewing the details of a delivery report

Even if a user's mailbox has been moved between Mailbox servers since a message was sent, Exchange takes care of connecting to the various Mailbox servers that have dealt with a message to build a full picture of its path from sending to delivery.

Administrator searches for delivery reports

Users can search their own delivery reports but sometimes need a little help. They might not even know that they can search delivery reports, and they create help desk requests such as "I sent a note to someone in another company about an important contract and need to know that the message was delivered!" The help desk can talk the user through the process of conducting his own search, but sometimes it's just easier to perform the search for him. Out of the box, administrators can perform a search on behalf of a user as long as they possess one of the following roles:

- Organization Management
- Recipient Management
- Records Management

You can add the ability to perform delivery report searches by adding the Message Tracking role to a user or group. For example, this command adds the role to a Central Help Desk security group. Any user who is a member of the group can search delivery reports on behalf of another user afterward, providing she also holds the right to view information about other users through another role such as View-Only Organization.

```
New-ManagementRoleAssignment "Message Tracking and Delivery Reports" -Role "Message
Tracking" -SecurityGroup 'Central Help Desk'
```

Before starting, it's good to understand where the data that underpin delivery reports come from. Exchange maintains message-tracking logs to trace the progress of messages from the time they are submitted from a mailbox to the Store, to the transport service running on the Mailbox server, and so on, until the message exits Exchange through a connector or is delivered to one or more recipient mailboxes. Message-tracking logs have been around for many versions so administrators can track messages or analyze message volumes and patterns of use. Administrators can continue to access message-tracking logs for many purposes such as traffic analysis or message forensics.

The Delivery Reports option is valuable because it enables administrators to assign the ability to retrieve details of a message's progress and then send those details to the person who requested them to prove that the message was processed properly as far as Exchange can report this fact. Usually, this means the boundary of the organization because Exchange can't report the progress of a message through foreign email systems or even into earlier Exchange systems.

Despite the fact that Exchange can keep many weeks of message-tracking logs, delivery reports only work for the past two weeks because this is how long Exchange maintains special indexes it uses for these searches.

TROUBLESHOOTING

Why isn't a delivery report available for a moved mailbox?

Because message-tracking logs are specific to a server, it follows that some delivery reports do not work after a mailbox is moved from one server to another. Delivery reports for messages sent or received after the mailbox is moved onto a server are available because the message-tracking logs that contain the necessary information are present on the same server. Any request for a delivery report for a message sent or received before the mailbox was moved will fail because the necessary data are unavailable.

Two configuration settings control the amount of data that can be included in delivery reports. Subject logging determines whether you can see the subject of messages when

you search. It is a lot easier to locate a specific message if you know its subject and can see it in the set of results after a search. Message subjects are logged on Mailbox servers and within the transport service, and if you want to disable the collection of these data, you have to do this with the Set-MailboxServer and Set-TransportService cmdlets. The setting applies to a single server only, so if you want to do this for all servers in an organization, first you have to fetch details of all the relevant servers and then pipe the set of servers into the appropriate cmdlet. For example:

```
Get-TransportService |Set-TransportService -MessageTrackingLogSubjectLoggingEnabled
$False
Get-MailboxServer | Set-MailboxServer  -MessageTrackingLogSubjectLoggingEnabled
$False
```

INSIDE OUT Message-subject logging and confidential information

Some companies disable message-subject logging in an attempt to preserve employee privacy because giving administrators the ability to see message subjects might expose information that users would prefer not to share. For example, if an administrator performs a delivery report search against the CEO's mailbox and finds a message from the CEO to the CFO with the subject "Merger with Fabrikam Corp.," he has probably discovered something that he should not have. Eliminating message-subject logging certainly stops administrators from stumbling onto sensitive information through delivery reports and message tracking, but it also makes it much more difficult for users or administrators to locate messages. Generally, it is best to retain message-subject logging because administrators should know that if they make an unauthorized search for confidential or sensitive information, they must be prepared to deal with the consequences of their action.

It's also sometimes important to know whether a message has been read after it was delivered to a mailbox. Read-status tracking controls whether you can see if the status of a message has been changed from Unread to Read. Clients change this status when a user opens a new message or the message is viewed through a preview pane for a set time (configurable on the client). Exchange disables read-status tracking by default, so if you want to see read status reported, enable it as follows:

```
Set-OrganizationConfig -ReadTrackingEnabled $True
```

Read tracking is configured at the organization level, so if you enable the feature, Exchange tracks read status for every mailbox in the organization. You might not want to do this for specific sensitive users, such as executives or other users who have confidential positions. In this case, you can disable read-status tracking selectively on a per-user basis:

```
Set-Mailbox -Identity 'Jane Smith' -MessageTrackingReadStatusEnable $False
```

Access to delivery reports is available through the Mail Flow section of EAC. Understandably, users are restricted to searching for information about messages they sent or received, whereas administrators can select any mailbox to search for. User-generated delivery reports only show information relevant to the delivery of a message to that mailbox. For example, you can see whether a group was expanded to address a message to the mailbox or whether a rule processed the message after Exchange delivered it to the mailbox, but you cannot see what happened when Exchange delivered the message to the other recipients.

To begin an administrative search of delivery reports for another user, open the Mail Flow section of EAC and click Delivery Reports. The first step is to select the mailbox to search, followed by the message recipients (for outgoing messages) or people who sent email to the mailbox, and words that appeared in the message subject. If the message was sent to or received from an external correspondent, you can enter her SMTP address. You don't have to include any subject information, but it is a good idea to do this because it improves the effectiveness and speed of the search. When you execute the search, Exchange looks through its message-tracking log data and locates any messages that match the search criteria (Figure 6-22).

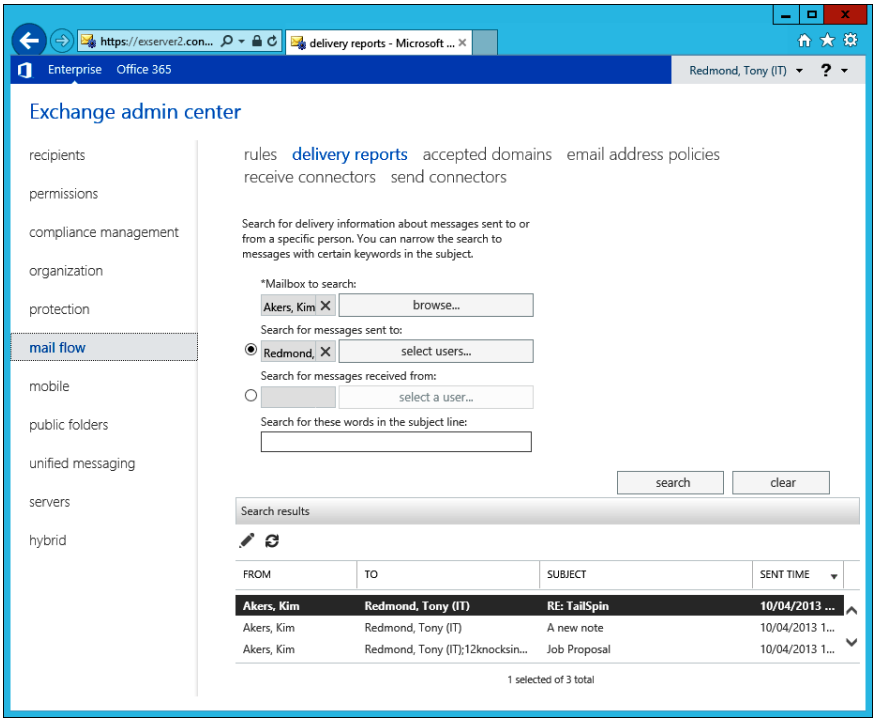


Figure 6-22 Results of an administrator-initiated delivery report search

Behind the scenes, EAC calls the Search-MessageTrackingReport cmdlet and provides it with the search parameters entered by the user who initiates the request. How to execute searches with EMS is discussed in the next section. Mailbox servers manage delivery report searches, so the first port of call is to contact a Mailbox server in the site. If you are looking for messages sent from a mailbox within the organization, the Mailbox server begins the search on the Mailbox server that currently hosts the database in which the mailbox is located. However, if you're looking for evidence of messages coming from an external sender, the search has to begin with the Mailbox servers in the Internet-facing site because that is the first place a message enters.

After the starting point is determined, Exchange contacts the Microsoft Exchange Transport Log Search service running on the server and queries it, using the search criteria. When a message is found, its path can be tracked from server to server until its final disposition is known. This might be a delivery to a mailbox or exiting the organization across a connector. Exchange can track messages across sites by sending queries through Exchange Web Services to Mailbox servers on another site. Those Mailbox servers then perform queries on their site, collate the details, and send them back to the requesting server. All the data accumulated during a search are then formatted and displayed.

You can browse the set of delivery reports that are found and select any that seem to match. If you click an item in the list, EAC displays basic information about the message, including the full recipient list and count of the number of recipients and the number of successful deliveries. You can get a full report of all the known steps the message took from sending through processing by the transport system to final delivery by clicking any of the recipients.

Tip

Like all groups, Exchange has to resolve the membership before it can route a message, so the membership of any group addressed in a message, including dynamic distribution groups, is listed in a delivery report. EAC displays 30 recipients at a time, so if a message was delivered to more than 30 recipients, you have to search the recipient list to find the person in whom you're interested before you can view full details of the delivery report for his mailbox.

Figure 6-23 illustrates the kind of information the help desk might need to satisfy a user request to check that a message has been processed successfully. You can see the list of recipients for the message and a summary of its disposition. Two copies of the message have been delivered successfully within the organization, and the other was transferred because it was addressed to an external recipient. By clicking the external recipient, you see the details of its transit, and a problem is immediately apparent. When transport attempted

to look up Domain Name System (DNS) data to discover how to transfer the message to a mail server in the external company, the lookup failed, and the message therefore remains in the outbound queue waiting to be transferred after the problem is fixed. It's probable that the help desk would have realized that problems existed before a user asked it to check, but it's always possible that people might be sending to a domain that is having problems that your monitoring systems couldn't know about. In this case, the help desk can tell the user that an issue exists and will be worked out, but in the meantime he might want to let his correspondent know that the message will be sent after the problem is fixed.

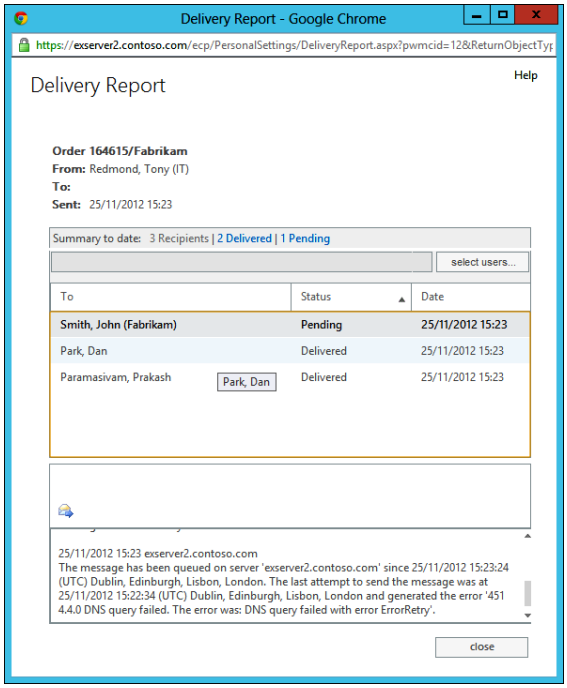


Figure 6-23 Checking the current delivery status for a message

After the problem is fixed and you have satisfied yourself that a message was delivered successfully, you can click the Email This Report link to email a copy to the user who reported the problem to prove that the message got through. At the same time, you might allow yourself a small amount of satisfaction that you have proved to a user that the email system really does work—this time.

Using EMS to search delivery reports

Behind the scenes, Exchange uses the Search-MessageTrackingReport cmdlet to execute delivery report searches. For example, a search of Samantha Smith's mailbox for messages

received from Tony Redmond might use a command like the one shown next. Note that unlike message-tracking log searches, you cannot state a date range. All delivery report searches look back two weeks. Two other points are that the BypassDelegateChecking parameter tells Exchange that you're performing a search of a mailbox that is not your own and therefore depends on the roles assigned to you to perform delivery report searches. The second is the use of the ResultSize parameter to tell Exchange that you want to find all matching messages. Up to 1,000 entries are returned if you omit this parameter.

```
Search-MessageTrackingReport -Identity 'Smith, Samantha' -Sender
'Tony.Redmond@contoso.com' -Subject 'Exchange' -BypassDelegateChecking
-ResultSize Unlimited | Select FromDisplayName, Subject, SubmittedDateTime
```

```
FromDisplayName         Subject                          SubmittedDateTime
---------------         -------                          -----------------
Redmond, Tony           For you to review (Exchange)     2/10/2013 4:44:48 PM
Redmond, Tony           Exchange design document         2/10/2013 11:49:06 AM
Redmond, Tony           Exchange Budget for 2013         2/10/2013 11:15:49 AM
```

Anyone can conduct a search for delivery reports for items in his mailbox. You use the same syntax for searches, provide the identifier of your mailbox, and omit the BypassDelegateChecking parameter.

Each of the messages retrieved by a search has a unique identifier provided in the MessageTrackingReportId property. You can use this value to retrieve full details of a selected delivery report. This operation can be a tad complicated if you have to search a large number of delivery reports, but here's the essential operation. First, you search for any message sent by a user to a distribution group. The command looks very much like your previous search except that you're looking for messages sent from a mailbox to a particular recipient (the SMTP address of the distribution group):

```
Search-MessageTrackingReport -Identity 'Tony Redmond' -Recipients
'SalesTeam@contoso.com' -BypassDelegateChecking -ResultSize Unlimited
```

Exchange fetches all delivery reports that match your search. You're interested in information about a particular recipient, so you need to go down through the delivery reports and filter out the data about the recipient. A ForEach loop that processes the output of the search does this:

```
ForEach {Get-MessageTrackingLog -Identity $_.MessageTrackingReportId
-BypassDelegateChecking -DetailLevel Verbose -RecipientPathFilter
'Tony.Redmond@contoso.com' -ReportTemplate RecipientPath}
```

This code examines the delivery reports for any detail concerning the recipient specified in the RecipientPathFilter parameter. Again, you use the SMTP address of the recipient. The

output isn't as pretty as EAC provides, but it does show what happened to get the message to the recipient.

For more information about delivery reports, the Exchange development group has blogged on this topic at *http://msexchangeteam.com/archive/2010/01/13/453792.aspx*.

Running EAC without an Exchange mailbox

Exchange enables you to allow EAC to be used by accounts that do not have an Exchange mailbox. Many companies have security guidelines that prohibit administrators from using their personal accounts (those that they use to access applications such as email and other resources) when they manage computers. The idea is to force administrators into a certain state of mind when they work in privileged environments so that a clear sense of separation exists between work performed as a system administrator and work performed as a normal user.

Most previous versions of Exchange forced every administrator account to be mail-enabled, which means that companies that wanted to enforce the separation could not do so unless they created two accounts for administrators, one for normal use and one with administrative permissions. From Exchange 2010 SP1 onward, RBAC allows an administrator to select a non-mail-enabled account and make it a member of a role group.

Figure 6-24 shows the Computer Administrator account being added as a member of a role group. You can add non-mail-enabled accounts to role groups as powerful as Organization Management or to more restricted role groups such as Help Desk.

To access EAC, the account owner logs on as normal and uses a URL such as *https://server-name/ECP*. Naturally, because he doesn't have a mailbox, any attempt to use Outlook Web App will be politely refused with the error stating that Exchange can't find a mailbox for the account.

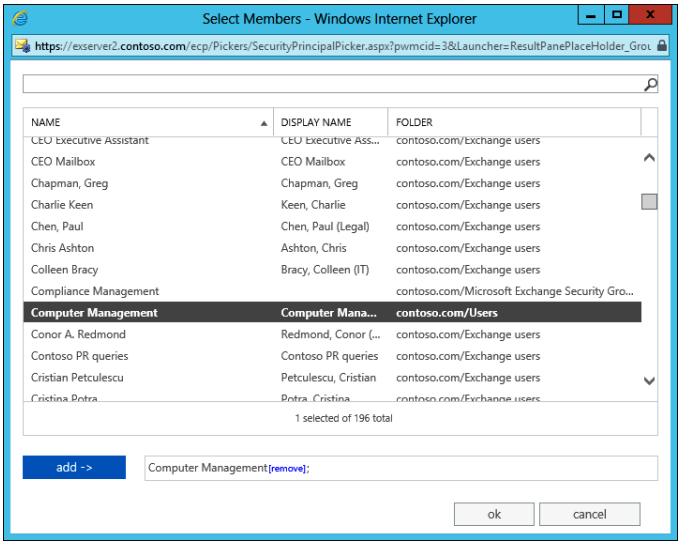


Figure 6-24 Selecting a non-mail-enabled account to include it in the membership of a role group

Setting diagnostics for Exchange servers

The ability to set diagnostics at different levels for the various components that function on an Exchange server has always existed. In Exchange 2003, you set diagnostic levels through the Management console by selecting the component (for example, ActiveSync) and the level that you wanted to apply. After a new logging level is set, Exchange complies by outputting more or less detail about its operations as events written into the Application Event Log. Exchange 2007 and Exchange 2010 support similar functionality, in this case by setting diagnostic levels as a property of a server.

The transition to EAC in Exchange 2013 resulted in the disappearance of the UI to manipulate server diagnostic levels. In addition, other developments such as the rewrite of the Store into managed code and the introduction of modern public folders meant that a large number of previously well-known event-logging categories were removed from Exchange 2013, replaced with a new logging category, or subsumed into yet another category. The upshot is that administrators who have created diagnostic scripts that depend on event-logging categories to help them debug a problematic Exchange server have some work to do reviewing their code and potentially adjusting calls to work with Exchange 2013.

The lack of UI to set server diagnostics in Exchange 2013 means that administrators have to manipulate diagnostic levels through EMS. For example, to set the logging level to High

for the operations performed through EAC (which still uses the old Control Panel name for diagnostic purposes), you type:

```
Set-EventLogLevel –Identity "MSExchange Control Panel\General" –Level High
```

Exchange then records events as administrators interact with EAC or as users interact with the Options section of Outlook Web App (which uses much of the same code as EAC). In this case, you can see that the user Administrator began working with the distribution groups section (evident in the URL) of EAC.

Microsoft documentation on the available event-logging categories and how they relate to different areas of server functionality is sparse to nonexistent. It can be difficult to know the correct event category to adjust to meet any specific circumstances. However, an easy way to retrieve a full set of current event-logging categories is by interrogating a server. To do this, use the Get-EventLogLevel cmdlet to return a list of the current diagnostic levels for a server. The list can also be used to verify that the correct settings are in place for each category. The cmdlet doesn't support a filter function so you can specify that you want only details of event categories for the Store or transport, but because you want a full list, you can just capture the output in a text file for easier examination. For example:

```
Get-EventLogLevel –Server ExServer1 > C:\Temp\EventLevels.txt
```

The list of event categories is roughly organized into areas of functionality such as Autodiscover, RBAC, replication, and so on. A little trial and error is therefore necessary to figure out the best event category for which you should increase the logging level when debugging a particular area of functionality.

As shown in Table 6-2, Exchange supports five levels of diagnostic logging based on the level assigned to events recorded by the application. Critical events and those assigned a level of zero are always written into the event log. Events with a higher level are captured if an elevated diagnostic level is chosen.

TABLE 6-2 Diagnostic levels for Exchange 2013

Level	Description
Expert	Highly verbose: Exchange essentially documents everything it does.
High	Quite verbose: Exchange logs any event with a level of five or lower.
Medium	Fairly detailed: Exchange lowgs any event with a level of three or lower.
Low	Reasonable detail: Exchange logs any event with a level of one or lower.
Lowest	Only critical events or errors with a logging level of zero are captured; this is the default level for all Exchange services.

CAUTION!

Be careful about setting diagnostic levels to Medium or higher. Exchange is quite happy to provide a vast amount of diagnostic information by writing events into the Application Event Log, but you run the risk of being unable to see the forest for the trees, and some essential piece of information will be overlooked simply because so much data are available. To prevent the Application Event Log from being clogged up with an excessive number of events, make sure that you reset the diagnostic level to Lowest when you've completed troubleshooting the problem that caused you to elevate the level. You can run a similar command to discover event categories that are set to elevated levels:

```
Get-EventLogLevel -Server ExServer1 | Where {$_.EventLevel -eq "High",
"Expert"} | Format-Table Identity, EventLevel
```

But what will you manage?

You now understand how to manage Exchange recipients by using EMS and EAC, and you have probed some of the detail about the objects that require management. It's time now to move on to the heart of Exchange 2013 and plunge into the Information Store.

CHAPTER 7

Addressing Exchange

Email address policies 333
Address lists. 345
Address book policies 348
The Offline Address Book 357
Hierarchical address book. 371
MailTips and group metrics 373
The display or Details Templates Editor 382
The next step. 386

Creating mailboxes is all very well, but the users of those mailboxes must be able to address messages before the mailboxes are truly functional. Exchange Server has quite a sophisticated set of interconnecting components that enable clients to address and send messages properly. This chapter covers how email addresses are allocated to mailboxes, how address lists are built for display to clients, how to create and implement an address book policy (otherwise known as GAL segmentation), how the Offline Address Book (OAB) works, how to add photo information to user and other objects, how MailTips work, and how to build a hierarchical address book (HAB). With all that in mind, let's plunge into the details of addressing.

Email address policies

Email address policies define the format of email addresses Exchange creates for mail-enabled objects such as mailboxes and groups. Exchange 2013 applies email address policies in a consistent and immediate manner because the Exchange Management Shell (EMS) cmdlets that create or update mail-enabled recipients call a common set of business logic. The only way to stop Exchange from applying an email address policy to an object is to exclude it explicitly from these updates. For example:

```
Set-Mailbox –Identity 'David Pelton' –EmailAddressPolicyEnabled $False
```

Typically, the effect of an email address policy is felt when a new mail-enabled object is created or updated or after an administrator changes a policy that creates a new email address type or format. After you install Exchange 2013 for the first time, the installation creates a default policy under the Mail Flow section of Exchange Administration Center (EAC). (Installing Exchange 2013 in an existing organization preserves whatever address policies are already in place.) The default email address policy creates email addresses by using the alias assigned to the object plus the SMTP domain of the Exchange organization and is applied to every mail-enabled object until another email address policy is created with a

higher priority. For example, an object with an alias of TR in the contoso.com organization will receive an SMTP address of TR@contoso.com.

Behind the scenes, each email address policy uses a recipient filter to determine the objects that come within the scope of the policy. Logically, the default email address policy must have a very broad recipient filter because it has to apply to any object for which Exchange can find no other matching email address policy. The recipient filter for the default email address policy is very simple because the filter catches everything. You can see details of the policy with the Get-EmailAddressPolicy cmdlet:

```
Get-EmailAddressPolicy -Identity 'Default Policy' | Select Name, *Filter*, *Exch*
```

```
Name                        : Default Policy
RecipientFilter             : Alias -ne $null
LdapRecipientFilter         : (mailNickname=*)
LastUpdatedRecipientFilter  : Alias -ne $null
RecipientFilterApplied      : True
RecipientFilterType         : Precanned
ExchangeVersion             : 0.1 (8.0.535.0)
```

This policy is the same as used with Exchange 2010, and its recipient filter ensures a match against all recipients that have an alias. It uses a simple filter that is preset within Exchange rather than a custom-coded filter that is designed to select a specific group of recipients. Precoded and custom filters, and how they underpin dynamic distribution groups, are discussed in "OPATH queries" in Chapter 6, "More about the Exchange Administration Center." Setting *8.0.535.0* as the value for the *ExchangeVersion* property enables any of the Exchange servers that operate alongside Exchange 2013 to manage the policy.

Email policy priority

Several email address policies can exist within an organization. All of them generate email addresses for the mail-enabled objects that fall within their scope, and these email addresses are valid for both external and internal communication. Note that email addresses can be generated for accepted domains only.

It is generally a good idea to restrict the number of email address policies in use to the minimum possible because it can become complicated to understand where email addresses originate, especially when policies are based on different recipient properties. For example, a mailbox might have email addresses generated by the following policies:

- A specific departmental or geographic email address policy (for example, Tony.Redmond-IT@contoso.com or Tony.Redmond@ireland.contoso.com)
- A more general industry-standard template for SMTP, such as first name.last name@domain (for example,Tony.Redmond@contoso.com)

- The Exchange default email address policy based on mailbox alias (for example, TRedmond@contoso.com)

When an object has multiple email addresses, the primary email address is placed on outgoing messages to allow recipients to respond to those messages and have the replies routed back to Exchange for delivery to the correct mailbox. The other addresses are referred to as proxy addresses and are present so Exchange knows how to process messages sent to those addresses. Sometimes proxy addresses are maintained for backward compatibility. For example, if the company has changed its name and Internet domain, you could decide to retain all the email addresses from the old domain as proxy addresses on mailboxes so that external correspondents who reply to messages sent from the old domain can still be routed correctly.

Each email address policy is assigned a priority or sequence order. (One is the highest priority.) You can use the up and down arrows EAC displays to increase or decrease the priority of an email address policy or run the Set-EmailAddressPolicy cmdlet in EMS:

```
Set-EmailAddressPolicy -Identity 'IT Department' -Priority '1'
```

Figure 7-1 shows that three email address policies are currently defined for the organization. A department-specific policy has precedence, followed by a general policy that applies to all recipients and then the default policy created when Exchange is installed.

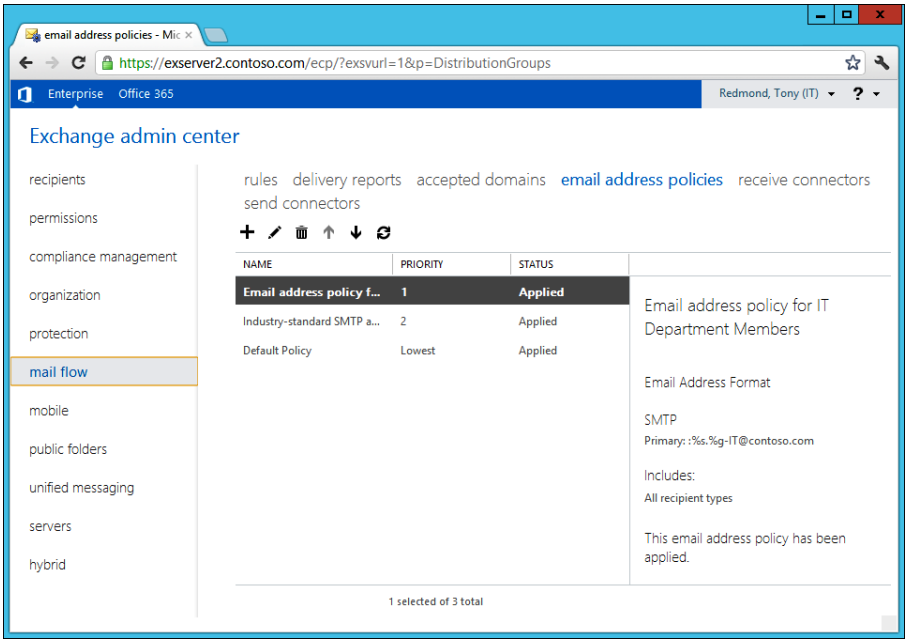


Figure 7-1 Viewing email address policies for an organization

Not every email address policy might apply to an object. Exchange searches the policies in priority order to find the highest-rated policy to use when it needs to apply a policy to an object, and the address generated by that policy becomes the default email address for the object. The practical effect is that this address is used as a reply address for outgoing messages. You can see the email addresses created for an object by viewing its properties. You can see from Figure 7-2 that the mailbox has three email addresses with the default or primary address identified by showing its format (SMTP) in uppercase. You can also see that this mailbox is under the control of email policies because the check box is set to enforce this condition.

When you create the first custom email address policy within an organization, it is automatically assigned a priority of one, and the default policy is moved down to a special priority of lowest to ensure that any custom email address policy takes precedence. The default email address policy should serve as the catch-all policy that ensures that an email address is assigned to every object and is therefore given the lowest priority.

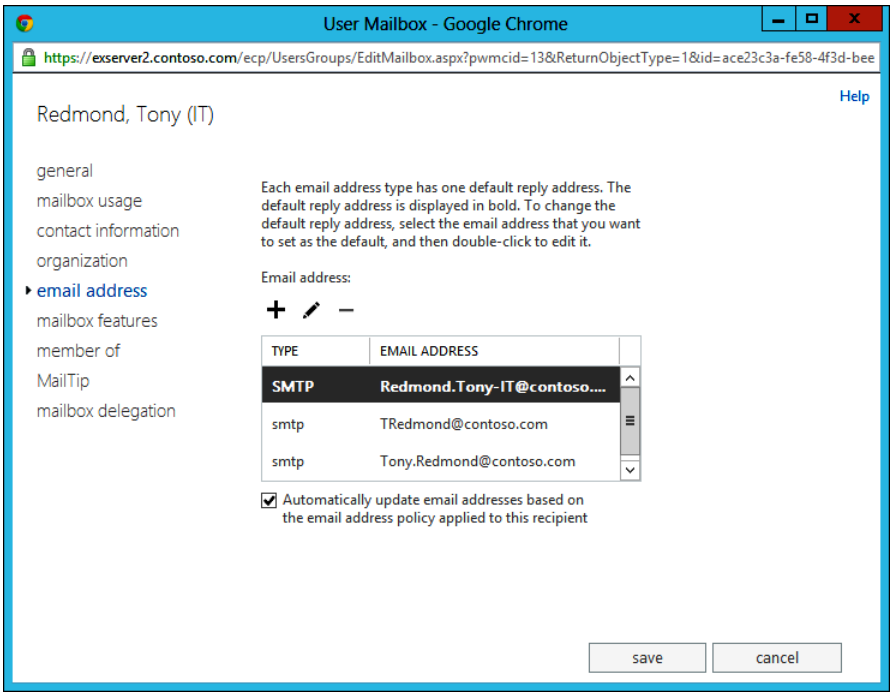


Figure 7-2 Viewing the email addresses for a mailbox

Creating a new email address policy

Exchange operates quite happily with just the default email address policy in place because the unique alias assigned to each new object ensures that the resulting email address is also unique. However, most companies use different conventions for generating email addresses that are shared across all email systems in use within the company. Exchange provides the flexibility to create and apply email addresses in your chosen format through custom email address policies that you can apply to some or all of the organization. By default, email address policies create SMTP format addresses, but you can also configure proxy addresses for other email systems such as Lotus Notes and X.400.

To begin, click New E-Mail Address Policy in the action pane. EAC displays the New Email Address Policy screen, shown in the left side of Figure 7-3. In this screen, enter the essential details of the new policy, including:

- **Name** As always, make the name meaningful, for example, "Address policy for IT Department members."
- **Email address format** If you're just concerned with Exchange, you only need SMTP format addresses. However, you might also need to provide addresses for other forms of email systems to allow interoperability with those systems or for backward compatibility.
- **Sequence order** A number of policies might be applicable to email recipients. The sequence number identifies the priority Exchange assigns when it generates email addresses for recipients.
- **Recipient filter** Just as with dynamic distribution groups, Exchange uses a recipient filter to determine which mail-enabled objects fall under the scope of the email address policy. You can click All Recipients to have Exchange apply the policy to mailboxes, groups, contacts, public folders, site mailboxes, and so on, or you can limit the objects the policy processes by creating a more complex filter. How to do this is discussed in the "Focusing on certain recipients by using filters" section later in this chapter.

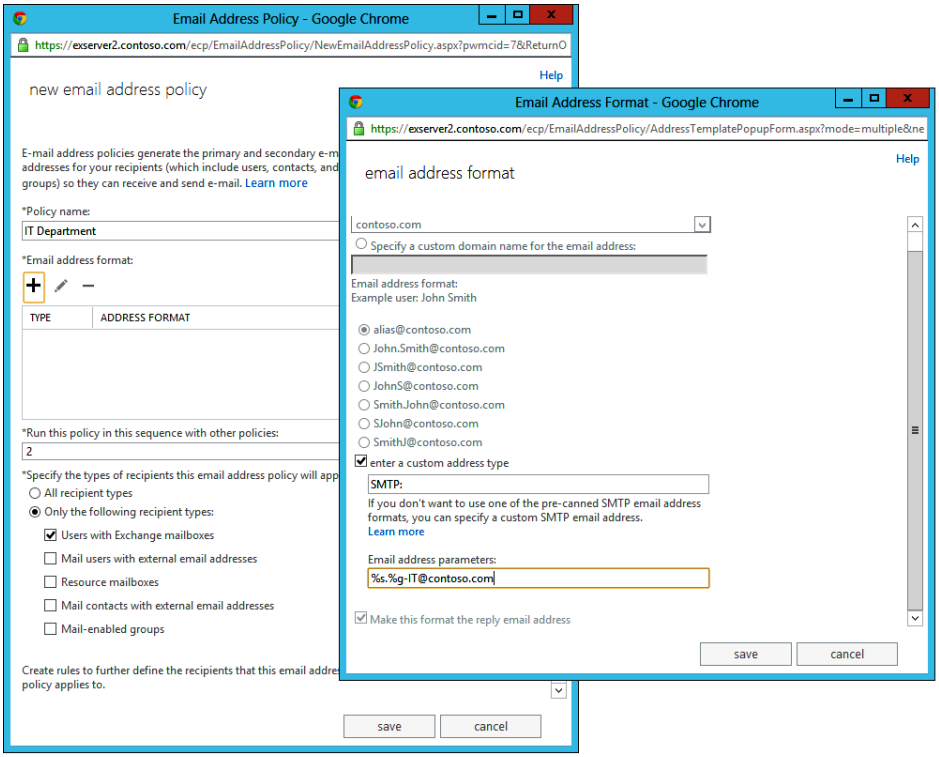


Figure 7-3 Creating a new email address policy

The rightmost screen in Figure 7-3 shows how you can specify a certain email format for use by a policy. In this instance, you're creating an SMTP format email address. EAC provides a number of common email address formats from which to select, but you can choose Enter A Custom Address Type if you want to create a non-SMTP type of address or an SMTP address that isn't one of the predetermined formats listed by EAC. Taken together, the values shown in Figure 7-3 result in the creation of an email address policy to identify mailboxes that belong to the IT department. (I'll come back to how this filter was determined shortly.) The selected format is:

%s.%g-IT@contoso.com

The values in Table 7-1 help interpret the meaning of the syntax that specifies the format of an email address. In effect, the selected format means:

Create email addresses by inserting the surname (%s), a full stop, the given or first name (%g), and then the -IT@contoso.com string.

This results in an address like Smith.Joan-IT@contoso.com.

You might think that this is an ugly email address and certainly not very user-friendly. This feeling is correct, but it is neither the ugliest form of email address that has been used in the past nor the ugliest that is contemplated in the future. For now, it simply serves as an example of what's possible to do with email address policies.

You can elect to add one or more email addresses to an email address policy. For example, you might decide to add an email address that is user-friendly such as Kim.Akers@contoso.com and have another address that is used for internal routing purposes. This is a reasonably common arrangement when companies support several email systems and use front-end bastion servers to cleanse the incoming email stream arriving from the Internet and then route messages to the appropriate transport server for the addressee's email system.

Another scenario occurs when users are assigned both a user-friendly email address designed for external publication and another that is used internally. For example, you might have Tony.Redmond@contoso.com and TRedmond@IT.EMEA.contoso.com. The first address works well for business cards; the second address contains some internal information that you might not want to share outside the company. Mergers and acquisitions or company renaming exercises create other scenarios in which multiple email addresses might be required.

TABLE 7-1 Email address policy mask variables

Variable	Value
%g	Given name (first name)
%i	Middle initial
%s	Surname (last name)
%d	Display name
%m	Exchange alias
%xs	Uses *x* number of letters of the surname; for example, if the surname is Smith and the variable is %2s, Exchange inserts Sm.
%xg	Uses *x* number of letters from the given name; for example, if the given name is Jane and the variable is %3g, Exchange inserts Jan.

After you create or alter a new email address policy, you must review its place in the priority order and adjust it as necessary. You also need to apply the policy. After they are created or updated, email address policies are in an unapplied state, including if you adjust their priority order. This means that Exchange has not applied the new or updated policy to any objects. To make the policy effective, click Apply in the action pane to force Exchange to update the objects that come under the policy's scope with new email addresses. When an email address policy is applied, Exchange does the following:

1. Finds the first matching email address policy for an object and creates a new email address (if required) for that object

2. Sets the primary email address for the object to be of the highest priority email policy

Note

Exchange will not remove any email addresses that exist for an object because this would prevent that object from receiving messages sent to those addresses. If you want to remove an email address from an object, you have to select it and delete the address. EAC won't allow you to remove the primary email address from an object.

Applying an email address policy to thousands of objects can take some time; it also causes a flurry of Active Directory replication activity to populate the new email addresses across all the domain controllers and global catalogs within the forest. Updating email addresses through a new policy also affects the OAB, and if a large number of objects are updated, it will force users to download the entire OAB. For these reasons, it is best to schedule mass updates of email addresses for times when user demand is low.

Sometimes you are forced to update many recipients to ensure that they receive email addresses of the right type. The example email address policy that you created for the IT department will be applied only to recipients that are stored in the IT organizational unit (OU). Exchange applies the policy after you create it to the recipients that already exist in the OU, but if you move recipients into the OU later (perhaps after they join the IT department), you must update their addresses. If you prefer, you can start the update process through EMS with a command such as the one shown next. Note that the Update-EmailAddressPolicy cmdlet does not include the ability to schedule the update for a future date and time. You can combine this cmdlet with a suitable scheduling utility if you want to apply the update at a particular time.

```
Update-EmailAddressPolicy -Identity 'IT Department'
```

Tip

An email address policy is capable of accommodating scenarios such as company mergers and acquisitions. The only issue you have to take care of beforehand is to create an accepted domain record for every domain you want to use in email addresses. For example, if you want to use contoso.com and contoso-europe.com, both must be known to Exchange as accepted domains before they can be used in email addresses. You don't have to make the domain you use when you install Exchange known as an accepted domain because this is done as part of the installation process.

Focusing on certain recipients by using filters

The most basic form of a recipient filter used with email address policies is that used for the default policy, which sets a scope that tells Exchange to apply the policy to every mail-enabled object in the organization. However, like dynamic distribution groups, you can create sophisticated recipient filters to focus the scope of an email address policy as tightly as you would like.

If you use the Get-EmailAddressPolicy cmdlet to examine the recipient filter for the default policy, you see that it contains this filter:

```
Alias -ne $null
```

However, the recipient filter for the email address policy for the members of the IT department is:

```
((Department -eq 'IT') -and (RecipientType -eq 'UserMailbox'))
```

In other words, "Select any recipient that is a user mailbox and has IT as its department."

EAC offers a little less flexibility in terms of creating recipient filters for email address policies than did EMC. The benefit here is that EAC focuses on the common criteria used for recipient filters and excludes some of the more complicated and less used criteria that EMC allowed administrators to manipulate (and possibly get wrong). As always, if you need to create a more complex recipient filter, you can do so through EMS.

For now, concentrate on using EAC to create a recipient filter. This approach is similar to creating a recipient filter for a dynamic group and offers the same selection of preformatted properties from which to choose. You can:

- Specify the kind of mail-enabled objects to which Exchange should apply the policy. Every mail-enabled object needs an email address, which is why the default email address policy applies to all kinds of mail-enabled objects. However, you might decide that rooms and resource mailboxes should have different forms of email addresses than user mailboxes. EAC enables you to select the different types of mailboxes to which a policy can apply.

- Specify a particular Active Directory container to locate the objects for the policy to process. Each email address policy has a RecipientContainer property that tells Exchange which objects to process. By default, the property is left blank, so Exchange should process every object in the organization that matches the recipient filter. In Figure 7-4, a particular OU in Active Directory has been selected as a rule for the recipient container to instruct Exchange that this policy only applies to objects in that part of Active Directory.

- Specify other rules to focus the policy even more tightly. These rules are those available for dynamic distribution groups and include the 15 custom attributes.

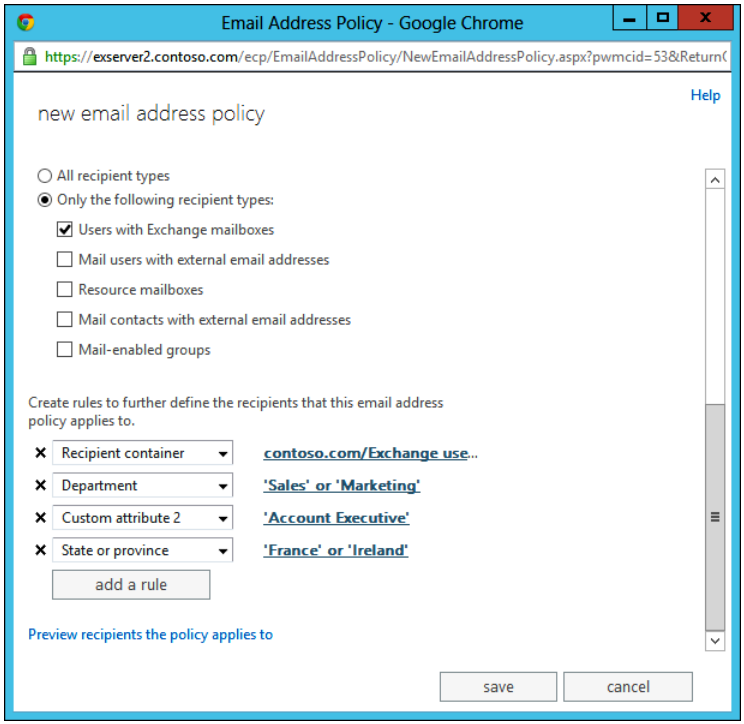


Figure 7-4 Adding rules to an email address policy

If you view the recipient policy in the email address policy created in Figure 7-4 by using the Get-EmailAddressPolicy cmdlet, you see:

```
RecipientFilter : ((((Department -eq 'Sales') -or (Department -eq 'Marketing'))) -and
(((StateOrProvince -eq 'France') -or (StateOrProvince -eq 'Ireland'))) -and
(CustomAttribute2 -eq 'Account Executive') -and (RecipientType -eq 'UserMailbox'))
```

As stated before, it's best to restrict the number of email address policies that are used across the organization to avoid confusion and the proliferation of multiple email addresses for objects. However, if you need to create a focused email address policy, it's probable that you can do this with EAC, and if the recipient filters generated by EAC are insufficient, you can then explore custom filters for email address policies.

Creating email address policies with custom filters

Why would you need to use EMS to create a recipient filter for an email address policy? The need to use EMS probably will not arise very often. The most common reason is that you want to filter based on a property that is not exposed in the EAC user interface (UI).

This requirement can be handled by creating an email address policy with a custom filter using the New-EmailAddressPolicy cmdlet. In this situation, the syntax rules are the same as those used to specify recipient filters for dynamic distribution groups, so the filters you can create are very flexible. For example, assume that you want to create an email address policy that applies to mailbox users in the Dublin office only. You could use New-EmailAddressPolicy to create the new policy and then immediately apply it to the matching recipients with Update-EmailAddressPolicy. Note that before Exchange will accept the value used here for the primary SMTP address template (@dublin.contoso.com), this domain must be created as an accepted domain for the organization.

```
New-EmailAddressPolicy -Name 'Dublin Office Users' -RecipientFilter
{City -eq 'Dublin' -and RecipientTypeDetails -eq 'UserMailbox'}
-EnabledPrimarySMTPAddressTemplate 'SMTP:%g.%s@dublin.contoso.com' -Priority 2
Update-EmailAddressPolicy -id 'Dublin Office Users'
```

Two big issues come into focus here. First, it's easy to create an email address policy that contains a recipient filter that never applies to any object. This happens if you specify a property Exchange cannot use to filter recipients or if you just create an incorrect filter. One way around this problem is to test the filter you propose to use by inputting it to the Get-Recipient cmdlet on the basis that if the filter returns the correct object set when used with Get-Recipient, it will generate similar results for an email address policy. For example:

```
Get-Recipient -RecipientPreviewFilter "City -eq 'Dublin' -and RecipientType -eq
'UserMailbox'"
```

Second, unlike the email address policies so far considered, policies created in this manner have a custom recipient filter. You cannot edit these policies with EAC afterward, so if you need to update the recipient filter subsequently, you will have to do it by writing a new recipient filter with the Set-EmailAddressPolicy command. Figure 7-5 shows how EAC recognizes that an email address policy has a custom filter that it cannot edit and the very useful preview facility that enables you to see exactly which objects Exchange will update with new email addresses when the filter is applied.

The ability of Exchange to send messages was originally based on X.400, but it now has a firm preference for SMTP, and this preference is reflected in the fact that the primary and sometimes only email address assigned to mail-enabled objects is an SMTP address. However, over its lifetime, Exchange has supported many types of email addresses to permit interoperability with foreign email systems such as Novell GroupWise.

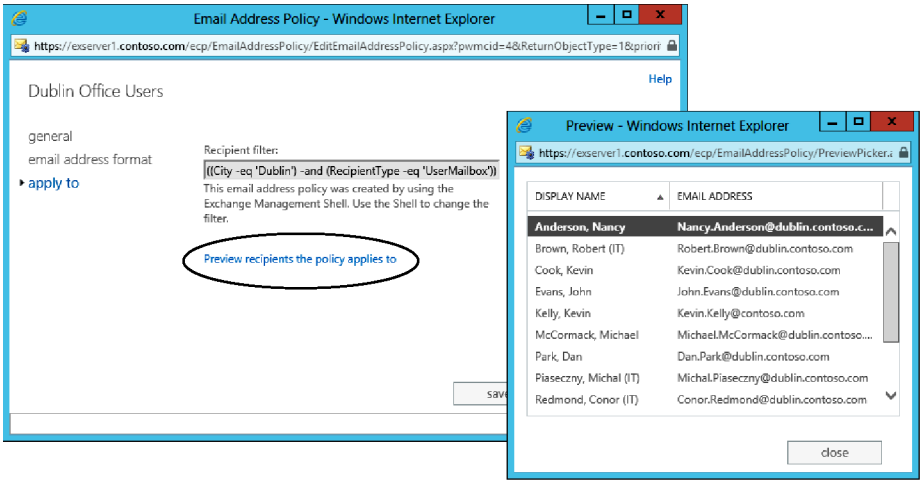


Figure 7-5 Previewing the effect of the filter contained in an email address policy

Each email address format is defined to Exchange in its configuration data. The entry for an address format stores details of the dynamic link library (DLL) that Exchange calls to create addresses in the format. Out of the box, Exchange 2013 provides DLLs for SMTP, Lotus Notes, and X.400 format addresses. Figure 7-6 shows details of the Notes address format as viewed through ADSIEdit. The DLL Exchange calls to generate email addresses of this type is ntspxgen.dll (highlighted), which must be available to Exchange in the \v15\mailbox \address\ directory.

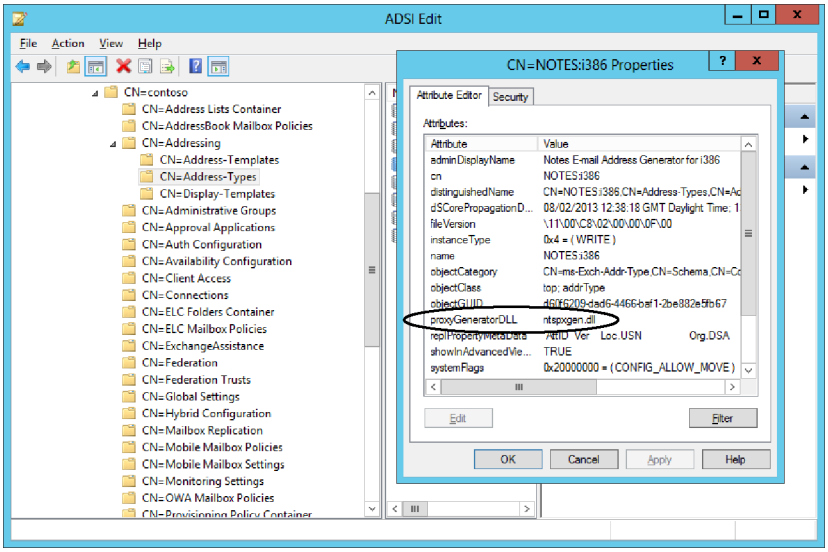


Figure 7-6 Viewing the details of the Notes email address format

Software vendors that provide solutions that integrate with Exchange and require a specific email address format must include their own custom DLL to enable Exchange to incorporate email addresses of the format into email address policies and generate the appropriate proxy addresses for the objects that fall under the scope of the policies. Sometimes this detail is omitted, and you're then forced to consider how to create and maintain the required addresses without using email address policies.

One approach that has been successfully used is to generate and apply the set of required proxy addresses to objects though EMS scripts. This approach works but, because it is manual, not integrated into the Exchange management framework, and prone to error, it is an approach that should be used only when it is impossible to manage proxy addresses through email address policies for some reason.

INSIDE OUT **Controlling Update-EmailAddressPolicy**

The Exchange 2013 version of the Update-EmailAddressPolicy cmdlet supports the UpdateSecondaryAddressesOnly parameter. This switch tells Exchange to leave primary email addresses untouched when the cmdlet runs and to update secondary email addresses only. You might need to use this capability if you synchronize Exchange with a foreign email system and want to be sure that the primary addresses are never updated because they are used as the link to that system.

Address lists

Address lists provide a convenient method to present directory objects in containers and can be combined in an address book policy (ABP) to restrict users to the Active Directory information they can see. By default, Exchange comes with a number of predefined containers defined by address lists:

- **All Users** All the mailboxes known to the organization
- **All Groups** All the distribution groups and dynamic groups known to the organization
- **All Contacts** All the mail-enabled contacts known to the organization
- **All Rooms** All the room mailboxes known to the organization
- **Public Folders** All the public folders (both traditional and modern) known to the organization

These address lists are available to all clients. In addition, the default Global Address List (GAL) is defined to be the sum of all the mail-enabled objects within the organization. EAC manages both normal address lists and GALs through the Organization section. Behind the scenes, the *-AddressList cmdlet set interacts with address lists, and the *-GlobalAddressList cmdlet set manages GALs. Your account needs to hold the RBAC Address Lists role before you can work with address lists. The role is included in the Organization Management role group.

Address lists are used to create virtual organizations through the framework of an ABP. If you are not interested in address book policies, you might never go near the standard set of address lists because they meet the needs of the vast majority of Exchange deployments and, certainly, those that support only up to a few thousand mail-enabled objects when it is relatively easy to browse or search the directory for recipients.

Larger Exchange deployments might consider it a good idea to break up the directory into smaller chunks to help users navigate. For example, if your GAL contains 100,000 objects, it might be more convenient to provide users with address lists based on department, office, or country to assist users to find recipients. Exchange certainly makes this possible, and it is very easy to create a set of new address lists based on a broad range of criteria, remembering that users will continue to have access to the totality of the GAL unless this is restricted by an ABP.

INSIDE OUT **OPATH filters once again**

On a technical level, just like dynamic distribution groups and email address policies, address lists depend on OPATH filters to extract the desired recipients from the directory. The basic rule then is that if you can create a query to select the objects you need, you can create an address list.

Imagine that you want to provide a country-centric view into the GAL. To do this, you need a separate address list for every country. (Note that clients have a limited amount of UI space to display address lists, so it's best to restrict yourself to creating the most important five or six address lists.) Like anything to do with queries against Active Directory, the assumption is that the data in the directory is sufficiently populated to generate reliable results. In the example shown in Figure 7-7, an address list for Contoso France is created, and the query is based on two major factors: all recipient types and the value of the Active Directory StateOrProvince attribute set to France.

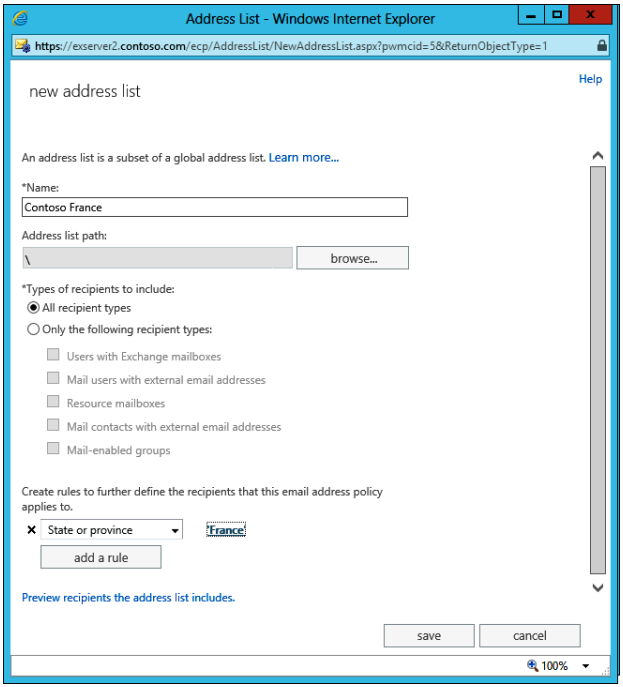


Figure 7-7 Creating an address list for Contoso France

Notice the Preview Recipients The Address List Includes option toward the bottom of the screen. Like the preview feature supported for email address policies, this enables you to validate that the query specified for the address list will create the desired set of recipients when run against the directory. You can use this option to check the query and adjust as required until you are happy that the correct results are displayed; then save the new address list. Looking at the saved object with the Get-AddressList cmdlet, you see:

```
Path                        : \Contoso France
DisplayName                 : Contoso France
Name                        : Contoso France
RecipientFilter             : ((StateOrProvince -eq 'France') -and (Alias -ne $null))
LdapRecipientFilter         : (&(st=France)(mailNickname=*))
LastUpdatedRecipientFilter  :
RecipientFilterApplied      : False
IncludedRecipients          : AllRecipients
RecipientContainer          :
RecipientFilterType         : Precanned
```

These properties should be very familiar because they are similar to those used by dynamic groups and email address policies. The critical pieces are the RecipientFilter and the IncludedRecipients properties.

TIP

Make sure of your display names. Like other Exchange objects, address lists have both Name and DisplayName properties. The Name property identifies the object; the DisplayName is what users see when the address list appears when they browse in places such as the People section of Outlook Web App. You don't need to specify a display name when you create an address list with EMS, but if you do, make sure that the DisplayName property is suitable and conveys the intention and purpose of the list.

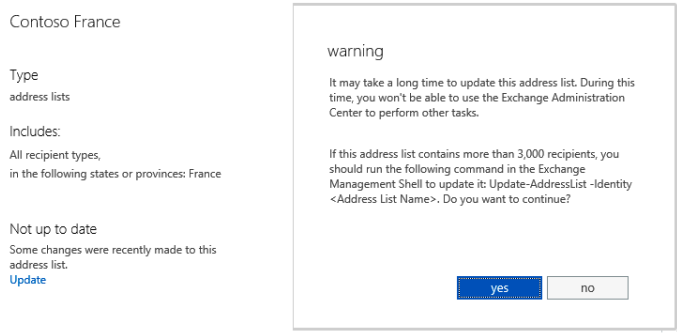


Figure 7-8 Populating the Contoso France address list

As soon as a new address list is created, it becomes visible to clients. For example, in Outlook Web App, the new address list shows up under Other Address Lists in the People section. However, just after creation, the address list is an empty container, and it needs to be populated before users will see anything after they select the list. If you select the new address list and look at its properties, shown in the rightmost property pane, you'll see that EAC indicates that some changes recently occurred in the address list and that it needs to be regenerated. Click Update to proceed and ignore EAC's warning that this action could take some time to complete (minutes rather than hours) in large organizations, where an address list might include tens of thousands of objects. After the update completes, users see the recipients identified by the query when they click the address list.

Address book policies

Many Exchange deployments will be quite happy with the default GAL, by which every mail-enabled object in the organization is visible to every user. However, situations do arise when companies want to compartmentalize the GAL so that different users see different sets of objects, even though everyone is part of a single organization. In effect, each set of objects represents a virtual organization.

Classically, the reason such segmentation arises is to allow a single Exchange organization to support multiple businesses or even complete companies, possibly managed by a hosting company that wanted to extract maximum advantage from a single infrastructure. Office 365 is an example of a massive shared infrastructure in which all the tenant companies are deployed across common servers, storage, and network. Each of the tenants is logically separated from the next by software boundaries, one of which is the view of the GAL created through address book policies (ABPs). The Office 365 deployment is unique, but it illustrates the problem ABPs seek to solve.

ABPs create a policy framework in which users are associated with a certain set of address lists that define the view they should have of the overall directory. Normally, users have access to a GAL and OAB and are unaware that the available objects have been restricted by policy. Historically, this setup is known as GAL segmentation because administrators segment the overall user population into sets that are identified by the queries that underpin ABPs. Each ABP consists of a GAL, an OAB, room lists, and one or more address lists. If you don't use room lists, you can include an empty list to satisfy the requirement. After an ABP is created, it can be assigned to mailboxes. When assigned, the owners of those mailboxes are restricted to the views included in the ABP by the Client Access Server (CAS) to which they connect.

It's important to realize that ABPs create a virtual view of the world that is easily circumscribed if users in the virtual organizations wish to communicate with one another. Just like when she wants to send mail to anyone else, all a user in one virtual organization needs to communicate with a user in a different virtual organization is that user's SMTP email address. The presence of ABPs within the overarching organization does not prevent the different sets of users from communicating with one another. If it is a business requirement to prevent the different sets of users from sending email to one another, you must introduce a transport rule to look for messages sent from one virtual organization to another. This is not a perfect solution because users can still communicate with one another by sending email from something like a consumer email system.

ABPs were first introduced in Exchange 2010 SP2. For this reason, they only work for users who connect to Exchange 2010 SP3 or Exchange 2013 servers. In addition, you should not run CAS servers on global catalogs because this forces the CAS to access Active Directory directly rather than going through the address book service. Active Directory knows nothing about ABPs, so valid but undesired results are returned for any address book lookups. Outlook for Mac 2011 uses Lightweight Directory Access Protocol (LDAP) queries to access Active Directory and encounters the same problem of inconsistent results if these clients connect to a CAS collocated on a domain controller or global catalog, all of which underscores the best practice that Exchange 2013 should never be installed on an Active Directory server. Another factor to consider is that hierarchical address books (HABs) do not coexist alongside ABPs; you can have one or the other feature but not both.

Finally, although you can create and manage the underlying address lists that form an ABP by using EAC, you cannot manage ABPs with EAC. ABPs have to be created and managed using EMS.

ABPs and groups

Groups are Active Directory objects, and as you already know, Active Directory knows nothing about ABPs. Therefore, if Active Directory is queried to discover the membership of a group, it will return the full membership without regard to any logical boundaries that an administrator seeks to establish through an ABP. For this reason, if you deploy ABPs, you should positively avoid creating distribution groups that contain members from different virtual organizations. If you do, you are likely to run into several problems, including the following:

- If a group member who belongs to a virtual organization sends a message to a group that contains members from multiple virtual organizations and requests a delivery or read receipt, he will receive notifications from Exchange that reveal the existence of the other virtual organizations.
- Exchange flags warnings when a user sends an encrypted message to groups that contain members who do not have valid digital IDs. The user will be told the number of group members who will have a problem seeing the encrypted message and the list of their addresses. However, the notification data is incorrect when a group spans organizational boundaries because not all recipient addresses will be shown. This might cause some confusion to users.
- The Get-Group cmdlet knows nothing about ABPs because it accesses Active Directory directly. Anyone who can run this cmdlet can therefore use it to expose the full membership of any cross-organization group.
- Outlook Web App enables users to maintain the groups they own and the membership of those groups. However, Outlook Web App does not include the filters that ABPs use to restrict access to Active Directory to create the virtual organizations. It is therefore best to modify the default RBAC management role for users to exclude access to the MyDistributionGroupMembership role (see the description in Chapter 6) because Outlook Web App requires group owners to have full visibility of the complete membership of a group.

None of these issues is particularly important, and all can be managed with a little forethought and planning. However, they do illustrate that the deployment of ABPs is not simple because both the underlying technology (Active Directory) and some parts of Exchange remain unaware of object segmentation. With this in mind, explore how to create a new ABP and what this means to users.

Creating and implementing an ABP

Before creating an ABP, understand the business requirement that is forcing the establishment of a new policy. Sometimes, as in the case of a hosted scenario, the requirement is obvious because you would not want to allow users belonging to one virtual organization to be aware of anyone else sharing the same Exchange infrastructure. In other situations, the business requirement is not so obvious and perhaps arises only in unusual circumstances. For example, assume that a company is preparing to sell off part of its business and wants to begin the process of separating the IT infrastructure during the period when it seeks regulatory approval for the sale. In this situation, it might be a good idea to move the users who belong to the soon-to-be-sold business into a virtual Exchange organization so that when the break happens, users won't see any difference in the view they have of the GAL.

To illustrate what needs to be done, assume that Contoso seeks to sell its Irish subsidiary to another company. Accepting that many other things need to be done to isolate and break off the IT components that might serve the users in the affected subsidiary, here's what you could do to create an ABP that provides the Irish users with a GAL that contains only Irish users and the rest of Contoso with a GAL that includes everyone but the Irish users.

First, create some address lists. These can be done through EAC or with EMS. Three address lists are required for the Contoso Ireland subsidiary:

- Contoso-Ireland-Users: Contains all the users and distribution groups belonging to Contoso Ireland
- Contoso-Ireland-Rooms: Contains all the resource mailboxes belonging to Contoso Ireland
- Contoso-Ireland-Contacts: Contains all the mail-enabled contacts belonging to Contoso Ireland

Now you must decide how to isolate the Contoso Ireland objects from the general directory. Many companies include country or office codes in the data held in Active Directory for users and other objects. If this data is consistent and accurate, it can certainly be used to identify the objects that belong to the Irish subsidiary. You will need to review and probably update records to ensure a high degree of accuracy.

Depending on the size of your organization and the split in responsibilities that exists between Exchange and Active Directory, you might also have to decide whether to rely on basic Active Directory attributes or those that are specific to Exchange. For example, the Office or StateOrProvince attributes seem to be good candidates by which to isolate Irish users, but these are basic Active Directory attributes that are manipulated with the Set-User cmdlet rather than with the Set-Mailbox cmdlet. If Active Directory management is not the

responsibility of the Exchange administrators, it might be better to rely on attributes that can be changed with Set-Mailbox, just in case an update performed with Set-User might affect some other process such as account provisioning and maintenance.

For the purposes of this example, you select CustomAttribute8, one of the 15 custom attributes Exchange supports, but you will perform an initial population of the attribute by scanning Active Directory to identify likely Irish users. For example, this code scans for a certain code in the Office attribute of user accounts and updates their mailboxes with 'Ireland' in the chosen attribute:

```
Get-Mailbox –Filter {'Office' –eq 'DBO'} | Set-Mailbox –CustomAttribute8 'Ireland'
```

You probably won't pick up every record that should be included in the set the first time around, but this will certainly be sufficient to begin the process. It is a simple matter to generate a report of the mailboxes that are marked with the custom attribute at any time to validate against other sources of information such as human resource lists. It's also important to remember that the accuracy of address lists depends on the underlying data. If you fail to ensure that the properties for new mailboxes are not set correctly, it's likely that users will start to see people who should not appear in the GAL.

Your first address list required by the ABP is created in EMS as follows:

```
New-AddressList -Name "Contoso-Ireland-Users" -RecipientFilter {((RecipientType -eq
'UserMailbox') -or (RecipientType -eq "MailUniversalDistributionGroup") -or
(RecipientType -eq "DynamicDistributionGroup")) -and (CustomAttribute8 -eq
"Ireland")} –DisplayName 'Contoso Ireland Users'
```

This command is far too much bother to type for the other address lists, so you can revert to EAC to create the other address lists. Open the Organization section of EAC, select address lists, and then choose New (+). Figure 7-9 shows the information you need to enter to create the new rooms address list. You can see that a rule has been added to restrict the selected resource mailboxes to those that have the required value in CustomAttribute8. The same steps can be taken to create the Contoso-Ireland-Contacts address list, except that in this case, the chosen recipient type will be mail contacts with external email addresses.

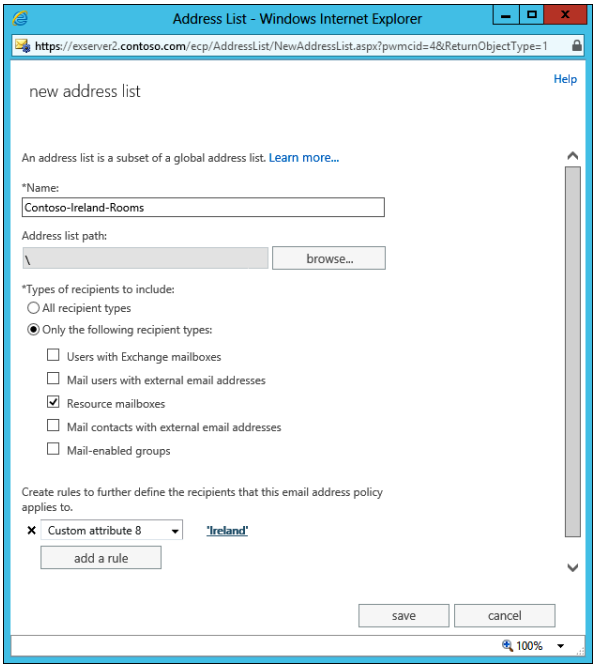


Figure 7-9 Creating a new address list

If you are in any doubt about the effectiveness of the filter used for an address list, you can use the same technique as explained earlier for dynamic groups to test the results Exchange generates when the filter is put into use. For example:

```
$Filter = (Get-AddressList -Identity "Contoso-Ireland-Users").RecipientFilter
Get-Recipient -RecipientPreviewFilter $Filter
```

Alternatively, as explained in the section covering address lists, you can just open the address list with EAC and use its preview facility. Perhaps this is another example of when real administrators use EMS and those who couldn't be bothered with complex commands use EAC!

After you are satisfied with the results generated by the address lists, you can proceed to the next step, which is to create a GAL for use by Contoso Ireland. This has to be a superset of the address lists that effectively combine to form the GAL, so you can create a suitable list by stating a filter that looks for any object with Ireland in CustomAttribute8. You cannot create a new GAL with EAC, so here's the EMS command:

```
New-GlobalAddressList 'Contoso Ireland GAL' -RecipientFilter {(CustomAttribute8 -eq
'Ireland')}
```

The general rule is that after you create a GAL, you should also create an OAB to enable offline clients to access the address lists, so you now create an OAB for Contoso Ireland:

```
New-OfflineAddressBook 'Contoso Ireland OAB' -AddressLists 'Contoso Ireland GAL'
```

You can then generate the new OAB files so that clients can download the OAB with:

```
Update-OfflineAddressBook -Identity 'Contoso Ireland OAB'
```

All the necessary address list components are now available, but you have not yet made them available to users. To do this, create a new ABP that combines the address lists you have created in a form that can then be assigned to mailboxes. Again, this has to be done with EMS. You should see EMS respond as shown in Figure 7-10.

```
New-AddressBookPolicy -Name "Contoso Ireland ABP" -AddressLists "Contoso-Ireland-
Users","Contoso-Ireland-Contacts" -OfflineAddressBook "\Contoso Ireland OAB"
-GlobalAddressList "\Contoso Ireland GAL" -RoomList "\Contoso-Ireland-Rooms"
```

Figure 7-10 Creating a new address book policy

To assign the new ABP to the users in Contoso Ireland, use Get-Mailbox to fetch the full collection of mailboxes in the organization (you cannot perform a server-side filter using the custom attributes), check for any that are marked as belonging to Ireland, and then assign the new ABP:

```
Get-Mailbox -ResultSize Unlimited | Where-Object {$_.CustomAttribute8 -eq 'Ireland'}
| Set-Mailbox -AddressBookPolicy 'Contoso Ireland ABP'
```

Mailboxes that are not assigned the new ABP will continue to use the default ABP (shown as blank), default OAB, and so on. Those users will continue to see the Contoso Ireland users listed in the GAL. In fact, they can also see the new address lists you have created for the Contoso Ireland ABP because they appear under Other Address Lists (Figure 7-11), which then reveals that these entities exist. The Exchange logic is impeccable. The other users are not under the control of any ABP, so by definition, they enjoy access to any address list within the organization.

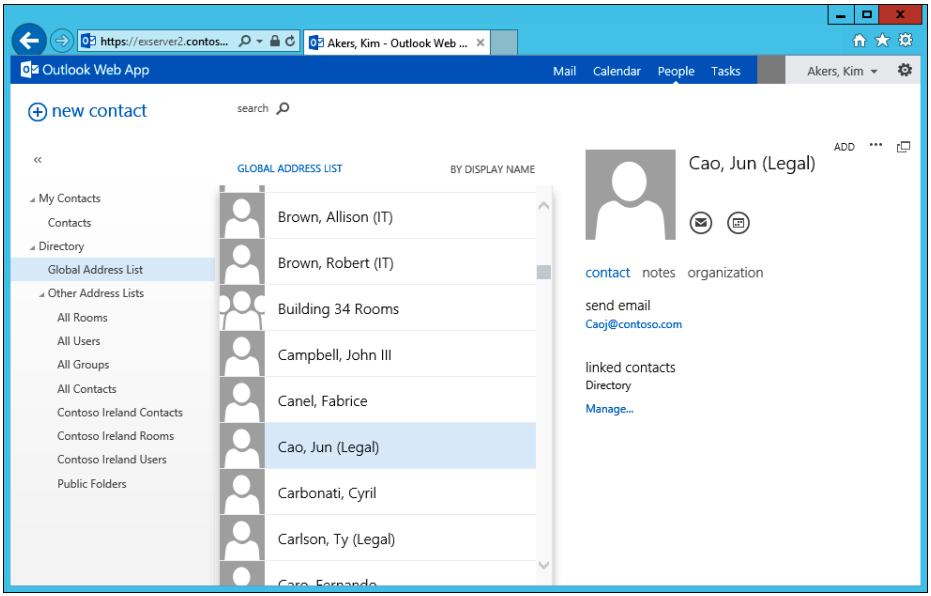


Figure 7-11 The Contoso Ireland address lists available to other users

If this is not the situation you want to exist, you have to create a set of address lists, GAL, and OAB that exclude the Contoso Ireland objects to form another ABP and then assign that ABP to the other users. In effect, every user in the organization should be assigned an ABP that specifically limits her to a certain set of address lists that does not include the default address lists (such as All Users).

If you need to remove a user from a virtual organization, you should update his mailbox to point to a different ABP (or null to use the default) and remove whatever value is driving his membership in the virtual organization. You should also update the GAL and OAB for the virtual organization so that the user disappears from sight. For example:

```
Set-Mailbox -Identity 'Tony Redmond' -AddressBookPolicy $Null -CustomAttribute8 $Null
-OfflineAddressBook $Null
Update-GlobalAddressList 'Contoso Ireland GAL'
Update-OfflineAddressBook -Identity 'Contoso Ireland OAB'
```

To check that the ABP has been assigned correctly, you can run another Get-Mailbox command, using the original filter based on the value of the users' Office attribute and see whether the correct values exist for ABP and CustomAttribute8. Sort the output so that it appears in the same order as the GAL. You should see something like the output shown in Figure 7-12 after running the following command:

```
Get-Mailbox -Filter {Office -eq "DBO"} | Sort-Object DisplayName |  Format-Table
DisplayName, AddressBookPolicy, CustomAttribute8
```

```
Machine: Exserver2.contoso.com
[PS] C:\>get-mailbox -filter {Office -eq 'DBO'} | Sort displayname | format-Table displayname, addressbookpolicy, custom
attribute8

DisplayName                          AddressBookPolicy                    CustomAttribute8
-----------                          -----------------                    ----------------
Alvord, James                        Contoso Ireland ABP                  Ireland
Anderson, Nancy                      Contoso Ireland ABP                  Ireland
Andrews, Ben (IT)                    Contoso Ireland ABP                  Ireland
Brown, Allison (IT)                  Contoso Ireland ABP                  Ireland
Brown, Robert (IT)                   Contoso Ireland ABP                  Ireland
Cook, Kevin                          Contoso Ireland ABP                  Ireland
Cox, Brian                           Contoso Ireland ABP                  Ireland
Daly, Jim                            Contoso Ireland ABP                  Ireland
Evans, John                          Contoso Ireland ABP                  Ireland
Hamilton, David                      Contoso Ireland ABP                  Ireland
Holm, Michael                        Contoso Ireland ABP                  Ireland
Kearney, Bonnie (Sales)              Contoso Ireland ABP                  Ireland
Keen, Charlie                        Contoso Ireland ABP                  Ireland
Keen, Jackie                         Contoso Ireland ABP                  Ireland
Kelly, Kevin                         Contoso Ireland ABP                  Ireland
McCormack, Michael                   Contoso Ireland ABP                  Ireland
Park, Dan                            Contoso Ireland ABP                  Ireland
Redmond, Conor (IT)                  Contoso Ireland ABP                  Ireland
Redmond, Eoin P                      Contoso Ireland ABP                  Ireland
Redmond, Tony (IT)                   Contoso Ireland ABP                  Ireland
Smith, Jane (Operations)             Contoso Ireland ABP                  Ireland
Young, Rob                           Contoso Ireland ABP                  Ireland

[PS] C:\>_
```

Figure 7-12 Checking mailboxes to ensure that the new ABP has been assigned

The newly assigned ABP becomes effective the next time the user connects to the address book service running on a CAS. Outlook Web App sees the most immediate effect because this client usually works in online mode and is forced to connect to the address book service whenever the user browses the GAL. It is then easy to compare the results of the Get-Mailbox command previously described with the GAL shown to the user by Outlook Web App (Figure 7-13). Hopefully, the two sets match up to tell you that the ABP is in place and effective. The user should not be aware that an ABP now controls her ability to see directory information unless she notices that some people who had shown up in the GAL prior to the application of the policy are now missing. Outlook users who work in online mode will also see the new GAL immediately. Those who work in cached Exchange mode must download the new OAB before they can see the effect of the ABP.

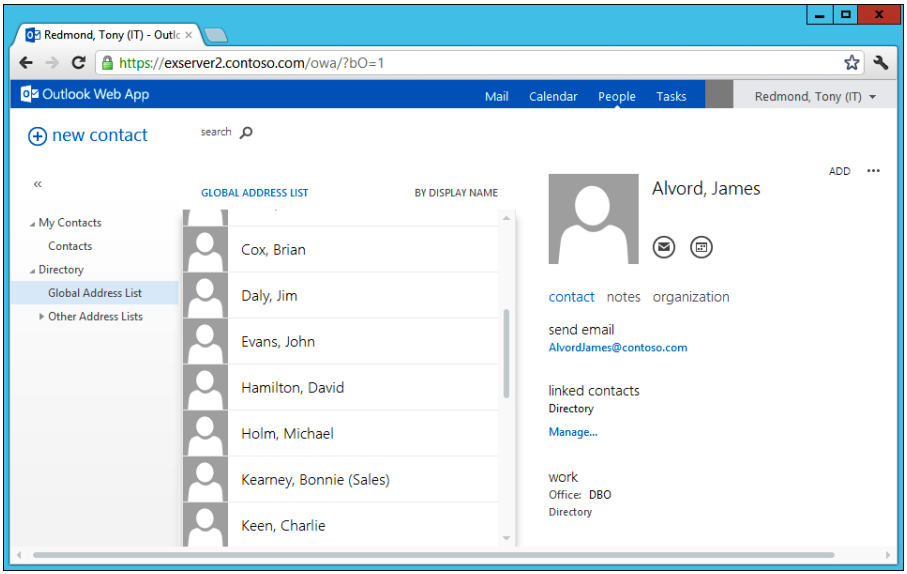


Figure 7-13 Viewing the GAL with Outlook Web App after the ABP is applied

As with most things in IT, it is easier to apply an ABP to a new deployment than to attempt to retrofit a policy on an organization that has been operational for a while. ABPs are not perfect and do not stop users from seeing people in the other virtual organizations if, for instance, they never download an updated OAB or if those people are in their contacts, including the nicknames cache Outlook maintains. ABPs can also cause confusion if users are not educated (or informed by the help desk) about their consequences. For example, after an ABP is assigned to a mailbox, that mailbox loses the ability to send messages on behalf of another user if that user is not included in the address lists the ABP uses. This is logical but can be confusing for a user who doesn't expect it to happen. (How can you send mail on behalf of someone else if she doesn't belong to your virtual organization?)

The Offline Address Book

The OAB is a snapshot of the GAL that Outlook clients can download from Exchange to provide a local directory source for address validation and lookup. All recipients in the GAL except those marked with the Hidden From Address Lists property are included in the OAB (Figure 7-14). You can discover a list of hidden recipients with the following command:

```
Get-Recipient -Filter { HiddenFromAddressListsEnabled -eq $True }
```

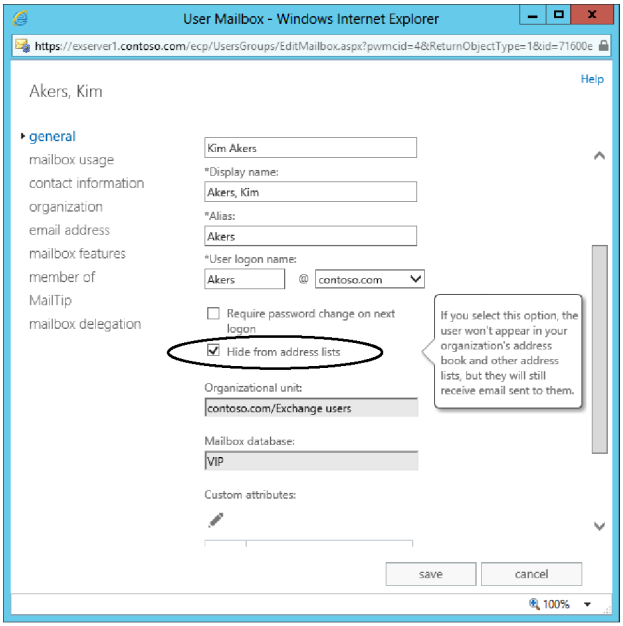


Figure 7-14 Hiding a mailbox from address lists

Outlook clients need to download a copy of the OAB before they are fully functional when working offline. The OAB contains a subset of the properties held for objects in Active Directory. However, users are seldom aware of this fact because the OAB contains all the data they typically need for email or to find recipients. Data that depend on pointers between Active Directory objects are unavailable offline. (Managers and their reports and group membership are the best examples.) The same is true for customized properties you add to the GAL unless you customize address templates to add them to the OAB.

One problem with the OAB is that new recipients are invisible until Exchange next generates updates for the OAB and clients connect, download, and apply the updates to their OAB. For example, if you add a new mailbox at 11 A.M. on Monday, and Exchange generates an OAB update by the following midnight, the earliest a user will see the new mailbox in the GAL is after he connects Outlook to Exchange and downloads the update, which could be the following Thursday or Friday. In reality, this is not usually a problem except when you deal with high-profile users, such as new executives, who want to be visible to the organization as soon as their mailbox is created. There are two workarounds to the problem. First, you can always use the SMTP address of newly created messages to route email to them. Second, you can use the All Users address list to look for new entries because this forces Outlook to connect to the server to browse the directory.

The OAB and Outlook

Outlook clients that are configured in cached Exchange mode download the OAB automatically. The first download occurs after the mailbox folders are replicated to the OST. Thereafter, Outlook checks daily for OAB updates and downloads them from Exchange if available. You cannot vary the daily check; it is hard-coded into Outlook. In any case, it is a good thing to have an automatic daily update because this keeps the OAB up to date and prevents the OAB from accumulating a great deal of invalid information because it hasn't been regularly updated.

> **Tip**
>
> **A week-old OAB is usually okay unless the organization is in the middle of a large merger or acquisition or has another reason for directory churn. A month-old OAB is much less satisfactory because of the volume of change that typically occurs in corporate directories.**

If an OAB update download fails, Outlook retries hourly until the download succeeds. However, if Outlook needs to fetch a full copy of the OAB, it will only try this operation once in a 13-hour period. Clients can also download the OAB on a demand basis by using Send Receive | Download Address Book.

You can minimize the amount of data downloaded by selecting No Details, which causes Outlook to download a basic copy of the OAB (basic recipient information and email addresses). These data still allow the OAB to be used to locate recipients and validate email addresses, but this method produces a file that is much less useful than when it contains data, such as phone numbers, produced by a Full Details download. Limiting data made sense in an era when networks were less available and slower than they are today. In some respects, the option to download a truncated OAB is of limited use unless you are forced to use something such as a dial-up connection.

Like other synchronization operations, Outlook uses a background thread to fetch the OAB files to enable users to remain working while the download proceeds. OAB data used to be made available through a system public folder, but the focus is now on web-based distribution managed by Mailbox servers. Clients with mailboxes on an Exchange 2013 server have to connect to an Exchange 2013 Mailbox server to fetch OAB files, whereas those with mailboxes on earlier Exchange servers are proxied to an Exchange 2007 CAS or Exchange 2010 CAS server.

After you download the OAB, Outlook creates or updates a set of six files on the PC (Table 7-2). These files vary in size, depending on the number of mail-enabled recipients

in the organization, and can occupy a reasonable amount of space on disk. For example, a very large organization requires an OAB of 383 MB to hold approximately 450,000 objects, or around 0.85 MB per 1,000 objects. The OAB files and updates are compressed when Outlook downloads them, and the compressed files are roughly half the size of the files when they are expanded on disk. However, it can still take a long time to fetch a complete OAB inside a large organization.

TABLE 7-2 OAB files

File	Use
UBrowse.oab	The core index for the OAB. Records contain the object type, display name, and a pointer to the rest of the object's data held in the details file.
UDetails.oab	All the details (if available) for objects populated through a Full Details download. This is the largest OAB file.
URdndex.oab	An index used to resolve relative distinguished names for recipient objects and to track changes to domain names.
UPdndex.oab	An index for domain names (such as contoso.com).
UAnrdex.oab	An index used to resolve ambiguous names entered by users when addresses are validated.
UTmplts.oab	A file containing language-specific strings used for dialog boxes and any other static items OAB templates use.

Generally, after the complete OAB is first downloaded, Outlook only needs to refresh it with update files that it fetches from Exchange. These files are generated daily by the OAB Generation Assistant (see the "How Exchange 2013 generates the Offline Address Book" section later in this chapter) and contain the changes that have occurred since the last update. If Outlook has been offline for several days, it needs to download all the daily updates it has missed to update the OAB. A full download is required if Outlook 2003 SP2 (or later) determines that more than half the total entries in the GAL have been updated since the last download. Earlier versions of Outlook required a full download when an eighth of the GAL had changed.

Working in a mixed environment

Consider the situation if 10,000 clients had to download 100 MB of OAB data from a single OAB server. (This is an extreme example to illustrate the point because no deployment of 10,000 clients would use only a single CAS.) The server has to handle the demand for 1,000 GB of data and could have to do this over a short period, which might stop it from doing much other useful work until the demand for OAB downloads subsides.

For the same reason, it's also a good idea to phase in new versions of Outlook so that you don't create a situation in which hundreds of users start their brand-new version of Outlook and immediately begin to synchronize their OST and download the OAB. You can also minimize the impact by providing multiple distribution points for OAB updates by configuring Exchange to distribute the updates to multiple OAB servers.

To offset the potential for server overload, Microsoft uses LZX compression for OAB update files and distributes binary patches only for updated records rather than for complete records. Two files are involved. The Data.oab file is a baseline file. The Binpatch.oab file is generated daily and contains the differential changes from the previous day (essentially, all the changes in the GAL in the past day). To bring an OAB up to date, Outlook downloads all the versions of Binpatch.oab that correspond to the days since the last update and merges them into Data.oab. It then generates the new indexes to refresh the OAB. Compression and binary updates are an effective mechanism to manage OAB distribution for organizations of all sizes.

Figure 7-15 shows what you might see if you examine the directory where Exchange holds the OAB files after they are generated. A file called Oab.xml (the OAB manifest) tracks all the updates that are available, including the full and differential files, templates used by Windows and Macintosh clients, and metadata such as the compressed and uncompressed file sizes.

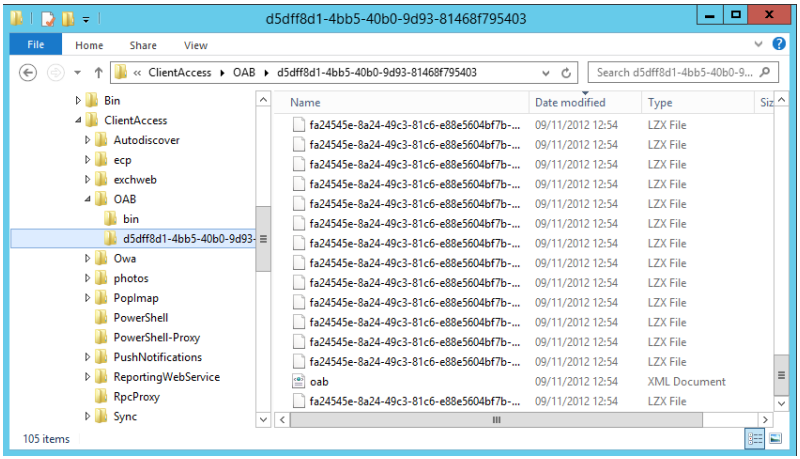


Figure 7-15 OAB data files used for client updates

When Outlook begins to download the OAB files, it flags this fact as part of its synchronization activity. If you use the Ctrl+Click key combination on the Outlook icon in the system

tray to view the client connection status and then click the Local Mailbox tab, you'll see the OAB download progress.

How Exchange 2013 generates the Offline Address Book

Following the redefinition of the Mailbox server role, Exchange 2013 changes the approach to how the Offline Address Book (OAB) is generated. EAC does not include any options to manage the process of OAB generation or distribution, so these tasks are handled through EMS.

In previous versions of Exchange, a specific Mailbox server is assigned the task of generating the OAB. The first Mailbox server installed in an organization automatically takes on the generation of the default OAB and retains it until an administrator reassigns the task to another Mailbox server. For example, in an Exchange 2010 organization, you can find the name of the server that currently generates the default OAB with the following command:

```
Get-OfflineAddressBook -Identity 'Default Offline Address Book' | Format-List Name,
Server
```

This mechanism worked very well for small organizations. However, as organizations expanded to tens or hundreds of servers, perhaps distributed across multiple geographic regions, it became obvious that having a single potential point of failure for a critical component wasn't a good idea, especially when other parts of Exchange had been upgraded to become highly resilient to problems. If the OAB generation server was offline for a period, update files for OAB would not be generated, and clients would start to operate with outdated information. This might not be a problem if the OAB was out of date by a week or so, but after that, users would start to notice that new mailboxes or other mail-enabled recipients were not present in the OAB and that deleted recipients were still present.

Exchange 2013 updates the OAB generation process so that multiple Mailbox servers can generate the necessary files required to keep clients up to date. The default configuration is that a single Mailbox server hosts a special type of arbitration mailbox called the organization mailbox, which stores the OAB update files during the generation process. When generated, the OAB update files are copied to the server directory previously described. Other mailboxes can be created to handle OAB generation if you need to spread the load across the organization. You can identify the current set of organization mailboxes that are used for OAB generation as follows:

```
Get-Mailbox -Arbitration | Where-Object {$_.PersistedCapabilities -like "*OAB*"} |
Format-Table Name, Database, ServerName
```

Outputting the server name is accurate only if the mailbox is located in a database that is not hosted within a DAG. When databases are part of a DAG, the active copy of the database that holds the OAB generation mailbox can move within DAG members. To know the server on which the active database copy is currently located, you have to run the Get-MailboxDatabaseCopyStatus command.

Like any location, the database in which the OAB generation mailbox is held might need to change over time. For instance, you might want to delete the database. Exchange won't allow you to delete a database while it holds any mailboxes, so you must move the OAB generation mailbox first to allow the database deletion to proceed. Assume that the database you're dealing with is DB1. The following command searches for OAB generation mailboxes in the database and moves them to database DB2.

```
Get-Mailbox -Arbitration -Database DB1 | Where-Object {$_.PersistedCapabilities -like
"*OAB*"} | New-MoveRequest -TargetDatabase DB2
```

This command works for both single-copy databases and those protected by a DAG. It also does the job when you want to move the OAB generation mailbox because you think the OAB load should be handled by another server, when all you need to change in the command is to specify a database on the target server.

Default OAB operation

The default organization mailbox used for OAB generation is called *SystemMailbox{bb558c35-97f1-4cb9-8ff7-d53741dc928c}*, a name that is probably unique in any organization. This mailbox is created when the first Exchange 2013 server is installed in an organization. How to create additional organization mailboxes with OAB generation capability is discussed in the "Creating new OAB generation mailboxes" section later in this chapter. Because the organization mailbox can be stored in a mailbox that's protected by a DAG, it is more resilient than the previous mechanism.

After the OAB update files have been generated, the assistant copies them from the organization mailbox to a folder under *Program Files\Exchange\V15\ClientAccess\OAB*. Each set of files for an OAB is held in a separate folder identified by the internal globally unique identifier (GUID) Exchange uses to identify the OAB.

Another change is that the role of the old System Attendant process (which doesn't exist in Exchange 2013) in OAB generation is replaced by a dedicated mailbox assistant called OABGeneratorAssistant (a process within the Microsoft Exchange Mailbox Assistants service). Like the other mailbox assistants, OABGeneratorAssistant runs according to a throttled workload policy that enables it to take advantage of resources whenever the Mailbox server is under low demand. Unlike the previous situation, which aimed to generate OAB updates at a specific time every day, the OABGeneratorAssistant runs on a work cycle basis, so OAB generation proceeds as system resources allow. The work cycle is defined as properties of the Mailbox server that hosts the OAB mailbox. The OABGeneratorWorkCycle property determines the length of the work cycle, and its default value is one day. The OABGeneratorWorkCycleCheckPoint property determines the interval between OAB generation runs and, again, the default is one day. Combined, the two properties instruct Exchange to generate OAB updates for clients on a daily basis.

If necessary, you can change the work cycle to have OAB updates generated more or less frequently. A settled organization that does not add new mailboxes or other mail-enabled objects very often might find it reasonable to have OAB updates generated weekly. To do this, you'd update the Mailbox server properties as follows:

```
Set-MailboxServer –Identity ExServer2 –OABGeneratorWorkCycle 07.00:00:00
–OABGeneratorWorkCycleCheckPoint 07.00:00:00
```

However, if your organization is in a state of constant flux with new mail-enabled objects being created on a very frequent basis, and you wanted to have OAB updates generated twice daily, you'd run a command like this:

```
Set-MailboxServer –Identity ExServer2 –OABGeneratorWorkCycle 01.00:00:00
–OABGeneratorWorkCycleCheckPoint 12:00:00
```

You can tell when the assistant has last processed an OAB by examining the LastTouchedTime property. For example:

```
Get-OfflineAddressBook –Identity 'Default Offline Address Book' | Select Name,
LastTouchedTime
```

```
Name               : Default Offline Address Book
LastTouchedTime    : 01/11/2013 13:53:04
```

In this case, you know that the default OAB was last updated by OABGeneratorAssistant on 11 January 2013 at 13:53. Although the OABGeneratorAssistant is designed to get new or updated objects into the OAB reasonably soon after changes occur, you can generate the OAB files manually at any time by using the Update-OfflineAddressBook cmdlet. For example:

```
Update-OfflineAddressBook –Identity 'Default Offline Address Book'
```

Exchange logs event 17002 in the Application Event Log when the new OAB files are available.

INSIDE OUT Make sure you have full visibility for Active Directory

When you work with address books, you probably want to include mail-enabled objects fetched from across the entire directory. This happens automatically when you work inside a single-domain forest, but things are more complex when multiple domains are involved. To make sure that you have visibility across the entire forest, run the Set-ADServerSettings –ViewEntireForest $True command.

As with previous versions, the OABGeneratorAssistant reads information from Active Directory by using the Name Service Provider interface (NSPI) when it creates its updates. Although the address lists as displayed in EAC might indicate that they are up to date, this only means that the task of defining the correct set of objects to appear in an address list is complete. It does not imply that the OABGeneratorAssistant has finished creating new OAB files for clients to download, so it's always best to run the Update-OfflineAddressBook cmdlet when the need exists to be sure that an OAB is updated and ready for download.

For more information about ABPs and how address lists can be combined into a customized OAB that is assigned to specific users through policy, see the "Creating and implementing an ABP" section earlier in this chapter.

What's in the OAB

You can manage the OAB generation parameters with EMS. To begin, you can examine the many interesting details of the OAB by running the Get-OfflineAddressBook cmdlet:

```
Get-OfflineAddressBook -Identity 'Default Offline Address Book' | Format-List
```

When you examine the output from Get-OfflineAddressBook, you can see the following:

- The name of the Mailbox server that generates the OAB, blank because this is an Exchange 2013 server.
- The address lists included in the OAB. In this case, just the set of objects specified in the default GAL is included.
- The versions of the OAB that are generated. Because it does not support clients like Outlook 2003 that depend on older format OABs, Exchange 2013 only generates version 4 type OAB files.
- The date and time of the last OAB update (LastTouchedTime).
- The number of records in the OAB (LastNumberOfRecords). This should approximate the number of mailboxes, groups, contacts, and mail-enabled public folders in the organization that are not hidden from address lists.
- The attributes included for each OAB record (configured attributes). Some of these support ambiguous name resolution (ANR), meaning that you can search the OAB based on partial matches of attributes such as Surname and Office. For example, you can find all the users in the London office by typing Lond in the OAB search dialog box. (Click the More Columns button to force the OAB to perform more than the default name search.) Because it's an ANR search, the OAB also returns users whose surname begins with Lond. Other attributes contain simple values, such as the MailTips attributes.

- How long (30 days) different files are kept on the server.
- The virtual directories on the server that support web-based distribution of the OAB files.

To see a complete list of OAB attributes, you can do this:

```
$LA = Get-OfflineAddressBook -Identity '\Default Offline Address Book'
$L2 =$LA.ConfiguredAttributes
$L2 | Format-Table -AutoSize
```

You can customize attributes that are included in an OAB by specifying the desired set with the Set-OfflineAddressBook cmdlet. For example, a simple OAB that contains just three attributes could be generated as follows:

```
Set-OfflineAddressBook -Identity 'Default Offline Address Book'
-ConfiguredAttributes 'Surname, ANR','GivenName, ANR', 'Account, ANR'
```

Each attribute is defined with its Messaging Application Programming Interface (MAPI) property name and its type, which must be one of ANR, Value, or Indicator. The correct type for each attribute is listed in the preceding output. If you need to revert to the default situation, you can do this with:

```
Set-OfflineAddressBook -Identity 'Default Offline Address Book' -UseDefaultAttributes
```

Client access servers maintain an OAB virtual directory to specify the URLs for internal and external clients to connect to download the OAB. The internal URLs are created automatically and are of the form *https://exserver1.contoso.com/OAB*. If external access is required, the *ExternalURL* property must be configured by running the Set-*OABVirtualDirectory* cmdlet to point to the correct URL. The Set-OfflineAddressBook cmdlet can be used to set the *GlobalWebDistributionEnabled* property to $True, which means that the settings for OAB virtual directories on all CAS servers in the organization will be set automatically. If this property is set to $True, you won't be able to configure the settings for the OAB virtual directory on an individual server.

Creating new OAB generation mailboxes

When an Exchange 2013 CAS receives a client request to download the OAB, it proxies the request to the nearest Mailbox server that hosts an OAB generation mailbox. By default, all OAB download requests therefore go to a single Mailbox server because only one OAB generation mailbox exists in the organization. This is an acceptable situation when the organization is small or connected with very capable network links so that client download requests from around the organization can be handled by the one server.

In larger organizations or when you have scenarios such as hub-and-spoke networks, you might want to distribute OAB generation by creating multiple OAB generation mailboxes.

For example, you could have an OAB generation mailbox in every Active Directory site so that all the OAB download requests are kept local within each site. A large site might host more than one OAB generation mailbox; if so, the CAS will round-robin client requests between the servers that host the mailboxes.

To create a new OAB generation mailbox, you first run the New-Mailbox command to create the mailbox in a database within the chosen site and then run the Set-Mailbox command to mark it as an OAB generation mailbox. For example:

```
New-Mailbox -Name 'OAB Generation Mailbox Dublin' -Database DB1-Dublin
-UserPrincipalName OABGenerationDublin@contoso.com -DisplayName
'OAB Generation Mailbox Dublin' -Arbitration
Set-Mailbox -Identity 'OAB Generation Mailbox Dublin' -OABGen $True -Arbitration
```

You'll be asked to confirm that you want to set the mailbox to be used for workflow purposes. When you do, the mailbox is marked as being suitable for OAB generation, and the OAB Generation Assistant begins to use it. Marking the mailbox as an arbitration mailbox also hides it from recipient lists. Now when you run the command to report on OAB mailboxes, you should find your new mailbox listed:

```
Get-Mailbox -Arbitration | Where-Object {$_.PersistedCapabilities -like "*OAB*"}
```

To validate that client requests are being handled correctly after you have deployed multiple OAB generation mailboxes, you can check the HTTP proxy logs on the CAS servers to discover which Mailbox servers and mailboxes are used. The log is found in *Exchange Server\V15\Logging\HttpProxy\Oab*, and the important fields to look for are the following:

- **AnchorMailbox** Contains the distinguished name of the mailbox to which the CAS server proxied the request
- **TargetServer** The name of the Mailbox server holding the OAB mailbox

Creating and using customized OABs

The foundation for the OAB is provided by a set of one or more address lists. The default OAB contains just one address list, the default GAL, so any object that is included in the GAL is also included in the default OAB. An address list is nothing more than a convenient way for Exchange to focus on a set of mail-enabled Active Directory objects by applying a filter to locate the objects. The filters are very similar to those used to create the membership of dynamic distribution groups. Each address list has a filter, and when Exchange generates an OAB, it applies the filter for each address list that is included in the OAB. Thus, when Exchange generates the default OAB, it applies the default GAL filter to find all the mail-enabled objects in the organization that are not marked to be hidden from address lists.

You can change the OAB that is provided to Outlook clients in two ways. First, you can use ABPs to assign customized versions of the GAL to users. (See the description earlier in this chapter for more information on ABPs.) Second, you can create one or more OABs and assign these to specific users by assigning an OAB to a database so that every mailbox in that database uses that OAB, or you can do it by individual mailbox.

If your Exchange organization supports a common set of users that everyone needs to know, there is no need to replace the default OAB because it will serve the purpose of offline access to the directory well. Nevertheless, hosting companies that deliver Exchange services to other companies use address lists to provide a customized OAB for each company of the users that belong to one.

Subsidiaries with a common Exchange infrastructure

Large companies that support different subsidiaries with a common Exchange infrastructure often use the same approach as hosting companies so that the users belonging to each subsidiary see only the users from that subsidiary in their OAB (and perhaps users from common shared corporate departments). This is accomplished by creating new address lists to filter out the user communities and then using the address lists to generate different OABs, which are then assigned to users.

Before you get too far into creating a new OAB, remember that OABs are only useful to Outlook clients that work in cached Exchange mode. Clients that work online have access to the complete GAL unless they are constrained by ABPs. The discussion presented here focuses on the steps an Exchange administrator has to execute to create a customized OAB for specific mailboxes when she does not want to deploy ABPs for some reason.

Your first task is to create a suitable address list. You can create a new address list that contains all the mail-enabled objects for the location or create an address list for each object type (contacts, rooms, equipment, mailboxes, and groups) that are then combined in the OAB you eventually generate. Exchange *maintains* default address lists such as All Rooms or All Contacts to make it convenient to locate these objects for different purposes within the product. In this case, multiple address lists would probably overcomplicate matters, so to keep things simple, create a single address list. Details of how to create address lists by using EAC are described in the "Address lists" section earlier in this chapter, and more information about how address lists can be formed into an OAB is presented in the "Creating and implementing an ABP" section earlier in this chapter. Those instructions won't be repeated here, but assume that you have created a new OAB such as the Contoso Ireland OAB described earlier in this chapter.

When you create a new OAB, Exchange creates a directory to hold the files clients download to access the OAB under the \V15\ClientAccess\OAB root (Figure 7-16). Each OAB holds its files in a separate directory, naming the directory after the GUID for the OAB. By looking at Figure 7-16, you know that this organization supports two OABs. To see the GUIDs for the OABs so that you can associate the OABs with their directory, run the Get-OfflineAddressBook command that follows. You can see from the output how easy it is to locate the files for a particular OAB.

```
Get-OfflineAddressBook | Format-Table Name, GUID -AutoSize
```

```
Name                                    Guid
----                                    ----
Default Offline Address Book            682524d0-c92b-400b-8e10-474a0f43cf03
Contoso Ireland OAB                     75027a4f-4d1a-4d8a-a662-003a4bd66d37
```

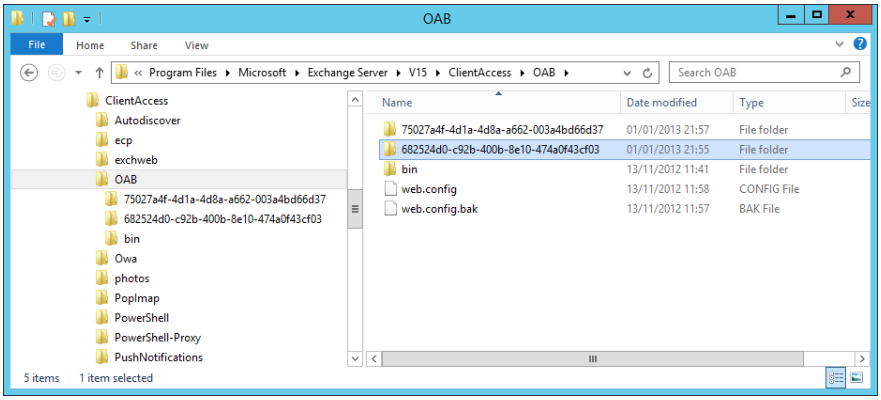


Figure 7-16 OAB directories on an Exchange 2013 server

A particular OAB can be assigned to individual mailboxes or to all the mailboxes in a database. Use the Set-Mailbox cmdlet to assign an OAB to a mailbox. For example:

```
Set-Mailbox -Identity 'David Jones' -OfflineAddressBook 'Contoso Ireland OAB'
```

After a specific OAB has been assigned to a mailbox, anytime that the user attempts to download the OAB with Outlook, he'll see that the OAB presented by Outlook (Figure 7-17) is based on a different GAL. Then again, he might not notice that such a subtle change has occurred!

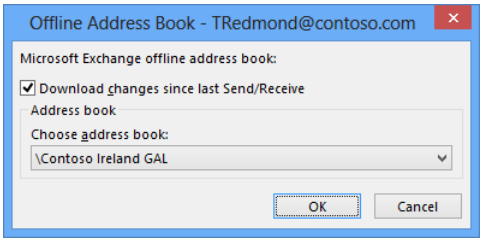


Figure 7-17 Selecting an OAB to download with Outlook

You replace Set-Mailbox with Set-MailboxDatabase to make the assignment for a mailbox database:

```
Set-MailboxDatabase -Identity 'DB1' -OfflineAddressBook 'Contoso Ireland OAB'
```

Alternatively, you can select the database from EAC and access the Client Settings tab (Figure 7-18) to select the OAB there. If an OAB is not explicitly selected for a mailbox database, Exchange uses whatever OAB is the current default. To make a new OAB the default across the organization, you use a command like that shown here. Only one OAB can be marked as the default at one time.

```
Set-OfflineAddressBook -Identity 'Contoso Ireland OAB' -IsDefault $True
```

The steps presented here create and distribute a new OAB to users, but this won't prevent them from accessing the default GAL when they work online with either Outlook or Outlook Web App. If you want to block access to other address lists, you have to configure and implement ABPs as explained earlier in this chapter.

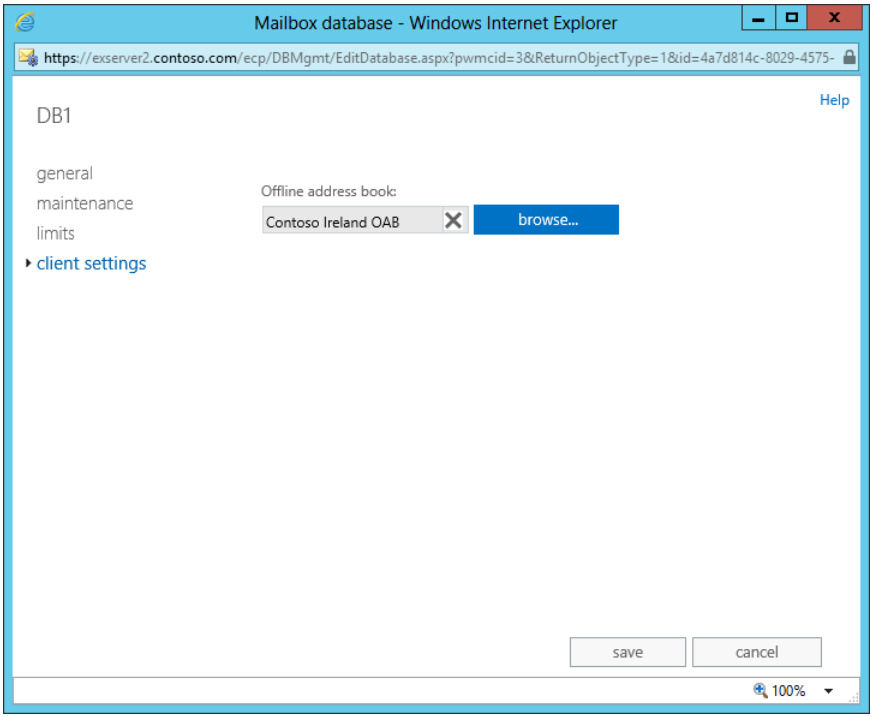


Figure 7-18 Setting a new OAB for a mailbox database

The conditions you can't account for offline

Even with access to the OAB, some conditions exist that are impossible to account for offline, in which case you won't see a MailTip or know something about a recipient. For example, a recipient mailbox might exceed its quota for a short time, a new mailbox might be added to Active Directory and isn't included in the OAB until the next time the client downloads an OAB update, or a user might add an out-of-office (OOF) notice. The MailTip for invalid internal recipient (perhaps an address for a mailbox that has recently been deleted) is also unavailable offline.

Hierarchical address book

The standard address book meets the needs of most companies throughout the world. Presenting users and groups sorted by first name or last name does not work so well in companies where the culture is strongly based on adherence to a hierarchical structure, especially in East Asian countries. In these situations, the desired result is to present the

organization in terms of seniority so that the most important members are listed first. Users can then search down through the address book of the organization, following its structure. For example, they can move from the CEO to the vice presidents who run major departments, to the directors, managers, and then individual contributors.

From Exchange 2010 SP1 onward, any Outlook client later than Outlook 2007 SP2 can display a hierarchical address book (HAB) if the necessary steps are taken to establish the hierarchy within Exchange. You can follow these steps to create a new HAB. Steps 1 and 3 are optional insofar as you don't strictly need to use a dedicated OU for the HAB. However, its use enables you to maintain a clear separation between the objects that form the HAB and the other Exchange groups and users. Steps 4 and 5 can be combined into a single set of Set-Group commands. They are isolated here to emphasize the need for the two distinct properties that must be set to allow objects to be included in the HAB.

1. Create an OU in Active Directory to act as container for the groups and user objects that will form the HAB.

2. Configure the organization with the alias of the group that will act as the root of the HAB. In this case, mark a group containing the CEO's email address as the root of the HAB:

   ```
   Set-OrganizationConfig –HierarchicalAddressBook DG-CEO1
   ```

3. Add the groups and users that will form the HAB into the OU. Again, this is an optional step.

4. Modify each group object that will be in the HAB to set its IsHierarchicalGroup property to $True.

   ```
   Set-Group –Identity DG-CEO1 –IsHierarchicalGroup $True
   ```

5. Modify each group and user object that will be in the HAB to set its SeniorityIndex property to an integer value that Exchange will use to sort the HAB. An object with a SeniorityIndex value of 1 will be at the top of the HAB, so if your CEO represents the top of the organization, you'd set this value to 1 for his object (or for the group object that might contain the CEO and members of his staff). Groups and users that occur farther down the hierarchy are assigned higher SeniorityIndex values.

   ```
   Set-Group –Identity DG-CEO1 –SeniorityIndex 1
   Set-Group –Identity DG-Senior-VP2 –SeniorityIndex 2
   Set-Group –Identity DG-VP3 –SeniorityIndex 3
   ```

6. Ensure that groups of each level include the groups of the next level down in their membership so that the members of each level are shown when a user navigates through the hierarchy. For example, the HAB shown in Figure 7-19 uses organizational titles from the CEO downward as the basis of its hierarchy. To create this view, the Senior Vice Presidents group is a member of the Chief Executive Officer

group, the Vice Presidents group is a member of the Senior Vice Presidents group, and the Senior Managers group is a member of the Vice Presidents group, with the same arrangement based on seniority continuing for the remainder of the hierarchy.

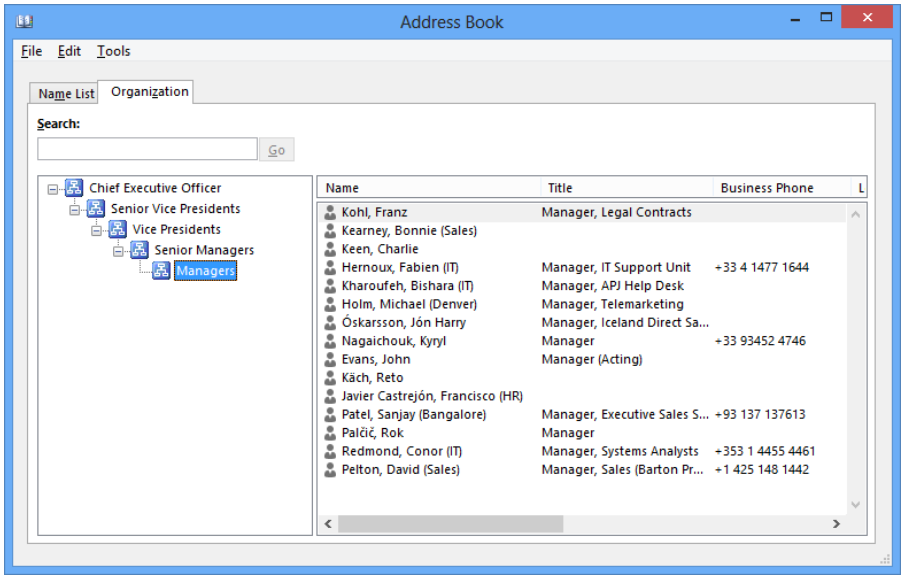


Figure 7-19 Viewing a HAB

7. To make the HAB more accessible, you can also set the PhoneticDisplayName property for groups and users. Outlook sorts the HAB first by the SeniorityIndex and then by PhoneticDisplayName.

```
Set-Group –Identity DG-HAB-Group1 –PhoneticDisplayName 'HAB Group 1'
Set-User –Identity User-HAB1 –PhoenticDisplayName 'User 1'
```

8. Open a supported version of Outlook and open the Address Book. Click the Organization tab to reveal the HAB.

Only a single HAB can exist in an organization, and HABs can be viewed only by Outlook clients that are configured to use cached Exchange mode and that have successfully downloaded an OAB that was generated after the properties of the groups that form the HAB were updated as just described.

MailTips and group metrics

All of us have stories about an unbelievable but all too common mistake we have made with email. Maybe you sent a message to the wrong user containing sensitive or other information that she shouldn't have received, or maybe you answered a note on which you

were blind–carbon copied and caused the original sender trouble when the other recipients realized that you were copied. Microsoft refers to situations like this as "unfortunate messaging scenarios," whereas administrators might use slightly more robust terms to describe the result of user mistakes that cause system performance to suffer, mail queues to accumulate, or lots of calls to flow in to the help desk. It's unfortunate, but humans make mistakes all the time.

Early email systems operated on a simple "send and forget" principle. You dispatched messages into the void and hoped that they'd arrive. As time went by, features such as out-of-office notices (OOF), delivery and read receipts, and nondelivery notifications were incorporated to give users feedback about what happened to messages after they were sent. Exchange 2010 introduced delivery reports, which enable users to see the path of a message within an Exchange organization, including the expansion of distribution groups into individual members to receive the message, delivery to servers, and delivery to external connectors for transmission to other email systems. However, worthy as these features are, they are all after the fact. Messages have to be sent before users can be informed that the message has been delivered, that someone is out of the office, or that their email won't be delivered because a recipient's mailbox quota is exceeded.

MailTips are designed to provide users with feedback about common problems before a message is sent. The idea is that warning users that messages might not be delivered or read will make users more productive and less likely to call the help desk to complain that their emails didn't arrive. It will also reduce system resource usage by eliminating the need to process messages that will fail. Microsoft hopes that MailTips will help people use email more intelligently. The feature is supported by Outlook Web App (including Outlook Web App for Devices) and Outlook 2010 or Outlook 2013 when connected to an Exchange 2010 or Exchange 2013 mailbox. Quite logically, MailTips won't work when Outlook is not connected to Exchange.

MailTips operate on a mixture of data originating from the following sources:

- Active Directory (for example, whether a recipient is restricted, the maximum size of attachments supported by the organization)
- Information Store (mailbox quotas, out-of-office notices, custom MailTips)
- Group metrics (metadata generated for the total number of recipients in a group and the number of external recipients in a group)

Group metrics information is stored in the group object in Active Directory in the msExchGroupMemberCount (number of members in the group) and msExchGroupExternalMemberCount (number of external recipients in the group) and is calculated for both normal and dynamic groups. Figure 7-20 shows the properties of a group as viewed through ADSIEdit with both the external count (0) and member count (82) clearly visible.

Using Active Directory to hold group metric data is a change from Exchange 2010, which instead uses a cache on the CAS.

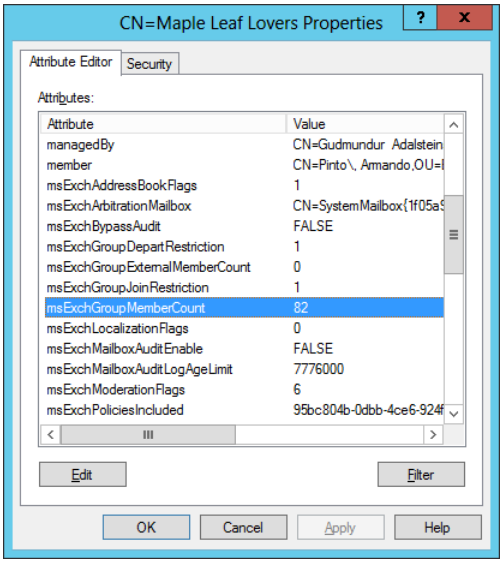


Figure 7-20 Count of group members stored in a group object in Active Directory

The Group Metrics mailbox assistant runs on every Mailbox server that is configured to generate group metrics data to count the number of recipients in every group in the organization, including dynamic distribution groups. On the basis that user workload is light during the weekend, Exchange 2010 servers generate group metrics data on a Sunday. Exchange 2013 uses the work cycle approach that's seen in other mailbox assistants to run daily at a time when system workload is light and its activity will not affect users. If system workload increases when the Group Metrics mailbox assistant is active, it throttles back and waits until the workload eases. If a server is under consistently high load over a complete day, the generation of group metrics data will be delayed. However, given that only a few groups might be added or have their membership updated during this period, the overall effect on the accuracy of the data shown to users through MailTips is limited.

By default, every Mailbox server that generates the OAB also generates group metrics data. OAB servers generate group metrics data because the OAB reuses some MailTips data (including group metrics), so it's necessary to generate the data before it can be included in the OAB.

To discover which servers generate group metrics data, run this command:

```
Get-MailboxServer | Select Name, ForceGroupMetricsGeneration
```

Any server that reports True for ForceGroupMetricsGeneration is configured to generate group metrics. You can configure other Mailbox servers to generate group metrics by running the command. Normally, this isn't necessary, but it is conceivable that you want to spread the generation load across the organization.

```
Set-MailboxServer –Identity ExServer1 –ForceGroupMetricsGeneration $True
```

On the other days of the week, the service generates incremental data for the number of recipients in groups that have been added or changed. Event 14034 is logged under the MailTips category when a server begins to process group metrics data. If you examine these event entries, you can find out how many groups were processed and whether the run was the weekly full scan or a daily delta.

The *\Exchange Server\V15\GroupMetrics* directory on a Mailbox server holds the files used for group metrics. These are:

- A cookie file with a .dsc extension that contains information to identify the Mailbox server that generates the group metrics data and the date and time when the data was last successfully updated. By comparing the date and time held in this file with the current date and time and the last changed time stamp on group objects, Exchange knows which groups it has to process to derive updated group metrics data.

- A simple text file called ChangedGroups.txt. After a successful update of group metrics data, the ChangedGroups.txt lists any group that Exchange processed. It is a simple list of the distinguished name for each group. For example:

  CN=Dublin Users,OU=Exchange groups,DC=contoso,DC=com

When Exchange generates group metrics data, it first identifies the set of changed groups and then counts the number of recipients in each one of these groups. The resulting count is then written into the properties of the group in Active Directory.

Client interaction

MailTips-enabled clients request data when a user adds a recipient or attachment to a message or uses the reply or reply all command to respond to a message. MailTips are also processed when a draft message is opened. These actions cause the client to invoke a background thread to query the MailTips web service on a CAS server on the local site and determine whether any MailTips apply to the message. The URL for the MailTips web service is returned by the CAS server to the client as part of the Autodiscover manifest.

The CAS server is responsible for gathering data from Active Directory and the local group metrics cache and responding with applicable data to the client. It also contacts the Mailbox servers that host recipient mailboxes to fetch information about mailbox quotas

and OOF notices. If some of the Mailbox servers are outside the local site, the CAS proxies a request to fetch the data to a Mailbox server running on the site that hosts the mailboxes.

> **Note**
>
> **To avoid unacceptably slow responsiveness, if the Mailbox server is unable to respond to clients within 10 seconds, the request times out, and the client proceeds without MailTip data. It's also important to say that you do not have to wait for MailTips processing to finish before you can send a message. If MailTips hasn't responded and you need to get the message to its recipients, you can click Send, and the message will go without further notice.**

To limit the amount of communication with the CAS, Outlook and Outlook Web App clients both maintain client-side caches that are populated as messages are processed. For example, if you attempt to send a message containing a very large attachment, the client checks its cache to discover whether it exceeds the maximum message size for the organization. If the data are not in the cache, the client retrieves them from the appropriate source (in this case, Exchange configuration data in Active Directory) and caches it locally for future reference. Cached information is aged out of the cache after 24 hours with the exception of mailbox full and OOF notices, which are likely to change more often and therefore age out after two hours. The MailTips settings used for Outlook Web App are contained in the Web.config.xml configuration file. It is possible, but not recommended, to change the cache aging limit and the number of items held in the cache by editing this file. Client-side caches are not persistent and are cleared whenever you exit Outlook or Outlook Web App.

Configuring MailTips

From an administrator perspective, the immediate value of MailTips is that they eliminate many of the reasons Exchange has to generate nondelivery reports (for example, destination mailbox is full, message size too large) and so reduce the strain on the messaging infrastructure. On the downside, the processing required to service client requests for MailTip data creates an extra load on CAS servers, which Microsoft characterizes as an increase of approximately 5 percent.

All the management for MailTips is done through EMS. Exchange administrators can configure MailTips on or off, but only at the organization level, by using the Set-OrganizationConfig cmdlet to set the parameters that control MailTips processing. These parameters are as follows:

- **MailTipsAllTipsEnabled** Controls whether MailTips are enabled. The default is $True.

- **MailTipsExternalRecipientTipsEnabled** Controls whether MailTips for external recipients are enabled. The default is $False. External recipients are determined by reference to the accepted domains list. Any domain in this list is deemed internal, whereas any other is deemed external.
- **MailTipsGroupMetricsEnabled** Controls whether MailTips that depend on group sizes are enabled. The default is $True.
- **MailTipsLargeAudienceThreshold** Controls the threshold for the number of recipients on a message before MailTips flags it as large. The default value is 25. This value is probably too low for large organizations in which big distribution groups are common. In this scenario, it makes sense to increase the value to 50 to stop MailTips from nagging users.
- **MailTipsMailboxSourcedTipsEnabled** Controls whether MailTips that depend on mailbox data such as OOF notices are enabled. The default is $True.

The website at *http://technet.microsoft.com/en-us/library/jj649091(v=exchg.150).aspx* lists the common conditions when MailTips are useful. In most cases, a MailTip stops a user from doing something that causes frustration (the user's message doesn't get through) or reduces traffic for Exchange (removing the need for nondelivery notifications caused by messages that cannot be processed). Warning users about sending to very large distribution lists is useful because it is too easy for a user to address a note to a distribution that she doesn't realize will cause Exchange to deliver the message to thousands of mailboxes. People who receive the message can cause a mail storm by using Reply All to generate a response that goes to everyone and provokes even more responses. The net result is usually mailbox servers that are swamped with traffic, the accumulation of large message queues, and slow delivery of other mail. MailTips won't remove the need for user intelligence, but they can give a hint at appropriate times to stop someone from doing something he wishes he hadn't.

User experience

You cannot manipulate the MailTips settings for a mailbox through EAC or EMS, so clients display MailTips according to whatever is available in the UI. If configured in cached Exchange mode, Outlook depends on details of MailTips that are held with other recipient attributes in the OAB, but when it is configured to work online, Outlook fetches MailTips data from the Mailbox server, which is also the way Outlook Web App works.

Figure 7-21 illustrates how Outlook 2013 and Outlook Web App display Mail Tips. (MailTips are also supported in the same way by Outlook 2010, but the necessary UI for MailTips is not included in Outlook 2007.) In this example, the message is addressed to three groups, each of which results in the display of a different MailTip. First, you are told that you don't have the necessary permission to send a message to the HR Management group (because

the group has been limited in terms of the senders from which it will receive messages). Second, you see that sending a message to Maple Leaf Lovers will result in delivery to 82 recipients, which is higher than the 25 limit set for the large audience threshold. (An even larger group is used with Outlook Web App.) Finally, sending the message to the CEO Office could be an exercise in futility because the message will be moderated and might not be accepted.

Configurable options in Outlook 2010 and Outlook 2013 enable users to configure the type of MailTips to see, whether to have the MailTips bar available at all times, and whether to expand the MailTips bar if multiple tips are available. For example, if your role involves the need to send messages to large groups on a regular basis, you might decide that there is no purpose in Outlook displaying this MailTip. Outlook Web App does not include any user-controllable options to control how MailTips appear.

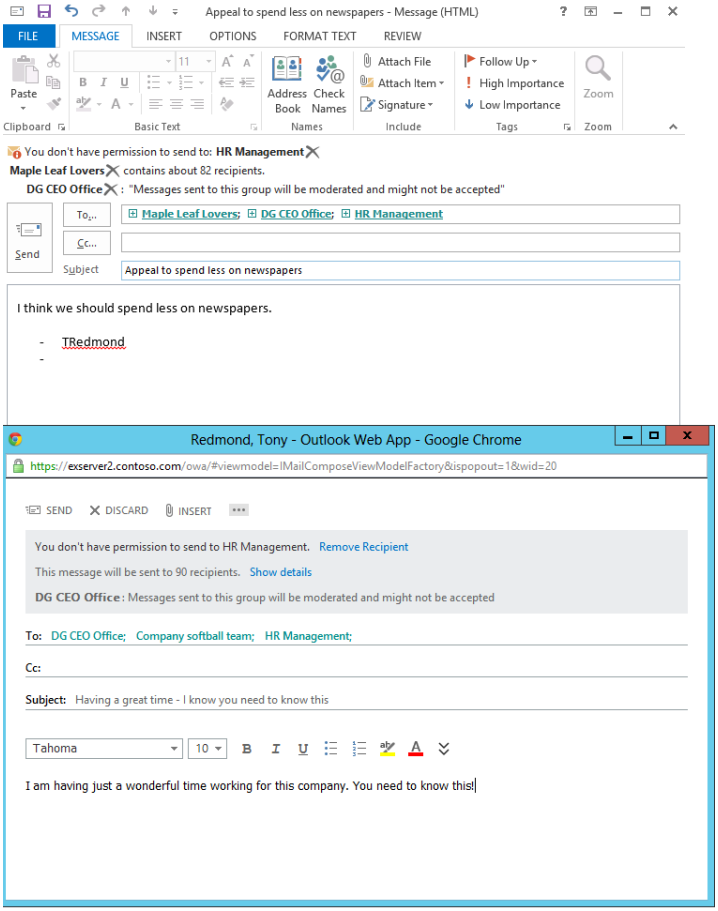


Figure 7-21 How clients display MailTips

Custom MailTips

You can add a custom MailTip to any mail-enabled object. Most commonly, this is done for mailboxes that are not monitored to notify users that their message might not be responded to quickly; to moderated addresses; to restricted distribution groups (including dynamic groups) that don't accept messages from users who aren't on an approved list; and to provide some guidance to users when they send to special mailboxes such as those used by help desks to set expectations about when the sender might expect a response.

Custom MailTips can be up to 175 characters. Given that many people are Twitter literate and capable of expressing profound thoughts in 140 characters or fewer, creating a suitable custom MailTip in 175 characters should not pose any real difficulty in terms of composition. The only issue is knowing where the limit exists. EAC does not attempt to count characters as you enter a new MailTip and signals a problem only when you attempt to save a MailTip that's too long. It's therefore a matter of trial and error to perfect a MailTip. EMS is equally unhelpful.

Custom MailTips are configured in exactly the same way as any other property for a mail-enabled object. For example, to create a MailTip for a mailbox, you use the Set-Mailbox cmdlet:

```
Set-Mailbox –Identity 'APJ Help Desk' –MailTip 'Messages to the APJ Help Desk are
handled on a best-effort basis; please call 91184 if you need urgent support'
```

Use the same approach for a distribution group:

```
Set-DistributionGroup –Identity 'Sales' –MailTip 'Only members of the Sales Execu-
tives group can send to this address'
```

MailTips are stored in Active Directory in HTML format. Therefore, you can input HTML when you configure a custom MailTip. Being able to use HTML is useful when you want to include a URL to point users to more information. For example, messages sent to the help desk might include a URL to enable users to log a support call:

```
Set-Mailbox –Identity 'EMEA Help Desk' –MailTip 'Please visit the Help Desk site <A
href = "http://help-desk-support.contoso.com" </A> to log a support call'
```

INSIDE OUT What you cannot do

You cannot edit a custom MailTip. Instead, you overwrite it with new text. Exchange 2013 and Exchange 2010 support only simple HTML text in a custom MailTip, so you can't incorporate any form of rule in the MailTip to control when a client might display the MailTip. For example, you can't create a MailTip that scans the message text for a matching pattern such as a Social Security number or even a simple term such as the

name of a critical project. In some ways, Exchange 2010 started the idea in terms of offering proactive advice to users to help them use email more efficiently. Exchange 2013 adds Data Loss Prevention policies, which use much the same technique to offer advice to users who include sensitive information in messages.

MailTips are stored as properties of the mailbox in Active Directory. The Mailbox server reads the data hourly, so any change you make to a custom MailTip can take up to an hour to become effective. The only way to force a change to become effective earlier is to recycle Internet Information Services (IIS), which is difficult in a production environment.

Multilingual custom MailTips

Exchange 2013 accommodates multilingual organizations by enabling you to create custom MailTips in all the languages you need to support. This is a little more complex than setting a simple MailTip in one language because first you must determine the list of languages, translate the string into appropriate text for each language, and then populate the array of translations by using the MailTipTranslations parameter for each of the Set-Mailbox, Set-DistributionGroup, Set-Contact, and Set-MailPublicFolder cmdlets as required. You have to create a default MailTip before or when you add the translated values. The default value is used whenever the language specified for a user mailbox does not match a specific value in the list. In this example, translated values were set for four languages in addition to the English default. Each of the languages is separated by a comma.

```
Set-Mailbox –Identity 'Jacky Chen' –MailTip 'Financial Services Manager'
–MailTipTranslations 'NL: Manager van de financiële Diensten', 'IT: Responsabile
di servizi finanziari', 'FR: Directeur de services financiers', 'ES: Encargado de
los servicios financieros'
```

The translated values are stored in the MailTipTranslations property as an array of HTML values. To see the values, type:

```
Get-Mailbox –Identity 'Jacky Chen' | Select MailTip* | Format-List
```

OAB support for MailTips

The MailTips functionality would be only a partial solution if it only worked when users were connected online. Microsoft therefore upgraded the OAB structure to include a new set of MailTips properties for recipients that enables Outlook to process MailTips offline. The properties are as follows:

- Message delivery restrictions
- Custom MailTips

- Maximum receive size
- In addition, some new properties are kept for groups:
  - Moderation enabled
  - Total member count
  - External member count

All these data, with the exception of the membership counts, are extracted from Active Directory. Membership counts come from group metrics.

The display or Details Templates Editor

Outlook uses detail (otherwise called display) templates to format and display directory information when users view details about objects in the GAL. You can customize the templates to add or remove fields or change the text descriptions (labels) of the fields. Customizing display templates is a relatively common occurrence in large organizations but less common in smaller organizations, which typically stay with the default versions shipped with Exchange.

Figure 7-22 illustrates a simple customization of a display template. The rightmost screen is the standard Organization property page, and the left side is the customized version. Outlook 2013 is used in both cases, but the same techniques apply to Outlook 2007 and Outlook 2010. Two new attributes about the mailbox are visible in the customized template:

- **Contoso Accounting Code** Most large organizations assign employees to cost centers or similar accounting structures. Users might want to access this information to know what they should enter in documents such as expense reports. You can adapt one of the standard Exchange properties for this purpose by renaming it, but it is usually better to use one of the 15 extended properties Exchange provides for customizations like this.
- **Employee Type** Again, depending on the culture of the organization, you might want to display information about a user's employment status (permanent, part-time, contractor, or, in this case, executive) to other users.

You can modify display templates through the Exchange toolbox. Click Details Templates from the list of utilities in the toolbox and then select Open Tool from the action pane. The tool is a new instance of Microsoft Management Console (MMC) that reads the information about available details templates from Active Directory and loads them into the console as shown in Figure 7-23. You can use the tool to modify templates in any of the

languages Exchange supports. If you operate in a multilingual organization, you should be consistent and make the same changes to the details templates for each language version of the Outlook client you deploy. For example, if you deploy the Swedish, Finnish, German, and English versions of Outlook, you should update the details templates for these four languages.

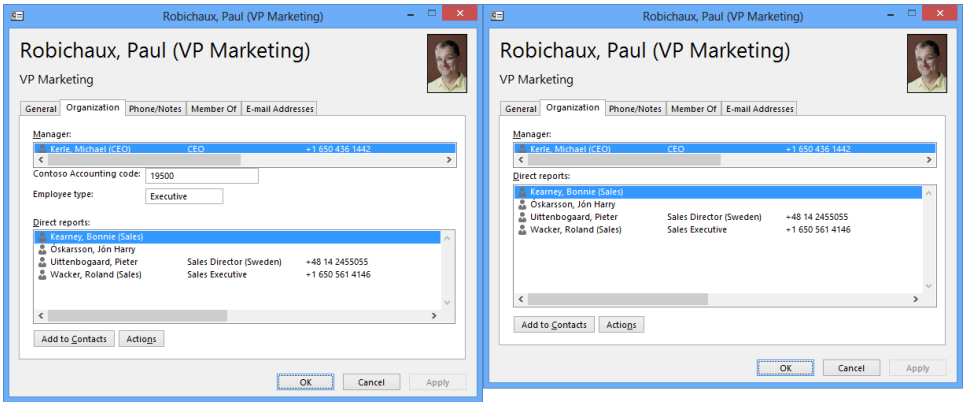


Figure 7-22 The effect of template customization

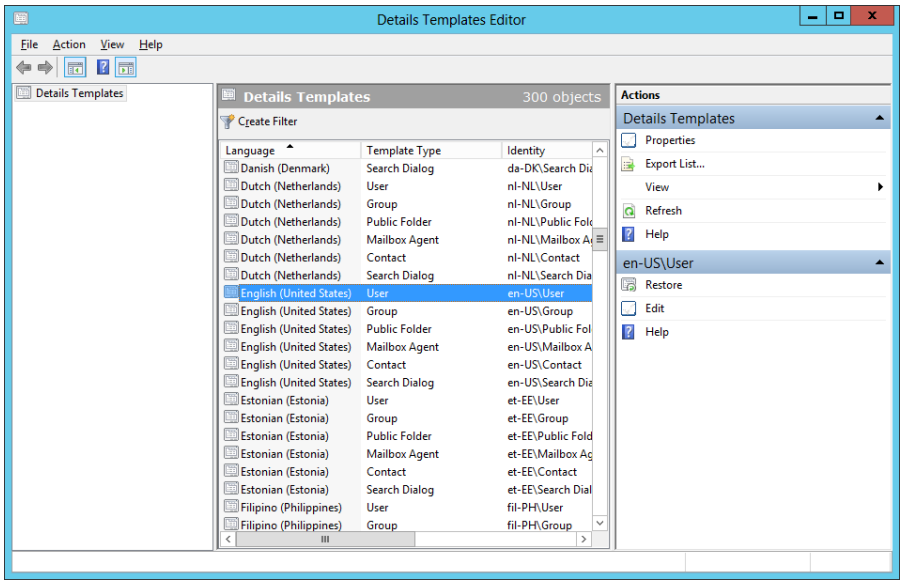


Figure 7-23 Viewing the installed templates using the template editor

The user template is the most commonly customized template. Because it was an example in Figure 7-22, I will explain how to apply the customizations described before. To work with a template, find it in the list and double-click it. Exchange fetches the details of the current template from Active Directory. The template in which you are interested is the en-US\User template. This is the English language—U.S. variant—of the user details template. Despite Exchange being used in English in many other places, such as the United Kingdom, Ireland, and Australia, that often do not share the same spelling or usage of words, you have only one English-language details template with which to work. The same is true of one Spanish template, one Portuguese template, and so on. The rule is therefore that whatever changes you make have to apply to the broadest possible user population that accesses Exchange in whatever language you choose.

Note

You can also see that the console includes a Restore option. This is your fallback in case you make an absolute mess of editing a template; the option restores a template to its original version as shipped in the Exchange kit. If you select this option, you will lose all the work you did on a template, but it's sometimes the best (or only) choice.

The template editor provided with Exchange 2013 is much the same as that used with Exchange 2007 and Exchange 2010. It is still basic by comparison to most layout editors, but it gets the job done. Figure 7-24 shows the Organization tab of the user details template loaded, and you can see that the added accounting code field is selected.

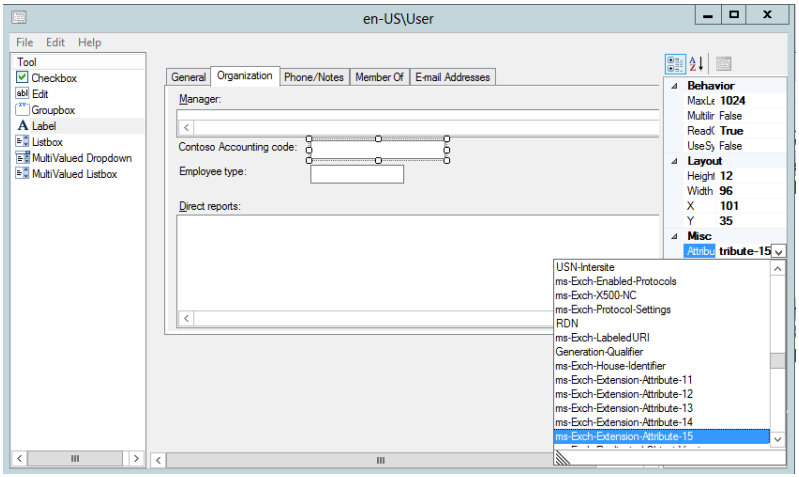


Figure 7-24 Customizing a details template

The toolbox on the left enables you to select different types of controls for the various properties you want to display. To add a new field, select the type you want, drag it into position on the page, and then set its properties, such as the attribute you want to display. In this case, Contoso Accounting Code is displayed in a simple edit box that is linked to the ms-Exch-Extension-Attribute15 mailbox attribute (otherwise known as CustomAttribute15). You can click a drop-down list of available attributes for the mailbox object.

The control types you can choose from are the following:

- **Checkbox** Used for on–off or yes–no fields such as flags indicating that an object has a particular status.
- **Edit** Displays only one value for an attribute. For example, users have only one first name, which comes from the Given-Name attribute. In the example, the edit box is the best option because the accounting codes and employee type are single-value attributes.
- **Listbox** Displays data for a list of attributes that come from different objects. For example, a user can be a member of many distribution groups, so the Member Of page has a list box that is populated from the Is-Member-Of-DL attribute.
- **Multivalued Listbox** Displays data for an attribute that can have multiple values. For example, user objects can have multiple email addresses, so the Email Addresses page has a single multivalued list box that is populated from the Proxy-Addresses attribute.
- **Label** Holds a language-dependent text string and typically appears alongside another control that holds data. Some characters in the label fields are underlined to indicate that users can navigate to these fields by using the Alt+letter key combination. For example, the S in the Supervisor field is underlined, so you can move to it with an Alt+S combination. You include the ampersand (&) character in the label control value to set this up, so the value in the label field in this case is &Supervisor.
- **Groupbox** Specifies a group of fields around which Outlook draws a line when it displays the template. For example, the General page of the user template has a line around the First, Last, Display, and Alias fields.

After you've saved the customized template, Exchange makes it available immediately to any Outlook client that connects in online mode. Outlook clients that connect in cached Exchange mode won't be able to see the customized template until the next time they connect and download the OAB.

INSIDE OUT **Points to consider**

Customizing details templates is easy, but you have to consider whether a customization adds any value before you rush to change something; you also need to consider some minor points. Remember that you will have to revalidate the customization after each upgrade of Exchange because there is no guarantee that Microsoft will not overwrite your customization or make a change to the templates that is not compatible with your changes in an update. As it happens, the transition from Exchange 2003 to Exchange 2007 and then to Exchange 2010 and Exchange 2013 was very smooth for template customizations, but this might not be the case in the future. If you operate multiple organizations, you will not be able to share template customizations directly between the organizations. Instead, you will have to export them using a tool such as LDAP Data Interchange Format (LDIFDE) from the organization where you make (and test) the customizations and import them into the Active Directory instances that support the other organizations.

The next step

At this point, the book has covered many of the elements that surround mailboxes. There's no place to head but to the core of Exchange—the Information Store—to discuss just what happens to messages and other items inside mailboxes.

CHAPTER 8

The Exchange 2013 Store

Long live JET! 388
Maximum database size 389
Dealing with I/O 392
Workers, controller, and memory 402
Managed Availability and the Managed Store. 407
Database management 407
Transaction logs 419
Background maintenance 432
Corrupt item detection and isolation 437
Rebuilding a database 448
Database usage statistics. 451
Mailbox assistants 454
And now for something completely different. 455

Maintaining and evolving a database isn't an easy engineering challenge, especially when the database has to deal with transactions that vary widely, from a simple 10 KB message sent to a single recipient to a message sent to a large distribution (with each address individually expanded in the header) accompanied by a 10 MB Microsoft PowerPoint attachment. Today's competitive environment and the demands of the service create the need to support very large mailboxes and to answer the continuing demands of the corporate sector for improvements in areas such as high availability. Microsoft might well have invested more engineering in the Store even if Google hadn't come along, but there is no doubt that competition can have a stunning effect on innovation.

The overall goals Microsoft engineering took on as it rewrote the Store into managed C# code (the reason it's called the Managed Store) in Exchange Server 2013 were the following:

- Continue to reduce I/O to take advantage of denser, cheaper disks and increase the flexibility available to storage architects when laying out storage designs for Exchange.
- Make 25 GB mailboxes work as well in Exchange 2013 as 1 GB mailboxes work in Exchange 2007.
- Improve the operational effectiveness of Database Availability Groups (DAGs) in areas such as site transition, use of lagged database copies, and automation.

Whenever you think about the Exchange Store, three major components come to mind:

1. The Extensible Storage Engine (ESE), the database engine that organizes the contents of mailbox databases at page level. Strictly speaking, ESE is not part of the Store. Instead, it is the underlying database engine on which the Store is built. However, ESE and the Store go together like twins, or maybe the odd couple.

2. The Information Store, which lays down a logical schema that defines the contents of mailboxes and how those contents are organized.

3. The Exchange database (EDB) and transaction logs. The EDB is the database file on disk. The transaction logs record any change made to a page within a database (adds, modifies, deletes) and provide the ability for Exchange to recover from a database failure by replaying transactions recorded in the logs. The transaction logs are also used to replicate data between servers and keep database copies synchronized so that copies can be quickly switched in to provide continuous service to users if the primary (active) database fails.

This chapter reviews the components of the Exchange 2013 Store and the major changes made in this version.

Long live JET!

Exchange 2013 continues to use ESE, which has been highly optimized over the years to meet ongoing product requirements for Exchange. Some are surprised that Microsoft persists with ESE as the platform for Exchange when it seems to have another perfectly good enterprise-class, high-performance database engine in SQL Server. On the surface, it's an attractive notion to unify all database development activities into a common platform that Microsoft and third-party developers can support. Money would be saved on engineering, testing, and support activities, and everyone could use a single target platform through a unified set of application programming interfaces (APIs). Microsoft has been down the path to investigate the effort required to move Exchange to SQL on a number of occasions. However, apart from the sheer engineering effort, a number of technical and nontechnical challenges would have to be overcome before Exchange could use SQL.

The demands on the two database engines are very different: ESE is optimized to process messaging transactions that vary from a one-line message sent between two people to a message sent to a very large distribution complete with a multi-gigabyte attachment (your favorite communication from the marketing department). SQL is optimized to handle structured transactions that do not tend to be highly variable. Could performance and scalability be maintained in a single database that attempted to serve both demands?

ESE enables users to create their own indexes (views) on the fly with a simple click of a column heading in Microsoft Outlook. The result is a colossal number of temporary views that differ from database to database that ESE manages in an elegant manner. SQL doesn't tend to be so accommodating. Again, SQL could provide the functionality, but would it take away from database performance?

SQL is designed to be a platform for applications. Exchange has had many attempts to become a platform for different types of applications, such as workflow routing, but is now simply a messaging server. Microsoft SharePoint (which uses SQL) has supplanted Exchange as the cornerstone of the Microsoft collaboration applications strategy. If Exchange used SQL, would Microsoft still be able to sell SharePoint as successfully as it does today?

Both SQL and Exchange have developed third-party ecosystems around their platforms that add enormous value for anyone who wants to deploy these products. Given that most partners who support Exchange today use Exchange Web Services (EWS) rather than ESE to access the Store, it shouldn't be a problem to move them to SQL because EWS could hide the changeover. In any case, who really cares what database engine powers an application as long as it does the job and delivers robust and scalable performance in a reliable manner? Do people ask what database Google uses for Gmail? The choice of the underlying database technology really shouldn't be an issue as long as email continues to flow.

Maximum database size

Microsoft has recommended a maximum size for an Exchange mailbox database of 2 TB. The current database page size of 32 KB makes the maximum theoretical size of a database at the NTFS file system limit (16 TB–64 KB). To be safe, the absolute limit for an Exchange 2010 or Exchange 2013 database is 16,000 GB, just below the NTFS file system theoretical limit. Few customers are likely to operate databases that approach anywhere near this value in the next decade, especially when they consider just how long it would take to restore such a monster database if that need were to arise.

Allowing for a 2 TB maximum database size seems startling on the surface, but it can be justified by three major technical and operational factors:

- Exchange 2013 does not support streaming backups, and you are forced to use Volume Shadow Copy Services (VSS) disk-based backups instead. VSS backups are faster and more capable of dealing with huge volumes of data. Taking a snapshot is a lot faster, and if you need to create a copy on tape to satisfy off-site archival procedures, you can do this after the VSS backup is available. Note that changes in the way Exchange supports VSS writers means that third-party backup software vendors need to produce new versions of their products to support Exchange 2013. Alternatively, on smaller systems, you can use Windows Server Backup as described in *http://technet.microsoft.com/en-us/library/dd876854.aspx*.

Tip

Note that you do not need to use VSS to restore a database. You can certainly recover a database from a snapshot or cloned copy, but it is much faster to use continuous replication and replace a failed active copy with a passive copy in a DAG.

- Exchange 2013 supports a replication model that allows up to 16 copies of a database. If you create three or more copies of a database, there's a fair argument to make that backup operations become less important than they were in the past because you can recover faster and more simply from a passive copy than by reaching for the most current backup. This fact does not eliminate the need for backups because legal and other requirements might mandate backup copies for off-site storage. See Chapter 9, "The Database Availability Group," for a discussion about the Database Availability Group (DAG) and how Exchange 2013 handles multicopy database replication.
- Background maintenance performed by the Managed Folder Assistant and other components means that Exchange Mailbox servers are quite capable of keeping very large databases in good health.

A large database really doesn't take longer to mount and make available to users; the primary consideration in terms of mount time is the number of transaction logs that might need to be replayed to bring the database up to date during the mount operation. Larger databases might have more transaction logs to replay, but the net impact probably won't be noticed.

INSIDE OUT The assumptions behind capacity planning

Considerations such as availability, recovery, and business requirements must be factored in to server planning. Tools available from Microsoft such as the Exchange 2013 Server Role requirements calculator (available for download from TechNet) help determine the basic technical design for Exchange 2013, but be prepared to hand-tailor the resulting output as you incorporate other needs that cannot be easily input into a spreadsheet before you arrive at the final design. All sizing tools make assumptions such as the default client type, user profile, and concurrency ratio, and the recommendations they make will be most satisfactory when your situation closely matches these assumptions. In other circumstances, your users might be busier than the assumptions, they might not use Outlook and Outlook Web App, or they might not connect as often as you'd expect, and therefore the results generated by a sizing tool will need to be adjusted before they make sense for you.

Sizing mailboxes

Calculating database size by multiplying the number of users by the mailbox quotas is still a valid way to approach the problem of what size database you can support, and it's certainly one that works in small to medium deployments. However, email systems have become more complex in the past few years, and other factors must be taken into account. Microsoft has posted an excellent discussion of the factors that drive Exchange 2013 hardware planning on *http://blogs.technet.com/b/exchange/archive/2013/05/06/ask-the-perf-guy-sizing-exchange-2013-deployments.aspx*. The concepts explained there can be used with some or all of the following questions to help frame the discussion.

- Are any legal or regulatory conditions in place that force retention of mailbox data for a specified period?
- What is the company's backup regime? Will you use DAGs to achieve higher availability and protection with Exchange 2013?
- How are users coping with current mailbox quotas? Are most within limits, or do you see a constant stream of requests for additional space? Are you happy to have users spend time juggling mailbox contents to keep under their quota?
- What deleted-items retention policy do you operate? Is it sufficient or should it be extended? How often do you have to restore from backup to recover user data?
- Do you force automatic deletion of items in mailboxes? If not, should you be using a retention policy?
- Do you use an archiving product with Exchange? Will Exchange archive mailboxes replace or complement this product? Will archive mailboxes be collocated in the same database as the primary mailboxes or in another database, or even grouped on a separate, dedicated archive server?
- Do you need to journal messages? How often do you have to perform legal discovery searches or otherwise react to complaints when email forms essential evidence that has to be found and retained?
- Has the messaging profile of your users changed in the past few years? Are they sending and receiving more email? What is the average size of the messages? Are they using calendars and other mail-enabled applications more?

These questions address a mixture of technical and business requirements and lead to an understanding of how users generate and use mailbox data. There's no simple formula to turn the responses into an answer for the mailbox quota you should provide and how many databases of a particular size will be required. Instead, you're more likely to realize

that you have to deliver differentiated services to various categories of users. Executives and other high-impact staff will have different quotas, retention periods, and archiving and legal requirements than factory workers, and their mailboxes will probably be in databases located on the most highly available servers. These servers might use different hardware configurations, be operated by dedicated staff, and have to support access through a variety of devices (mobile, Outlook, and Outlook Web App) rather than the more limited variety that might be allowed to less-important users.

INSIDE OUT **Larger mailbox quotas required!**

One of the changes made in Exchange 2013 is in the way the Store calculates the space used for mailboxes that is then charged against a user's mailbox quota. All databases incur some level of overhead to manage internal structures. In the past, Exchange did not charge this overhead against mailbox quotas, but Exchange 2013 does. The net effect is that database sizes stay the same because the same level of overhead exists, but mailbox quotas might have to be increased because the overhead has been charged. In effect, the reported size of a user mailbox will appear to grow by some 30 percent after it is moved to Exchange 2013, despite nothing additional being held in the mailbox. At least, not in the eyes of the user. A proactive step is therefore to review the quotas assigned to user mailboxes before they are moved to Exchange 2013 to ensure that sufficient quota remains to enable Exchange 2013 to charge the additional overhead to the mailbox. Some organizations have simply increased all quotas by 50 percent to ensure that they won't encounter difficulties during migration, with the intention of reviewing quotas when everyone is on Exchange 2013. This seems to be a very practical and pragmatic approach.

Dealing with I/O

Even though the type and volume of email had changed dramatically since Microsoft first laid out the Exchange database schema in 1996, Exchange Server 2000, Exchange 2003, and Exchange 2007 use the same schema that emphasizes efficient storage over utility. Cheaper storage makes it less attractive to focus on storage efficiency. To achieve a reduction in I/O, the Store has steadily moved from forcing disks to do many small, random I/Os to fetch data to using larger, sequential I/Os. The physical performance difference between random and sequential I/O almost guarantees better performance and lower I/O activity for any application if the code is written to move away from random I/O. The current database schema emphasizes contiguity over storage, essentially by keeping mailbox content together. Because more data is contiguous, the Store can read data out in large sequential chunks rather than in many random and smaller chunks.

Exchange has not supported single-instance storage since Exchange 2010. You might assume that databases will grow larger because mailboxes hold their own copies of messages. In fact, this isn't always true because of the way databases compress message content. However, a side effect of the elimination of single-instance storage is that more transaction logs are generated to capture the insertion of individual copies of items in the table for each mailbox. For example, if you send a message with a 1-MB attachment to 10 mailboxes in the same database, the Store has to create at least 10 transaction logs.

Views (or secondary indexes) are a very valuable user feature. Outlook allows users to sort items within folders by using a wide variety of properties (author, date received, subject, and so on). Every time a user opts for a different sort order, the Store creates a new view. Thereafter, as new items arrive in the Store, a lot of work occurs to update the views. Eventually, if a view is not used, it expires, and the Store removes it.

Large mailboxes tend to have many views, and those views are usually for the default folders (Inbox, Sent Items, and Deleted Items). Many people don't like the work involved in filing email in different folders and are happy to let messages build up in the default folders, and they only ever take action when prompted to by quota exhaustion. Search technology facilitates this behavior because no penalty exists if you never file a message; search will always find an item faster than you can, even with a solid filing system. The result of declining discipline in filing is folders that hold many thousands of items, in turn causing the Store to generate additional I/O to access large folders. This prompted Microsoft to recommend that the largest folder should hold fewer than 5,000 items in Exchange 2007. The improvements made since enable Exchange 2013 to tolerate up to one million items in a single folder (assuming that the view for the folder is a standard view that's optimized within the schema and does not have to be rebuilt). However, this is very much a notional limit because even the most modern clients will run into performance difficulties when they attempt to deal with such a large folder. It's still best to coach users to keep fewer than 10,000 items in a folder whenever possible—or to deploy retention policies to help users keep their mailboxes under some degree of automated control.

How Exchange 2013 supports large mailboxes

Exchange 2013 is designed for a world in which mailboxes of 25 GB and above are often encountered in deployments. Here are some of the ways the Exchange 2013 database schema and layout facilitate large mailboxes.

- **Use an optimized Store schema designed to deal with large mailboxes more efficiently.**
- **Force pages within the Store to be laid out contiguously rather than having pages scattered around the database.**

- **Use large, 32-KB pages that can store complete messages and ensure that the Store allocates pages in large, contiguous chunks rather than randomly within the database.**
- **Use large I/Os to fetch sequential chunks of data. The logic here is very simple. Today's disks are capable of providing data at 300 or more sequential I/O operations per second (IOPS), whereas their ability to handle random I/O remains constrained at around 50 IOPS. Given that disks can provide sequential data six times faster than random data, it makes perfect sense for Microsoft to focus its engineering investment on moving activity within the Store from random to sequential. Apart from the performance benefit, this step enables a wider range of storage designs to support Exchange.**
- **Use a large, 100-MB cache size for mailbox databases in a DAG. This step enables the Store to keep more dirty pages in memory. A difference between Exchange 2010 and Exchange 2013 is that passive database copies move from a 5-MB to a 100-MB cache, which reduces the amount of I/O consumed to keep a database copy updated. This cache size (basically memory assigned to hold data that have been recorded in transaction logs but not yet committed to the database) is referred to as the checkpoint depth. Increasing the cache improves the chance that Exchange can update a page in memory rather than having to generate an I/O to retrieve it from disk. Increasing the cache size introduces some additional complexity that the Store takes care of, including the potential for longer shutdown times because the cache needs to be flushed to disk and longer recovery times from crashes because more transactions have to be replayed from transaction logs. However, the overall benefit in I/O improvement is worth taking these risks.**
- **Use compression for some object types within the Store.**

The work to enhance the Store over the last two releases has had a huge impact on I/O performance. Inside a reasonably small database, it's probably acceptable to assign pages to store new items in whatever pages are free within the database. For a 20-KB message, this meant that the item might be placed in several pages scattered in different locations within the database. From a logical perspective, the database can find the five pages and provide their locations to the Store whenever a client requests the item, but this activity generates a lot of very small I/Os. A 32-KB page size makes it more likely that a single item will fit on one page and ensures that pages required to hold new items are assigned contiguously in large chunks.

When clients need to access an item, the Store uses a technique called gap coalescing to read in a range of pages in a single large I/O rather than hunting around the database for individual pages. The Store also writes pages in contiguous chunks to smooth I/O. Consider when four pages in a contiguous set of six pages are updated in the cache and need to

be written to disk. Exchange 2007 writes each of the four dirty pages in individual I/Os. Exchange 2013 writes the six (dirty and clean) pages in one large I/O, and ESE discards the unchanged pages as it updates the database. Using gap coalescing reduces the overall I/O demand Exchange exerts and increases overall data throughput.

The change that updates views on demand rather than after every change is also important in terms of I/O reduction. Microsoft refers to this as a change from a nickel-and-dime approach, in which the Store is in a state of constant turmoil as views are updated after the arrival of new items, to a pay-to-play approach, in which the Store updates a view only when a client wants to use it. Consider when a user creates a view through Outlook Web App (Outlook clients that work in online mode also create views in this manner; when Outlook works in cached Exchange mode, it creates views on the client) to look at items in her inbox sorted by author (sender) rather than the default sort order (received date/time). Such a view is usually created so a user can look for items sent by a specific individual, when a view sorted by author is much more useful than the default view sorted by the time an item is received. However, assume that the user then switches back to the default view and doesn't look at a view by author for another week. Even though the client does not intend to access the view, the Store keeps updating it just in case. This might be good in that a view is immediately available, but it incurs a horrible overhead for the Store to keep track of all the views and update views as new items arrive. The resulting updates generate I/O activity because the changed views have to be written back into the database. Eliminating these updates and using an on-demand model is a very intelligent change, especially because the multicore servers used today are very capable of handling the small extra demand for the processor to compute a view when required.

In its efforts to improve I/O performance, Microsoft also paid attention to the concept of database write smoothing. Every disk has its performance limitations that restrict the amount of data it can handle at any time. If you attempt to write too much data to disk at one time, the disk will be swamped, and a disk contention condition will occur. The application cannot write its data as quickly as it wants to, and everything halts until the disk can accept more data and relieve the contention condition. Write smoothing means that the application is intelligent enough to detect when disks are approaching the limits of their performance and can throttle back its demand to allow the disks to continue to process data at a consistent rate.

Exchange uses the checkpoint depth as its measure to determine when it should throttle database writes. This approach is advantageous because it works for all disks. When the checkpoint depth is between 1.0 and 1.24 of the checkpoint target, the Store limits the outstanding write operations for each logical unit number (LUN) to one until the condition improves. If the checkpoint depth goes past 1.25 of the target, further throttling is imposed by increasing the maximum number of outstanding write operations per LUN until a balance is attained and the depth begins to reduce. Pending writes remain in cache until they can be written to disk.

Microsoft claims that it has driven a reduction of over 90 percent in I/O operations since Exchange 2003. The actual gain must be measured in your own specific hardware environment and might be better or worse, depending on server and storage configurations, user workloads, and types of clients. However, it's fair to say that the focus on reducing I/O has worked and has greatly improved the overall performance characteristics of the product.

INSIDE OUT When are large mailboxes justified?

Many older Exchange deployments started with mailbox quotas of 50 MB to 100 MB and have gradually increased over time to a range of 250 MB to 10 GB, with some select mailboxes being allocated a higher quota. Usually, these mailboxes are owned by executives or other users who have a business need for higher email volumes or for holding data for longer periods. The other users resort to PSTs to hold messages they want to keep so that they stay within their allocated quota.

Although the practical limit for an Exchange 2013 mailbox is around 100 GB, in reality, this usually translates into an average corporate mailbox size of 5 GB to 10 GB. A mailbox quota of 5 GB is sufficient to store approximately 100,000 items with a 50 KB average size. For most people, it will take five years to accumulate that much mail, and it is possible that the company's legal team would prefer users not to store quite so much mail!

Before you rush out to enable 25-GB mailbox quotas, it's wise to reflect on the business requirements you might meet with large mailboxes. These reasons might be valid for your company:

- **Reduction in unproductive time spent by users who struggle to stay within lower quotas. How much does it cost if a message containing some time-critical information cannot be delivered to a mailbox because its quota is exhausted?**
- **Elimination of PSTs through the deployment of archive mailboxes.**
- **Elimination of expensive deleted-item recovery from backups through the deployment of an extended deleted-items retention period (more than 60 days).**
- **Ability to conduct online searches through all email available to the company rather than the subset that is available in online mailboxes.**
- **Protection of all email rather than just the items in the Exchange database.**
- **Provision of all mailbox content to multiple clients. Outlook Web App and mobile devices cannot access PSTs, and mobile devices cannot access archive mailboxes.**

Of course, keeping vast quantities of email has its downside. Search facilities are excellent today, but reviewing a large set of messages retrieved by a search to find a particular item is never a thrilling experience. From the system perspective, increasing user mailbox quotas means larger databases, bigger backups, and more data to manage, including the very real concern some companies have that retaining so much data makes the company more vulnerable to discovery actions. Like everything else in life, a middle path should be walked between keeping everything and keeping nothing. When you have a set of reasons to justify the deployment of larger quotas, you can begin to figure out how to deploy the servers and storage necessary to support large mailboxes.

Maintaining contiguity

Enormous care is taken to allocate contiguous space within Store databases. However, this is not the end of the story because pages have a lifetime that begins with their creation and extends through modifications and deletes. Given the need to support multi-terabyte databases, the old way of maintaining page contiguity through scheduled background maintenance could not work.

As the Store processes transactions, it checks to ensure that maximum contiguity is maintained and that it is using available space efficiently. If necessary, a background thread is created to address any problems—perhaps it will shuffle some pages to make them more contiguous or to free up space in the database. All this work is throttled automatically so that background operations never interfere with the ability of the Store to service user requests, just like Windows Desktop Search only indexes items on a PC's hard disk when the PC is inactive.

The result of these operations is that Exchange 2013 databases start out and remain more contiguous than their predecessors. Of course, some extra CPU resources are consumed to perform run-time processing, but this should not be a problem in most situations because the normal bottleneck for Exchange has been I/O rather than CPU for the past decade.

As you read the discussion about contiguity and about the 32-KB page size Exchange now uses, the thought might run through your mind that large pages arrayed in bigger chunks might increase the overall on-disk size of a database. This conclusion is correct, but it is offset by the introduction of data compression within the database.

Microsoft believes that the potential extra growth in database size is fully mitigated by the way message headers and HTML and text body parts are compressed. Of course, the exact ratio of compression varies, depending on the mix of content within a database (Rich Text Format [RTF], text, HTML, different types of attachments). For example, RTF messages are

already compressed and therefore are not compressed again when they are written into a database. Outlook 2010 and Outlook 2013 generate HTML-format messages by default so databases that support these clients get more value from compression than databases that support earlier clients such as Outlook 2003, in which the default message format is likely to be RTF. Exchange achieves additional efficiency by first clustering records and tags into contiguous chunks within a page before attempting to compress the data.

INSIDE OUT **Data compression doesn't include attachments**

Although great value is attained by compressing message bodies, Exchange doesn't attempt to compress attachments. Tests by the Exchange development team demonstrate that most attachment types that circulate with messages such as newer versions of Word documents and Microsoft Excel workbooks don't compress well because they are already stored in a compressed format. The same is true of other attachment types you might not want to see circulated by email, including JPEG, PNG, MP3, and WMA. Some older formats such as Word 2003 do compress well, but because these versions are being rapidly replaced, their attachments will form a decreasing percentage of the overall attachment volume. The decision was therefore made to avoid incurring the CPU overhead of compression to achieve what might be a marginal decrease in overall storage requirement, especially at a time when storage costs are reducing rapidly. The same logic was used when the decision was made not to attempt to compress RTF message bodies.

The database schema

The schema sets out how data are physically and logically organized within a database. Apart from some minor tweaking, Microsoft used the same schema from Exchange 4.0 through Exchange 2007, but the schema used since Exchange 2010 made some major changes, the biggest of which is the way the contents of mailboxes and folders are organized. The previous schema laid out a mailbox database as follows:

- A mailbox table containing a pointer to every mailbox in the database.
- A folders table containing a pointer to every folder in every mailbox in the database.
- A message table containing an entry for every message in every folder in the database.
- An attachments table containing an entry for every attachment in the database.

- A message/folder table maintained for every folder that lists all the messages in a specific folder. For example, the message/folder table for your inbox contains pointers to every message in your inbox.

In essence, you have four big tables shared by every mailbox in the database and a set of message/folder tables for the different folders. This schema, shown in Figure 8-1, obviously works successfully, but ESE generates more I/O than necessary to navigate through the tables in response to user requests, especially as some of the tables become very large in databases that support thousands of mailboxes. For example, a database supporting 4,000 mailboxes, each of which has 150 folders containing 20,000 items, has a folders table with 600,000 rows and a messages table with 80,000,000 rows. The message/folder tables aren't so much of a problem because most users have noted the advice given over the years to restrict the number of items within a folder to less than 5,000.

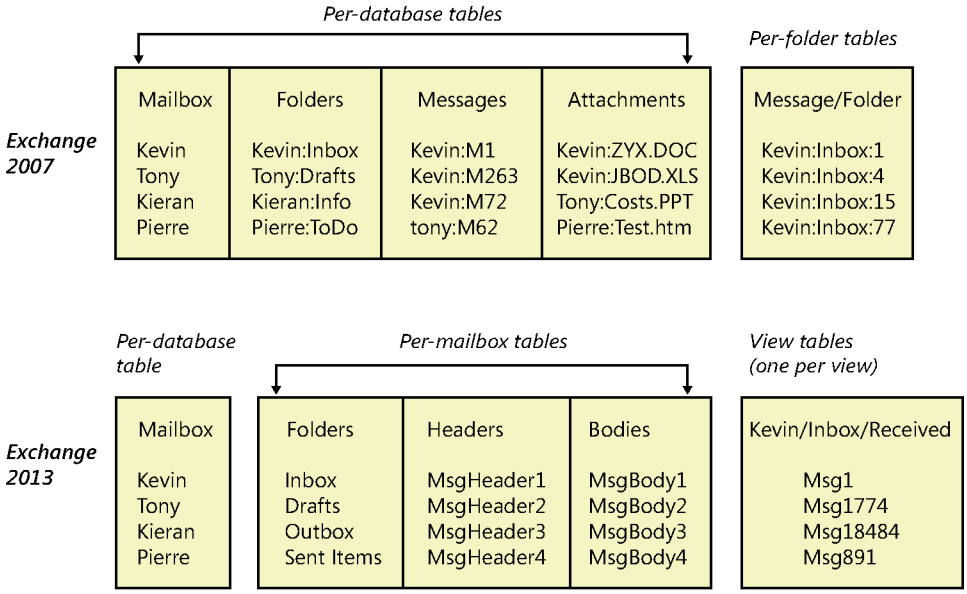


Figure 8-1 Difference between Exchange 2007 and current Exchange database schemas

As illustrated in Figure 8-1, the layout imposed by the current Exchange schema is as follows:

- A mailbox table containing a pointer to every mailbox in the database (as before).
- A folders table for each mailbox in the database. Although there are many more folders tables, the size of each is much smaller. Taking the example cited earlier, there are now 4,000 folders tables (one for each mailbox), each of which holds 150 rows.

- A message header table for each mailbox in the database. The message header table holds all the Messaging Application Programming Interface (MAPI) properties (subject, address list, priority, and so on) for the messages in a mailbox. Again, there are 4,000 message header tables, but the number of items in the largest mailbox sets the maximum size of a table, around 20,000 rows in this example.

- A message body table for each mailbox in the database. The message body table holds the message content (the body text), including attachments. Again, there are 4,000 message body tables for this example database, and the largest mailbox determines the number of items in the maximum size of the table.

- In addition, ESE maintains view tables on an as-required basis. A view is an ordering of a folder (for example, the Inbox ordered by date received). Previous versions of Exchange use secondary indexes to populate views. Exchange now holds views as tables. Although there can be many views (more than the number of folders because users can create multiple views per folder), the maximum size of a view table is determined by the number of items in the largest folder.

Two major types of tables are found within an Exchange mailbox database. Global tables hold information about all mailboxes in the database, whereas mailbox tables hold data specific to individual mailboxes. Global tables include the catalog; a registry of all tables that exist in the database; another table (Globals) that tracks information such as the database version; and the Mailbox table, containing basic information about the mailboxes in the database such as the owner, locale, and last logon time. Another global table tracks information about message delivery, and another is used by mailbox assistants to track events. The Mailbox table is a good example of one that is optimized for sequential I/O. Of the mailbox-specific tables, those that store folder information, messages, attachments, and the physical indexes are also optimized for sequential access.

In summary, the current database storage schema focuses on smaller tables that store data on a per-mailbox basis rather than on much larger per-database tables. This approach allows ESE to be more efficient as it navigates through a database and generates fewer I/O transactions for the storage system to process. It also enables Exchange to support up to 100,000 items per folder and still deliver reasonable responsiveness to clients that work online. The performance of cached mode clients depends on other factors such as the speed of the local hard disk and the version of the client. (Outlook 2013 or later is best because of the intelligent caching behavior incorporated into this client.)

Exchange 2013 I/O improvements

The overhaul of the database schema contributed enormously to the dramatic improvements in I/O reduction seen in the last two product releases. Exchange 2013 features a rewritten Store due to a transition to managed code. As part of the transition, the Exchange

developers were able to remove a large amount of redundant code that lingered from a time when the code might have been required in previous versions. At the same time, they were able to take advantage of the lessons learned from Exchange 2010 deployments, both by customers and at Microsoft, including Exchange Online running within Office 365.

The following changes contribute to better I/O performance in Exchange 2013:

- Within Exchange 2010 Database Availability Groups, passive database copies have a 5 MB checkpoint depth. This meant that the Store cached a relatively low amount of data for passive copies with the idea that this would enable faster activations of a passive database to become active. The low checkpoint depth and some aggressive pre-reading of data to maintain cached data at the 5 MB level meant that the I/O requirements for a passive database copy were the same as for an active database copy, despite the fact that all mailbox activity was focused on the active copy. Exchange 2013 increases the checkpoint depth for passive copies twenty-fold to 100 MB, which reduces the amount of processing required to maintain the depth. The result is that a passive copy now requires only 50 percent of the I/O of an active copy. Keeping more cached data for a passive copy might prevent fast failovers if the cache has to be flushed before activation (as is the case in Exchange 2010), so changes were made at the ESE level to persist the cache during activation and thus preserve the ability to move a database copy quickly from passive to active. In fact, with the changes, database transitions are reckoned to be somewhat faster in Exchange 2013 with the expected failover at around 10 seconds as opposed to 20 seconds for Exchange 2010.

- The Store performs ongoing background maintenance to keep databases healthy (see the "Background maintenance" section later in this chapter). In its early days, Exchange databases needed a lot of tender loving care because a mixture of bugs (server and client), storage glitches, and other issues caused frequent problems. Today, the more robust and resilient Store (including features such as single-page patching) means that a less aggressive approach can be taken to background maintenance, a feeling that was confirmed through instrumentation of databases running in the Exchange Online service. Exchange 2013 throttles back maintenance from 5 MB/second to 1 MB/second. Maintenance is still done, just a little more slowly than before. The upside is that less I/O is consumed for maintenance.

- Further database and schema tuning contribute to reduced I/Os too. For example, collection of MAPI properties that are frequently used in view operations are held in binary large objects (BLOBs) so that they can be fetched together in one chunk. Read operations have been examined and converted to sequential I/O whenever possible to eliminate random I/O further and make Exchange deployments feasible on even low-performing storage subsystems.

Chasing I/O improvements requires constant effort to track changes in the code (Exchange and Windows), disk technology (across all classes of storage), and to accommodate customer demands. The Exchange developers have done extremely well over the past decade to move from when storage was the Achilles heel of Exchange deployments, especially at scale when the necessary storage was expensive and often difficult to manage, to today's situation, when Exchange 2013 will run on just about any storage you care to deploy. It should be interesting to see what they'll do in the future.

Workers, controller, and memory

As part of the rewrite to managed code, Information Store processing is now separated into the work done by a Store controller process and a set of worker processes, one for each database (active or passive copies) mounted on a server. Each worker process deals with the transactions flowing into its database and does the work required to keep the database healthy, such as background maintenance. The major advantage gained by moving work into a set of worker processes is that any problem that occurs is automatically isolated to a single process and therefore can affect only the database to which the process is connected. Of course, if the controller process dies for some reason or is terminated by an administrator (for instance, to stop the Information Store service), all the worker processes also terminate. Figure 8-2 shows that four *Microsoft.Exchange.Store.Worker* processes are running on a server. You can therefore conclude that the server supports four mounted databases (which could be active or passive copies). You can also see the Store controller process, *Microsoft.Exchange.Store.Service*. You can discover which worker process controls a database by running the Get-MailboxDatabase cmdlet. The value of the WorkerProcessId property returned by Get-MailboxDatabase is the identifier, or PID, of the worker process.

The change from a single monolithic Store process to a coordinated set of worker processes also affects the way Exchange 2013 uses memory on Mailbox servers. The old Store process uses a mechanism called dynamic buffer allocation to seize as much memory as it can on a server and then throttles back its usage depending on the requirements of other applications and Windows. This is why Store.exe appears to occupy huge amounts of memory on previous Exchange servers.

As the Store becomes active, worker processes are initialized to manage each of the databases on a server. Some databases will be active and some passive. Exchange 2013 begins the process of memory allocation by building the ESE cache, using 25 percent of available system memory. This memory holds database structures in memory instead of having to go to disk to fetch pages because of client transactions. The cache is then divided by the number of mounted databases on the server to establish a maximum target memory for each database (Max Cache Target). If the Set-MailboxServer command is used to configure a *MaximumActiveDatabases* property to control the maximum number of databases that can be activated on the server, this value determines the dividing factor for the ESE cache.

Worker processes use this factor to determine how much of the cache they request ESE to allocate for their workload.

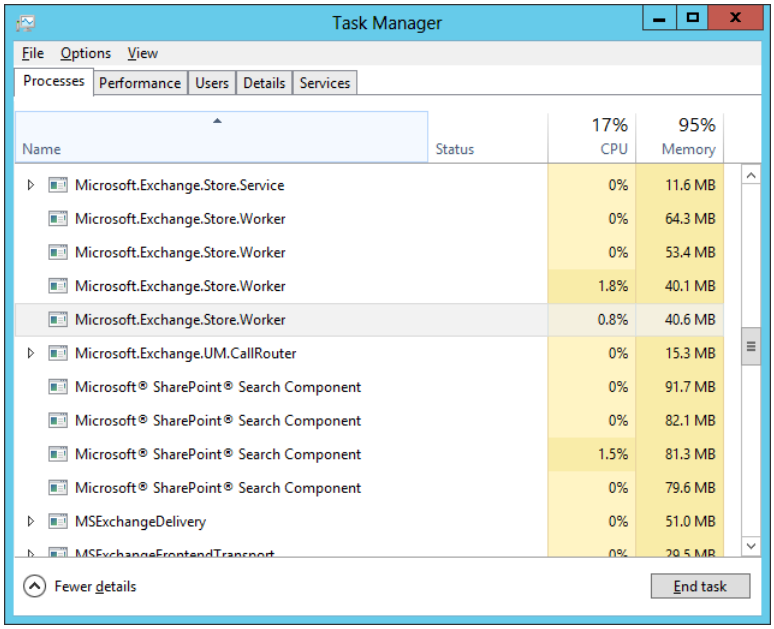


Figure 8-2 Store processes on an Exchange 2013 Mailbox server

The cache size is determined when the worker process starts and is not dynamically adjusted following the addition or reduction of mounted databases on the server, and setting the *MaximumActiveDatabases* property will not immediately affect the determined size for the ESE cache. If a need exists to calculate a new ESE cache value, such as adding or removing a database, you have to restart the Information Store service to force the worker process to recalculate the cache size.

A worker process can grow to the Max Cache Target limit that is in force when its database is mounted. The worker processes dealing with active databases are allocated the full amount, whereas those associated with passive databases get 20 percent of the maximum target. Of course, if a database copy is activated, the worker process increases the cache used to reflect the fact that the process is now handling active client connections.

For example, assume that you have a Mailbox server equipped with 32 GB of RAM. Following the rule, a quarter of the memory (8 GB) is computed as the ESE Max Cache Target. If four active databases are mounted on the server, their share of the ESE cache will be allowed to grow to a maximum of 2 GB. Such a server is likely to have many more databases mounted on it. If you were to add four passive copies, you would now have

eight mounted databases, so the maximum amount of memory each database can be allocated is now 1 GB. The worker processes for those copies will each start with 200 MB (20 percent of 1 GB) to enable them to process replicated transactions from the active copies. Even after increasing the number of local database copies to eight, you are still using only 4.8 GB of the 8 GB that Exchange assigns to the ESE cache to run four active and four passive copies. With this amount of free cache, the server is easily able to accommodate a scenario in which all eight of the mounted databases are activated. As you'll see in just a little while, you can make better use of the available memory by adjusting the value of the *MaximumActiveDatabases property for a server.*

Given that most Mailbox servers are now being configured with amounts of memory that would have been deemed scandalously over the top just a few years ago, a server equipped with 96 GB of memory is no longer unusual. On such a server, Exchange will reserve 24 GB for the ESE cache. If the server mounted 50 active databases, each would be allowed to have (24 GB/50) or approximately 490 MB of ESE cache.

A more likely scenario is that the mix of databases on a large server would be divided 50-50 active–passive. When 25 active copies and 25 passive copies are supported on a server, it's unlikely that more than a few of the passive copies will need to be activated at any one time. You can therefore use your knowledge about the operational environment to increase the amount of cache available to each active database copy by setting the value of MaximumActiveDatabases to restrict the number of databases Exchange can activate on a server. ESE now uses a different calculation to determine the Max Cache Target. It is:

Divider = (MaximumActiveDatabases × *0.8)* + *(number of databases* × *0.2)*

Max Cache Target Per Worker = (Total ESE cache allocated from server memory/Divider)

For instance, if you set the MaximumActiveDatabases to 30 for the server in question, the calculation is now:

Divider = (30 × 0.8) + (50 × 0.2) = 34

Max Cache Target Per Worker = (24 GB/34) = 723 MB

Overall, if the server mounts 25 active databases and 25 passive databases, the cache usage will be (25 × 723 MB) + (25 × (723 MB/5)) = 21,690 MB, or 21.18 GB. With this workload, the Store uses approximately 22 percent of the available 96 GB of system memory.

The increase in cache size from 490 MB to 723 MB obviously allows the worker process for each database to hold many more pages in memory and should therefore aid performance because the cache is more efficient if it can hold more data in memory. The additional cache can be assigned because Exchange does not allocate memory from the total cache to

passive copies that do not need to be activated, which is usual in a DAG in which multiple database copies are distributed across multiple servers.

A computer and its operating system will always be constrained in terms of available resources. On memory-scarce servers, it might not be possible to assign the amount of memory that the Store deems desirable for the worker processes. When insufficient system memory is available to allocate 25 percent to the ESE cache, the server moves pages for the least accessed memory to disk, and the worker processes are likely to follow suit and begin to page from disk. When a significant portion of the cache a worker process uses is on disk, ESE logs event 906 (Figure 8-3) to advise that it is having to use disk to store its buffers rather than maintaining them in memory. This is not an issue if the condition is temporary and caused by a peak in demand for memory from other processes or if you run Exchange on a server that is known to be underpowered, as in the case of a small, virtualized test server. However, it is a problem that will affect overall system performance that should be dealt with if event 906 appears an ongoing basis.

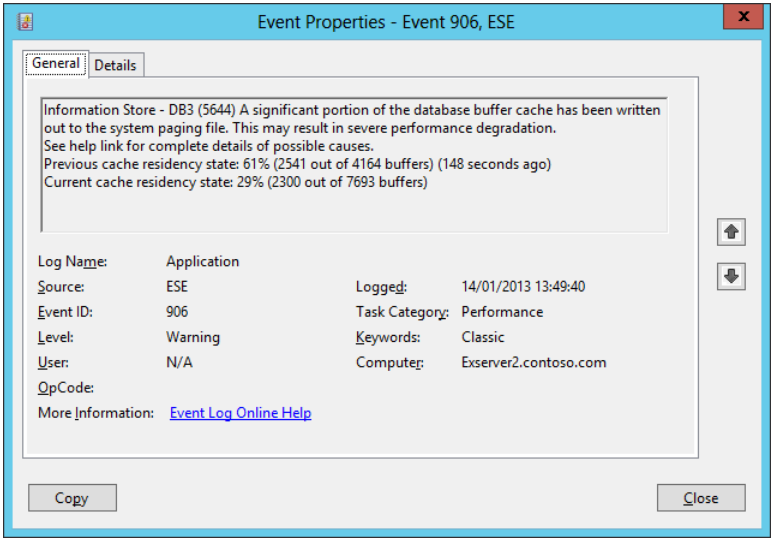


Figure 8-3 Event logged when memory is short on an Exchange 2013 server

On smaller Exchange servers that are relatively underequipped with memory yet expected to run Exchange and other applications, it is possible to constrain the total memory allocated to the ESE cache by using ADSIEdit to adjust the value of the *msExchESEParamCacheSizeMax* attribute for the Information Store object on the server so that the Store attempts to use less than 25 percent of available memory. This technique has been used to trim memory usage on Exchange servers since Exchange 2000 and is documented in *http://support.microsoft.com/?kbid=266768*. However, because Exchange 2013 uses 32-KB pages, you should use 32 KB as the divisor to calculate the number

of pages to assign to the cache and not 4 KB as specified in the article. For example, to constrain Exchange to use 1 GB for the ESE cache, the value is calculated as 1,024,000 KB/32 KB = 32,000. To complete the process, the same value should be set for the *msExchESEparamCacheSizeMin* attribute.

Over and above the ESE cache, the Store uses another set of memory to handle the general overhead to manage activity such as database transition (roughly 100 MB in the *Microsoft.Exchange.Store.Service* process) and to create and cache objects such as address book information, folder hierarchy, and the permissions used to execute business logic in the worker processes. The amount of memory used for this processing varies with the number of active and passive databases on a server and the number of client connections to each database. As a rule of thumb, the worker process for each active database will consume a private working set for its overhead of between 300 MB and 400 MB, whereas the worker process for a passive database will use about 80 MB. In total, this is more overhead than seen on an Exchange 2010 server, but the purpose of using memory is always to avoid going to disk for expensive I/O operations; this is another example of how Exchange continues to trade memory to reduce its I/O footprint.

Two other factors differ in the way Exchange 2013 Mailbox servers consume memory when compared to previous versions. First, the Search Foundation processes that index the content held in mailbox databases consume much more memory than does the equivalent MSSearch service on an Exchange 2010 or Exchange 2007 server. Four SharePoint Search component processes are active on Exchange 2013 Mailbox servers and are visible in Figure 8-2. Sometimes called *noderunner* processes after the executable name, the four processes perform the following functions:

- Content indexing
- Content processing
- Query handling
- Administration and orchestration

Overall, these processes can consume a large amount of memory on a Mailbox server, with between 10 percent and 12 percent of available memory deemed normal. The exact amount used depends on the number and content of the items stored in mailboxes, but a large Mailbox server hosting tens of millions of items across 20 mounted active databases might use a cumulative total of 10 GB of memory for content indexing.

The process that performs indexing (a separate index is maintained per mailbox database) usually consumes the most memory because it caches a substantial amount of information to be able to respond quickly to user queries. The actual amount of memory consumed depends on the number of items in mailboxes, the size (content) of the items, the

properties of the items, and whether the items have attachments that can be indexed. Broadly speaking, after a certain fixed cost in memory is absorbed to start up the search processes, the amount of memory the content indexing process consumes increases in line with the number of items.

INSIDE OUT **How many databases are supported by Exchange 2013?**

The Standard edition of Exchange 2013 supports up to five mounted databases, but the Enterprise edition supports up to 100 mounted databases. The important word here is "mounted," because any version of Exchange 2013 will allow you to create a maximum of 235 databases on a server. However, you will only be able to mount the number of databases permitted by the server license. Test or unlicensed servers have the same five-database restriction as the Standard edition of Exchange 2013.

Managed Availability and the Managed Store

All access to a mailbox is gained by the protocol stack running on a Mailbox server. If a protocol such as Outlook Web App is down, all active databases on a server cannot be accessed through that protocol. The introduction of the Managed Availability system to Exchange 2013 is of huge importance to the Store because it provides a framework to measure the health of components and then takes action to restore service for problematic components.

The Managed Availability worker process (*MSExchangeHWWorker*) is responsible for three major functions that are designed to detect and address protocol failure. Probes operate at regular intervals to assess the health of components by reference to performance counters and other system symptoms. Managed Availability uses health mailboxes in databases to measure the health of databases and the transport system. Monitors observe data from probes, which decide whether the components are operating normally. If a monitor observes an abnormal condition, a responder is activated in an attempt to bring the observed component back to health. Actions can be escalated if attempts to restore health are unsuccessful, up to and including a forced system restart.

Database management

Database management covers many topics. This section starts with a discussion about the basic files that constitute an Exchange mailbox database (see Figure 8-4) and then goes on to discuss some of the finer details about how transaction logs and the other files are used.

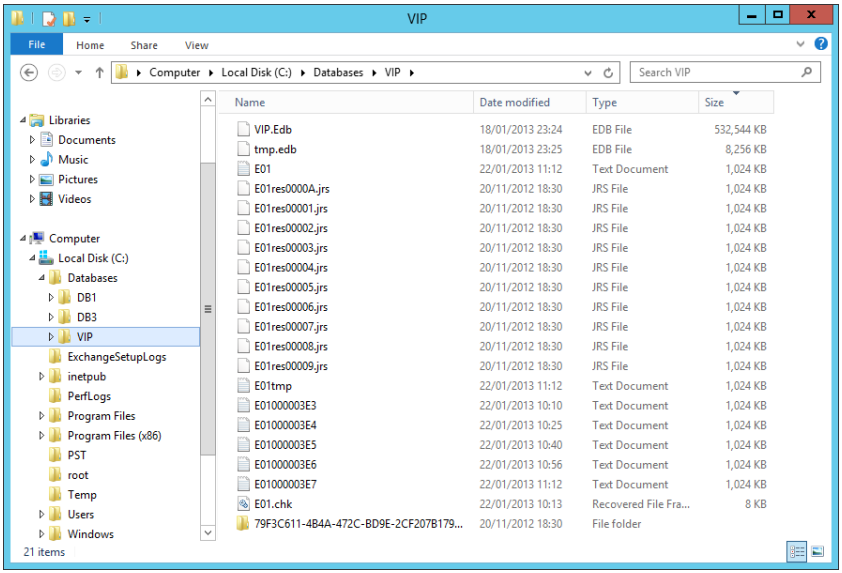


Figure 8-4 Files for a mailbox database

The most important files you see in the directory used to hold an Exchange database are:

- **Database file** For example, Vip.edb. All the tables and data required to provide content to clients are held in the database.
- **Current transaction log** For example, E01.log. This log file is the one into which the Store is currently writing active transactional data.
- **Previous transaction logs** The remainder of the transaction log set that holds previously captured transactions. The Store regularly removes these log files automatically (if circular logging is enabled) or following a successful full backup. The transaction log set forms the basis for replication of data between servers to keep database copies updated within a DAG. In Figure 8-4, the transaction log set goes from E0100003E3 to E0100003E7 (generations 995 to 999, indicating that the current log is generation 1000).
- **Checkpoint file** For example, E01.chk. The checkpoint file tracks the progress of the Store in writing logged transactions (those in its cache and captured in transaction logs) into the database.
- **Reserved logs** For example, E01res00001.jrs. Exchange uses these files to provide some additional space for the database to flush transactions into if the need arises.

Some data about active database transactions might be stored in Tmp.edb. This file is usually only a few megabytes in size and is deleted whenever the database is dismounted or when the Information Store service stops. The Store logs transactions for this database in Exxtmp.log (E01tmp.log in this example).

Ever since the introduction of the DAG, databases are not directly associated with servers (except that they are hosted by servers), and database management is performed at the organization level. Figure 8-5 illustrates a typical view of Exchange Administration Center (EAC) for a small organization positioned in the Servers section of the console. Because these databases don't have more than a single copy, you can conclude that they probably don't belong to a DAG.

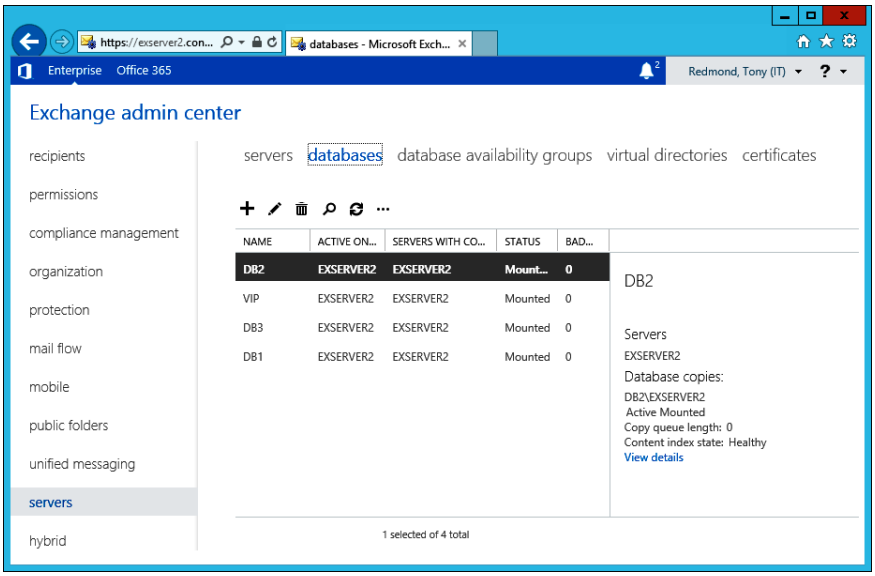


Figure 8-5 Managing databases through the Exchange Administration Center

Unlike EMC, EAC doesn't support filtering of databases to create a specific view. However, EAC does offer a search facility that enables you to look for specific databases. If you need to focus on a certain set of databases, you can do so by applying a naming convention to the databases so that those belonging to specific sites or other logical groupings can be easily identified. Exchange 2013 supports role-based access control (RBAC) scoping to restrict administrative access to specific databases. This is done on a write basis, meaning that EAC does not apply a filter to databases, and administrators are therefore able to view details of all available databases. The RBAC scope comes into effect when an administrator attempts to change a property of a database. If an administrator has the permission to work with the database, EAC will allow him to do so and effect the change. If blocked by an RBAC scope, he will see an error.

Creating new mailbox databases

New databases are created through the Servers node of EAC. It's always a good idea to create new mailbox databases on the server that will initially host them because this avoids the potential for any errors due to network connectivity. Navigate to the Servers section, choose Databases, and then click New (+). Figure 8-6 shows the first step in the creation process when you choose the name of the new database, the server on which Exchange will initially mount the new database, and the location of the database files.

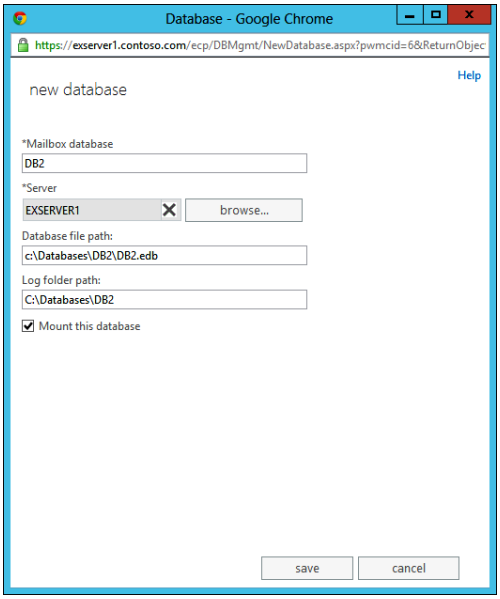


Figure 8-6 Choosing the name and server for a new mailbox database

Locations for both the new database file and the transaction logs must be on a fixed drive (one with a drive letter that does not change). When you create a new mailbox database with EAC, Exchange reads the value of the DataPath property of the server that will host the new database to discover the default database location. You can find this value with a command like this:

```
Get-ExchangeServer -Identity 'ExServer1' | Select Name, DataPath
```

Creating new databases and their transaction logs in a common location under the Exchange root directory is a convenient approach for test servers, but it is not recommended in production. The directories that store mailbox databases should be outside the C:\Program Files\Microsoft\Exchange Server structure because this should be used only for program binaries and other associated files. Best practice is to separate each mailbox database and its transaction logs by placing it in its own directory. Given the size of current

disks and in light of the improved I/O performance of the server, Exchange 2013 allows multiple databases to be accommodated on a single disk, but some still prefer to follow older practice and assign individual disks to databases on the basis that a failure of that disk will only affect a single database. If the directory you specify for a new database does not exist, Exchange will create it.

> **Note**
>
> **You cannot update the DataPath property for a server with EMS because this property is not exposed to the Set-ExchangeServer command. However, you can modify the default path using ADSIEdit by editing the MsExchDataPath attribute for the Exchange server object. Although Microsoft support won't recommend this approach, no side effects appear to be caused by making this change, and if you decide to do this, it is best to be consistent and make the same change on all Mailbox servers.**

In the example shown in Figure 8-6, the transaction logs are placed on the same drive as the database. Except on test systems, this used to be considered bad practice because a hardware failure that affected the database could also affect the transaction logs and potentially remove the ability to recover data into a restored database. Although transaction logs no longer provide the primary protection for database redundancy (unless you run standalone servers), some consideration still needs to be given to where transaction logs are placed and how they are managed.

Of course, if the database is part of a DAG and is protected through replication, things are a little easier because transaction logs are not the sole source for recoverable data, but even with DAG-protected databases, you need to take operational considerations into account before making the final decision of where to place transaction logs. For example, if you think you will move mailboxes on a regular basis, you might want to place transaction logs on a separate drive to ensure that there is sufficient space for the transaction logs that Exchange generates as mailboxes move between databases.

You must mount a new mailbox database after it is created before it can be used. The initial mount of a database prompts the Store to create the underlying files such as the database (.edb), transaction and reserved logs, catalog index, and so on. In Figure 8-6, the Mount This Database check box is selected, which is intended to instruct EAC to issue a *Mount-Database* command after the new database is successfully created. Click Save to give EAC the go-ahead to create the new database.

When the database is created, you'll see a warning message to tell you that the Information Store service should be restarted (Figure 8-7). This indicates that the new ESE cache allocation algorithm is not dynamic and will not factor the presence of the new database into its calculations until after the process is restarted. (See the "Workers, controller, and memory"

section earlier in this chapter.) You can safely leave the restart until you need to create active user mailboxes in the new database.

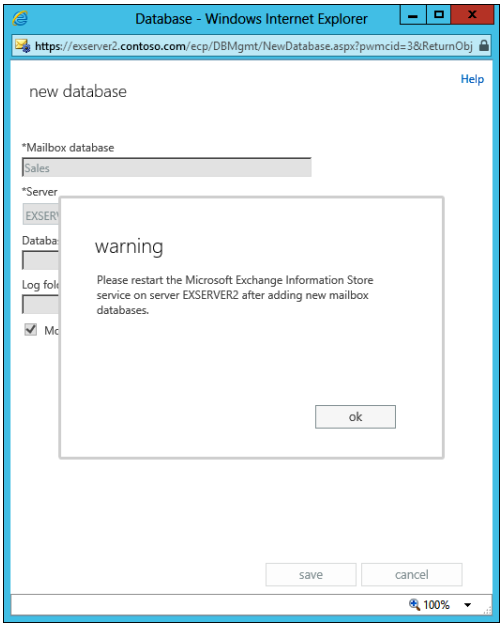


Figure 8-7 Creating a new mailbox database with EAC

An example of typical EMS code that creates a new mailbox database is:

```
New-MailboxDatabase -Server 'ExServer2' -Name 'Sales' -EdbFilePath 'c:\Databases\
Sales.edb' -LogFolderPath 'c:\Databases\Sales'
```

EMS signals the same warning about needing to restart the Information Store service when you run this command to create the new mailbox database. After the database is successfully created, you can mount it and make the database available to host new mailboxes. This step is not automatic, and the new database will remain dismounted until you explicitly mount it, even if the server is restarted to restart Exchange.

```
Mount-Database -Identity 'Sales'
```

Exchange doesn't assign an Offline Address Book (OAB) to a new database. This isn't an issue because Exchange will use the default OAB if no other OAB is assigned. However, it can be a problem if you create multiple OABs for use by different parts of the company and want to assign specific OABs to certain user communities. In such a scenario, it is common to create a mailbox database for each community and assign the correct OAB to the database. You can check the current OAB that is assigned to a database by selecting it and then viewing the Client Settings property tab (Figure 8-8). If the OAB property is left blank,

Exchange uses the default OAB. If this isn't what you want to happen, click Browse to see the full set of OABs known within the organization, select the correct OAB, and save the change.

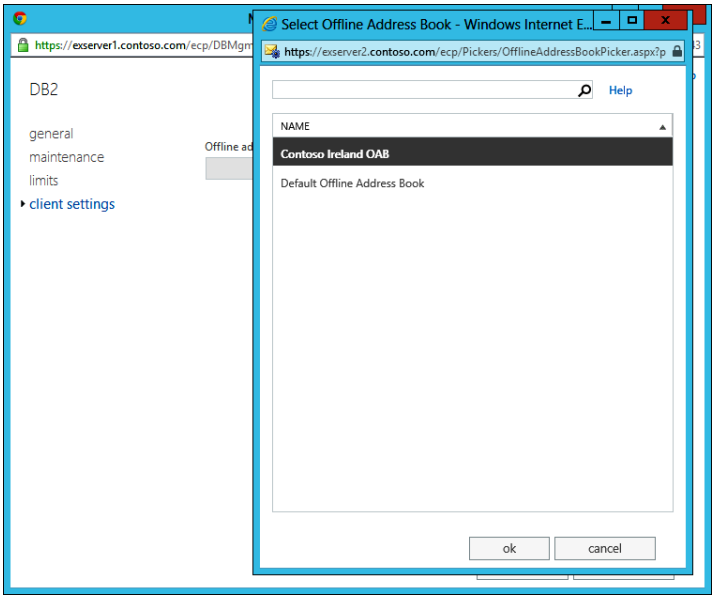


Figure 8-8 Assigning an Offline Address Book to a new mailbox database

In a coexistence situation, in which Exchange 2013 operates alongside older servers, it is important for OABs to be assigned to the databases mounted on the older servers. If this is not done, the danger exists for clients that connect to these databases then to attempt to connect to the Exchange 2013 OAB. You can check for mailbox databases that don't have assigned OABs with the following command. The databases with a blank value returned for the OfflineAddressBook property need to be updated.

```
Get-MailboxDatabase | Select Name, OfflineAddressBook
```

You can update any database that doesn't have an assigned OAB to use the default OAB with this command:

```
Get-MailboxDatabase | Where {$_.OfflineAddressBook -eq $Null} | Set-MailboxDatabase
-OfflineAddressBook '\Default Offline Address Book'
```

New mailbox databases automatically become available as targets for the automatic mailbox provisioning system, meaning that EAC or EMS can immediately create new mailboxes in these databases or that mailboxes can be moved to the database by the Mailbox Replication service (MRS). This might not be what you intended, especially if you create a new database with specific characteristics to support the needs of a particular user

community and want to reserve this database for that community. In this case, you need to block the new database from the automatic provisioning process. See Chapter 5, "Mailbox management," for details about how to exclude or suspend databases from automatic mailbox provisioning.

Updating mailbox databases after installation

When you install an Exchange 2013 Mailbox server, the Setup program creates a default mailbox database for that server and places the new database, its transaction logs, and its search catalog in a directory under the Exchange installation directory. A unique name is assigned to the database to ensure that it will not clash with any other mailbox database within the organization, so you end up with a database placed in a location such as:

C:\Program Files\Microsoft\Exchange Server\V15\Mailbox\Mailbox Database 2136564033\Mailbox Database 2136564033.edb

The location and name of the database is probably not optimal for a production server, so you should consider a number of tasks when the server is up and running properly:

1. Rename the database so that it complies with the organizational naming standard for databases. The name must still be unique, so you have to determine what it should be and then check that the name is unique.
2. Mailbox databases are created with circular logging disabled. You might want to enable circular logging, especially if the database will be used within a DAG. See the "The question of circular logging" section later in this chapter for more information about circular logging.
3. Move the database and transaction log files to a more suitable location away from the disk where the Exchange binaries and other system files are installed.
4. If required, assign an OAB to the database.

Renaming a database is easy. Select the database and view its properties, and then type the new name as shown in Figure 8-9. Alternatively, you can use the Set-MailboxDatabase cmdlet to do the job. For example:

```
Set-MailboxDatabase -Identity 'Mailbox Database 0518033349' -Name 'DB2'
```

You probably will want to move a database to a more suitable location. For example, because all copies of a database share the same path within a DAG, the location assigned to the default database created on a Mailbox server is seldom appropriate. You can either create additional databases and delete the default database or move the default database to a better location.

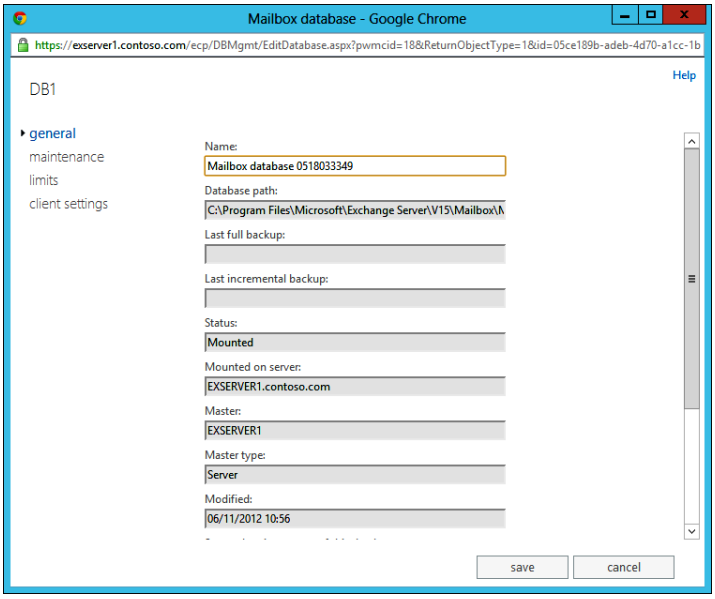


Figure 8-9 Renaming a database

Note

To prevent the potential of interruption caused by network glitches, attempt this operation only when you are connected to the server on which the database is currently mounted. In addition, it's best to settle on a path for a database as soon as possible after it is created and before it becomes part of a DAG because you only have smaller files and a single database copy to deal with. Do not attempt to move a database when a backup is active because the move will fail.

To move the database with EMS, use the Move-DatabasePath command, first dismounting the database if it is online and then mounting it again after the move completes. For example:

```
Dismount-Database –Identity DB4
Move-DatabasePath -Identity 'DB4' -EdbFilePath 'E:\Exchange\DB4\DB4.edb'
-LogFolderPath 'E:\Exchange\DB4'
Mount-Database –Identity DB4
```

The Store has to copy the physical database files from the original location to the new location. Obviously, if you move a database just after it is created, this operation will be very fast because relatively little data must be moved compared to an attempt to move a

500-GB database that has been in production for several months. The fact that a database has moved its location is noted as event 216 from source ESE in the Application Event Log.

If you look at the original location immediately after the database has been moved, you might see that the search metadata files are still there. This is because the Search Foundation is responsible for building the content indexes for the database, and it has to acknowledge the move. The search metadata will be moved to the new location in a short time, and all will be well.

Moving database locations inside a DAG is more complicated because multiple copies might exist for a database, and the copies must be kept in the same location on all servers. See Chapter 9 for more information on this topic.

Backups and permanent removal

You can stop the Store from permanently removing items from databases until a successful backup has been performed. You control this stage of deletion by setting a property of a database (Figure 8-10). The equivalent EMS command is:

```
Set-MailboxDatabase-Identity 'DB3' -RetainDeletedItemsUntilBackup $True
```

When this property is set, items remain in the recoverable items folder until a successful backup is performed, even if they exceed the retention period. You do not want items to expire from the dumpster, be removed by background maintenance, and then be unrecoverable from a backup because of a failure to make a backup or a failure during processing that rendered a backup invalid. It is best practice to set the property on all mailbox databases unless you have a database that is intended to hold only transient information.

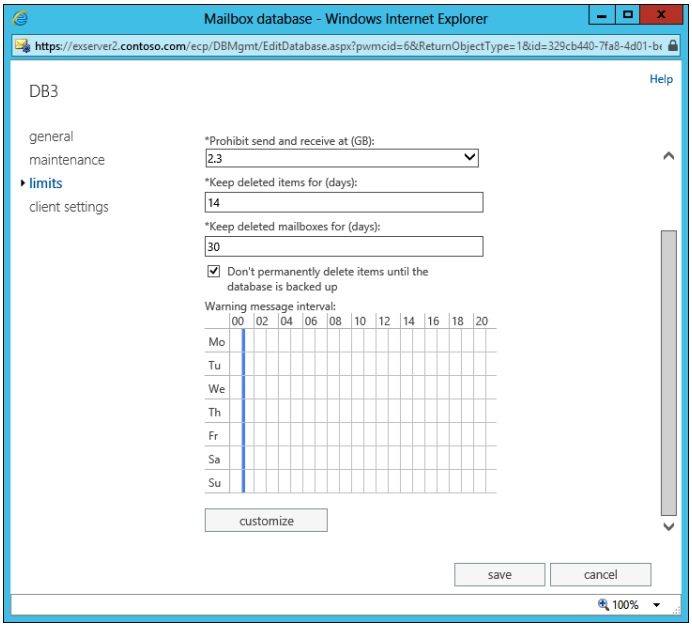


Figure 8-10 Don't delete until the database has been backed up

Removing a database

After you install a new Mailbox server, you might decide that the default mailbox database created on that server is not required. This often happens when you install a new server that will join a DAG and is intended to support copies of databases that are already active in the DAG. In this instance, no real value is to be gained by keeping the default mailbox database, and it is good practice to clean up things by removing the database immediately after the server is installed. Before attempting to remove a database, you should always check that no mailboxes are present, possibly as the result of a new mailbox being created and automatically placed in the database. Exchange won't allow you to remove a database if mailboxes exist in it, and EAC issues a warning if it detects any mailbox. (You can ignore the references to mailbox plans; these are only found in the cloud services.) In any case, it's better to know about potential issues beforehand rather than to find out in the middle of attempting to remove the database, and it's wise to check before attempting to delete a mailbox database that no arbitration, archive mailboxes, or public folder mailboxes exist in the database and that no mailbox move requests are in place that use the database as a source or target.

To check for these possibilities, you can run the Get-Mailbox and Get-MoveRequest cmdlets. For example, the first four of these commands check for mailboxes, arbitration mailboxes, public folder mailboxes, and archive mailboxes. (Note that this command checks for mailboxes on the server where the database is currently mounted to avoid a more general

search across all available mailboxes.) The last two commands look for mailbox move requests associated with the database:

```
Get-Mailbox –Database 'Mailbox Database 031899018'
Get-Mailbox –Database 'Mailbox Database 031899018' –Arbitration
Get-Mailbox –Database 'Mailbox Database 031899018' –PublicFolder
Get-Mailbox –Server ExServer2 | Where {$_.ArchiveDatabase –eq 'Mailbox Database
031899018'} | Select Name
Get-MoveRequest –SourceDatabase 'Mailbox Database 031899018'
Get-MoveRequest –TargetDatabase 'Mailbox Database 031899018'
```


Any mailboxes that are discovered must be moved to another mailbox database before you can remove the database. When all the mailboxes are moved, you can clear the move requests with:

```
Get-MoveRequest –SourceDatabase 'Mailbox Database 031899018' | Remove-MoveRequest
Get-MoveRequest –TargetDatabase 'Mailbox Database 031899018' | Remove-MoveRequest
```

Two calls to Get-MoveRequest are necessary to clear out any requests that use the database as a source or as a target. You can't combine these into a single Get-MoveRequest command. Move requests might linger for a while in a state of Removing after you run the Remove-MoveRequest command. When this happens, it is because the move requests are waiting for MRS to complete the requested clean-up operation. You will have to wait for MRS to do its work before you can delete the database. For more information about working with mailbox move requests, see Chapter 10, "Moving mailboxes."

When everything is ready to allow the deletion to proceed, you can select the mailbox database in EAC and click the wastebasket icon. EAC then prompts you to confirm the action and, if confirmation is given, removes the database.

Removing a database through EAC only deletes the entry for the database from Exchange configuration data in Active Directory. It does not remove the underlying database (.edb), transaction logs, or content index metadata. These files are retained just in case a mistake has happened and you want to bring the database back into Exchange. When you're absolutely sure that the database is no longer required, you can delete the files by using Windows Explorer or File Explorer.

If you prefer, you can run the Remove-MailboxDatabase command instead of going through EAC. Once again, this command only removes information from Active Directory, and you have to clean up the underlying files afterward.

```
Remove-MailboxDatabase –Identity 'Mailbox Database 031899018'
```

Just like creating a new mailbox database, the reduction in database numbers is not incorporated in the ESE cache allocation algorithm until the next time the Information Store service is restarted. This might mean that cache is not managed as effectively as it could be, so it is good to arrange for a restart as soon as convenient.

Transaction logs

Transaction logs are the basic mechanism to capture and record transactions that occur for an ESE database. ESE uses a dual-phase commit for transactions to meet the Atomicity, Consistency, Isolation, and Durability (ACID) test. A transaction is defined as a series of database page modifications that ESE considers a single logical unit. All the modifications must be permanently saved before a transaction is complete and held in the database. For example, the arrival of a new message in a user's inbox is represented by a number of page modifications. The message header might occupy one page, the contents could be held on other pages, and the message might be shared with a number of users, including the originator, so it appears in different folders within the database. ESE has to be able to perform all the updates to modify all pages affected by the transaction to save the transaction. If it does not, ESE discards the transaction.

All transactions that occur in a mailbox database are captured in transaction logs, including system transactions generated because of housekeeping or other background maintenance operations. Because of the asynchronous way the Store saves transactions to the log while batching transactions for efficient committal to the database, it is entirely possible for users to read and write messages in memory without the Store ever going to disk to fetch data.

> **Note**
>
> **Fast, efficient, and secure access to data is the major advantage delivered by the write-ahead logging model Exchange uses, and it is important for every administrator to understand how the model works.**

Log sets

ESE deals with transaction logs as if they formed one very large logical log, which the Store divides into a set of generations to be more convenient to manage. Each log represents a single generation within the log set (also called a *log stream*) and is assigned a sequential generation number for identification purposes. The header of a transaction log contains an identifier (or signature) that associates it with the correct database.

Obviously, a single message that has a number of large attachments can easily span many log files. ESE manages the split of data across the logs automatically and can retrieve data from several logs to form the single message and its attachment if the need arises to replay the transaction into a database. On a busy server, millions of transactions might flow through the logs daily, and it is common to see hundreds if not thousands of logs created each day.

Apart from the activity generated by users, transaction logs capture background and maintenance activity such as the generation of nondelivery reports (NDRs), mailbox moves, the import of mailbox data from other messaging systems, and so on. Any operation that causes data to flow in and out of a database is captured in a transaction log, as is any operation that changes data in place within the database. If you want to estimate the number of transaction logs a server will generate daily, you can use the rule of thumb of 25 MB of data (or 25 logs) per active user per eight-hour working day. The Exchange team's blog asserts that the number of logs per user per day varies from 7 to 42, depending on the average size of message and the number of messages users send and receive, so taking 25 logs is a reasonable middle value that has stood the test of time. Use of circular logging within a DAG reduces the number of transaction logs you have to manage. This topic is discussed in Chapter 9.

INSIDE OUT **Transaction logs should be 1 MB**

Transaction logs are 1 MB. (Earlier versions use 5 MB files.) The 1 MB size was selected from Exchange 2007 onward to facilitate easy replication between servers and to ensure that Exchange can quickly resolve any data divergence that might occur due to an interruption in replication. Choosing a small log size also makes it less likely that data will be lost if a log file cannot be copied or is corrupted due to a storage or other hardware failure. Any variation from the expected 1 MB file size is an indication that the file might be corrupt for some reason. If you see unexpected log sizes, you should stop the Information Store service and check the event log, database, and disks for errors.

The transaction log set for a database is assigned a two-character prefix followed by an 8-digit hex number representing the generation number of the log. Each log in the set is named by combining the log set prefix with the eight-digit hex number. For instance, the file E0000000F61.log represents generation 3937 of the log set associated with the database that uses the E00 prefix. More than 4 billion log files can be created for a database before the Store has to reuse file names. The prefix for the first database created on a server is E00, the prefix for the second is E01, the third E02, and so on, so the current transaction log for the first database is E00.log. The same log prefix is used for the transaction log set for all database copies within a DAG.

In the older form, public folders have their own databases with a quite separate set of transaction logs. These logs capture the same kind of transactions that occur for items in mailbox databases but are maintained as quite separate entities. Modern public folders are stored in mailboxes in regular mailbox databases, and their transactions are interleaved in

the transactions for all the other mailboxes in the same database. Email activity associated with site mailboxes is also captured in transaction logs, as is the synchronization of metadata between Exchange and SharePoint; the work done with documents held in the underlying SharePoint site is recorded by SharePoint.

Every time the Information Store service starts up, the Store automatically checks the databases as it mounts them to verify that the databases are consistent. A flag in the database header indicates whether the database is consistent or inconsistent, depending on whether the Store was able to shut down the database cleanly the last time it shut down. A clean shutdown flushes all data from the Store's cache and commits any outstanding transactions into the database to make the database consistent.

An inconsistent database is one that has some outstanding transactions that the Store has not yet committed. If the Store detects that a database is inconsistent, it attempts to read the outstanding transactions from the transaction logs to replay the transactions into the database and make the database consistent. This operation is referred to as a soft recovery. Exchange initiates a soft recovery for each database on a server when the Information Store service starts up and mounts the databases (Figure 8-11). Event 302 is logged when the soft recovery completes. You see the same set of events recorded if you dismount and then mount a database. Although event 300 proclaims that the database engine "is initiating recovery steps," in this instance, it really means that Exchange is validating the database just in case any transactions are outstanding. If the Information Store service had previously been shut down cleanly, any waiting transactions will have been flushed from cache to make the databases consistent, and no recovery is necessary, but it's good to check.

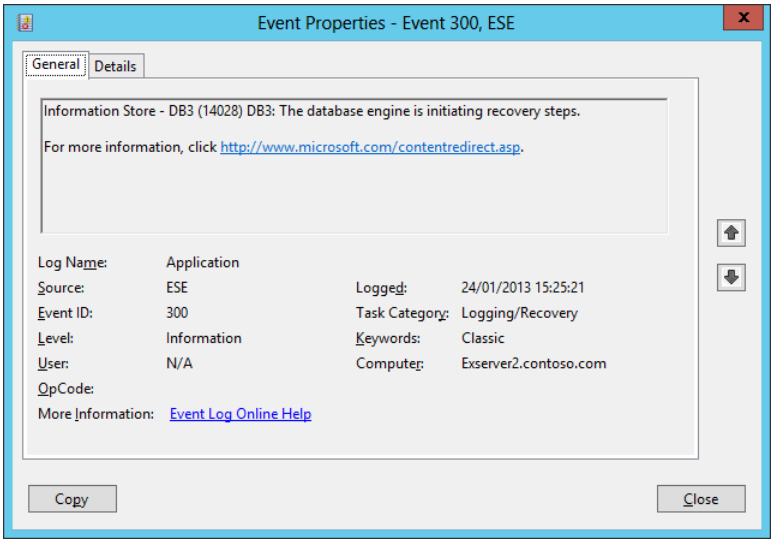


Figure 8-11 ESE logs event 300 as a mailbox database is mounted

When a database has been shut down abnormally, such as when a bug or hardware failure caused a worker process to terminate abruptly or the complete server to halt, Exchange goes through a real soft recovery process to replay all outstanding transactions from the logs to update the databases before clients are allowed to connect. Such recoveries occur much less frequently today than with earlier versions of Exchange. No database property is accessible through EAC or EMS to validate that a database is consistent, so if you really need to know this, you can run the ESEUTIL utility with the /MH parameter to check the database header. Databases that have been dismounted properly will report a state of Clean Shutdown.

Dismounting databases or stopping the Store makes them consistent, albeit inaccessible. Taking a full backup creates a somewhat similar situation in that the database after the backup is consistent. However, because the database is still online, the Store continues to commit transactions immediately after the backup finishes, so the database is only consistent for a very short time.

Transaction logs are tied to their databases in two ways. First, ESE writes a unique identifier (or signature) into each log as it creates the file. The log signature must match the signature of the corresponding database before ESE can use the contents of a log to recover transactions. Second, ESE records the path to the directory where the database is located in the transaction logs. You can find information about identifiers and locations by running the ESEUTIL utility with the /ML parameter to dump the header information from a transaction log as shown in the following sample. Note the signature reported for the transaction log and the identical value reported for the database.

```
Extensible Storage Engine Utilities for Microsoft(R) Exchange Server
Version 15.00
Initiating FILE DUMP mode...

    Base name: e01
    Log file: c:\databases\vip\e01000003fe.log
    lGeneration: 1022 (0x3FE)
        Checkpoint: NOT AVAILABLE
    creation time: 01/25/2013 10:54:51.743
    prev gen time: 01/25/2013 10:47:24.289
    Signature: Create time:11/20/2012 18:30:16.574 Rand:1021831792 Computer:
    Env SystemPath: c:\databases\VIP\
    Env LogFilePath: c:\databases\VIP\
    1 c:\Databases\VIP\VIP.Edb
        dbtime: 1540528 (0-1540528)
        objidLast: 5397
        Signature: Create time:11/20/2012 18:30:16.699 Rand:365102769 Computer:
```

After a client submits a message to the Store, an ESE session that is responsible for the transaction follows a well-defined order to apply the transaction to commit the new message to the Store. The same order is followed for other transactions such as deletes and moves. First, ESE obtains a timestamp using the internal time (called a *dbtime* held in an 8-byte value) maintained in the database header. To modify a page, ESE must calculate a new dbtime based on the current value in the header. When it has calculated the new dbtime, ESE writes the records that make up the transaction into the current transaction log. After this operation completes, the session can modify the appropriate pages in the database. Page modifications occur in an in-memory cache of dirty pages, so ESE might first have to fetch the necessary pages off the on-disk database.

When a client performs an operation that modifies a database page, ESE follows this sequence:

1. The database page is fetched from the ESE in-memory cache. If the page is not cached, ESE fetches it from disk.
2. A log record is generated to describe the page fetch and update the cache.
3. The database page is modified and ESE marks the page as dirty. The page is not written immediately to disk because it might be modified by subsequent transactions. The version store tracks dirty pages to make sure that they are accounted for in case the database session is terminated abnormally. The version store is an internal component that keeps an in-memory list of modifications made to the database that the Store uses for purposes such as transaction rollback or to resolve attempts to apply multiple modifications to the same page.
4. The database page is linked to the record in the cache to prevent ESE from flushing the page to disk before the transaction log record is written. ESE always commits pages to the database after it is sure that the pages have been successfully captured in a transaction log.
5. When the log buffer (1 MB) is full or a commit record is recorded for the transaction, ESE commits the changed page by recording it in the current transaction log. On idle servers, this operation might require a log roll (the creation of a new log generation). If a database is replicated, Exchange can use block replication to share information with servers hosting other copies of the database. Transaction logs continue to be copied. See Chapter 9 for more information about how block replication works.
6. Eventually, the dirty pages are flushed from memory and written into the database.
7. The checkpoint is advanced.

To ensure that data are always protected, it is a cardinal rule for ESE that database writes cannot occur before their transactions are first committed into a transaction log. If you look at the steps that make up a complete transaction, you see that the last step is to commit the transaction to disk. For example, the last step shown in Figure 8-12 is a commit command for the transaction identified as ESE session 8. Other prior steps in session 8 begin a transaction and insert some data to replace an existing page. The commit is a synchronous operation, so no other transaction can occur for that session until the write to disk is complete. Enabling battery-protected write-back caching on the disk that holds the transaction logs improves performance by allowing the write to complete in the controller's memory and thus release the synchronous wait. The controller is then responsible for writing the data to disk.

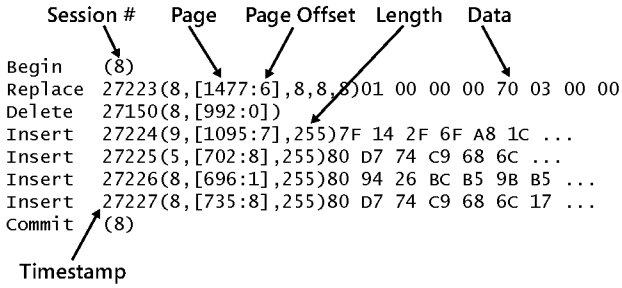


Figure 8-12 Data in a transaction log (from Exchange 2010 Inside Out)

Each log record represents an individual page modification. Transactions start with a Begin transaction record and proceed with the individual operations until the transaction is complete, a Commit transaction record is added, and the transaction is committed to the transaction log. The sequence of transaction records enables ESE to replay complete transactions into databases when it needs to recover data. If ESE reads in a transaction and cannot find a commit record, it considers the transaction incomplete and will not replay it into a database.

The delivery of a single new message causes ESE to modify many pages because many tables might need to be updated. ESE also updates the index for each table, and if the message contains a large attachment, its content will be broken up into several *long value* chunks (literally, large chunks of data that collectively form the item's content), all of which generate log records. All these transactions are captured in log files. If you monitor the log file directory for a database when a large message is delivered to multiple mailboxes in that database, you will observe a spurt of log generation activity. For example, a 1-MB message sent to 10 mailboxes generates at least 10 transaction logs.

However, if you delete an item that has a very large attachment, Exchange only needs to capture the page numbers of the pages that now contain deleted data, and the actual content does not appear in the logs. As you can see from Figure 8-12, the entries in a log

represent low-level physical modifications to the database, and records for different transactions are interspersed throughout a log file. Replaying a transaction is therefore quite a complex matter. Each transaction log contains a sequential list of operations the Store has performed on pages in memory. The log captures details of when a transaction begins, when it is committed, and whether the Store needs to roll it back for some reason.

Each record in the log is of a certain type. Record types include Begin (a transaction is starting), Replace (some data in a page is being updated), Delete (data is removed), Insert (data is added), and Commit. Transactions from multiple sessions are interleaved throughout a transaction log. This means that the Begin record type also identifies the session that performed a transaction. You can think of a session as a thread running within the Store process. The session forms the context within which ESE manages the transaction and all the associated database modifications. Each session could be tied back to a particular client, but the database has no knowledge of individual clients (MAPI or otherwise) because all it sees are the threads that operate on its contents.

Figure 8-12 illustrates how a set of transactions might appear in a log. In this example, the first transaction in session 8 (or thread 8) is replacing a record in the database. Every physical modification to a database is time stamped. ESE uses timestamps later if it has to replay transactions from a particular point in time. The page number and an offset within the page are also recorded. The length of the data to be replaced is then noted and followed with the actual binary data that is inserted into the page. The next transaction is a record delete. The set of insert transactions demonstrates that transactions from multiple sessions are intermingled within a log. Sessions write data into the log file as they process transactions. Any dump of a log file from even a moderately busy server will record transactions from scores of sessions.

When the Store replays transactions from logs to make a database consistent, it has to interpret the contents of all the transaction logs that have accumulated since the last good backup to find the transactions that it requires and then assemble the different records for those transactions from the logs and replay them into the database.

Dirty pages in the cache are flushed to disk every 30 seconds. Pages are not necessarily flushed in order because they might be modified several times during their life in the cache. Along with flushing, ESE calculates a checkpoint to mark the current recovery point so that it knows what transaction logs are required to update a database and make it current if it needs to replay logs in case of an outage. Each transaction log has a generation number that is incremented by 1 as each 1-MB transaction log file is closed off. The log generation number tracks the position and sequencing of transactions in a log file in the log stream. ESE uses the generation numbers to determine the exact log files it needs for any recovery operation. The generation number of the last committed log file is also written into the checkpoint so ESE can calculate the required log files. For example, assume that the current

log file generation is 0x36b0 (14,000), and the log file generation written into the checkpoint file is 0x3683 (13,955). In simple terms, using these values, you can say that:

- Any log files with a generation value less than 0x3683 that contain complete transactions (begin and commit records) have had their transactions written into the database.
- Any log files between the checkpoint value (0x3683) and the current generation (0x36b0) might have some of their transactions written to the database, but because you cannot be sure of what these transactions are, all the transaction logs between 0x3683 and 0x36b0 are required for replay purposes if you have an outage and need to recover the database. The last generation number required is called the waypoint, so you can say that the difference (in generation numbers) between the checkpoint and the waypoint represents the set of transaction logs required for recovery.

Transaction logs hold only 1 MB of data. On heavily loaded systems, when new transaction logs are being created all the time, it is possible for new generations of logs to exist past the waypoint. These logs contain transactions that have not been committed into the database. The transactions could be incomplete (no Commit record exists) or are in the process of being written when the failure occurs. For whatever reason, these logs are not required for recovery purposes. However, if a failure occurs and the database is updated, the transactions in these log files will be ignored and lost.

From this description, you'll realize that transaction logs are critical to the good health of any Exchange server. Although Exchange servers can use block replication to transfer data between them to synchronize database copies, this mechanism only works when networks are very reliable. Transaction logs provide the backstop for Exchange replication, and, without the logs, databases would not be as reliable and robust as they have proved to be in production.

Checkpoint file

The checkpoint file enables the Store to know which transaction logs are required if it needs to perform a recovery. The Store updates the checkpoint file every 30 seconds when it flushes dirty pages into the database. There is always a separate checkpoint file for a set of transaction logs (otherwise known as a log stream), and because Exchange maintains a set of transaction logs per database, it follows that there's a separate checkpoint file per database.

When a database is mounted, Exchange reads the checkpoint file to determine whether it should process any outstanding transaction logs to bring the database completely up to date. This is known as a soft recovery, and it happens every time a database is mounted to ensure that an inconsistent database (one that has outstanding

transactions) is restored to full health even if it has been shut down unexpectedly due to a computer failure or software bug. Unless tens of thousands of logs are necessary to update a database, a set of transaction logs usually doesn't take long to replay. The checkpoint file enables Exchange to quickly identify the point at which the last written transaction occurred and avoid the overhead of examining all available transaction logs and assessing whether the transactions they contain are in the database.

However, if the checkpoint file is lost for some reason, Exchange examines all available transaction logs to determine whether any transactions exist in the logs that have not been committed into the database. If some transaction logs are unavailable due to storage failure or another reason, you have to run the ESEUTIL utility with the /p switch to make the database consistent again and allow Exchange to mount it. In this situation, it is more than likely that users will experience some data loss because there is no way to recover the transactions in the lost logs.

Transaction log checksum

Every transaction log contains a checksum that Exchange validates to ensure that the log data is consistent and valid. Microsoft introduced the checksum to prevent logical corruption from occurring as the Store replays transactions into a database during a recovery process.

Exchange uses a type of sliding-window algorithm called Log Record Checksum (LRCK) to validate checksums for a selected group of records in a log to ensure log integrity. The Store reads and verifies these checksums during backup and recovery operations to ensure that invalid data are not used. If the Store detects invalid data through a checksum failure, it logs a 463 error in the system event log. If the Store fails to read the header of a transaction log and is unable to validate the checksum, it signals error 412 in the Application Event Log. Transaction log failure inevitably leads to data loss because the only way to recover from this error is to restore the last good backup. All the transactions since that backup might be lost.

Transaction log I/O

The Store always writes transactions in sequential order and appends the data to the end of the current transaction log. All of the I/O activity is generated by writes, so it is logical to assume that the disk where the logs are located must be capable of supporting a reasonably heavy I/O write load. In comparison, the disks where the databases are located experience read and write activity as users access items held in their mailboxes.

On large servers, the classic approach to managing the I/O activity generated by transaction logs is by placing the logs on a dedicated drive or LUN. This solves two problems. First, the size of the disk (today, usually 2 TB or greater) means that free space should always be available. If you accumulate 500 GB of logs (500,000 individual log files), it means that your server is under a tremendous load, that you're using a lagged database copy, or that the Store has not been able to truncate the logs for some reason (for example, no recent successful full backup). Second, in all but extreme circumstances, a dedicated drive is capable of handling the I/O load generated by transaction logs.

Arguing against the classic approach is the fact that the Exchange developers now favor placing transaction logs on the same volume as their database. This would have been heresy in the past because of I/O concerns and the perceived need to keep logs and database separated so that a disk failure would not affect both. The reason the developers have changed tack is simple. First, disks are getting larger and are capable of holding both database and logs without ever running out of space. In fact, individual disks are now so large that they can host multiple databases. Second, the advent of multiple database copies means that losing a disk is a lot less serious than it was in the past. Indeed, database auto-reseed now allows a new database copy to be created without administrator intervention. Third, the dramatic and continued reduction in I/O generated by Exchange means that single disks are capable of handling demand. In fact, the continuing improvement in the I/O profile of Exchange makes fast disks, including solid-state drives (SSDs), somewhat of an unnecessary luxury except in very specific circumstances in which their deployment is justified by requirements that are verified through testing. All of this goes to prove that there is no one good answer to the question of how to lay out databases and transaction logs across available storage. Instead, the flexibility of Exchange is seen in that multiple approaches work.

INSIDE OUT Applying some logic

Having a dedicated drive for log files is a luxury you might not be able to afford. But the logic applied to justify the drive—reserve enough space for log file growth and keep an eye on I/O activity—should be remembered when you decide where the logs should be stored on your system. For example, it's a bad idea to locate the logs on the same drive as other hot files such as a Windows page file.

As noted in the earlier discussion about creating a new mailbox database, another classic best practice is never to place the logs on the same drive as a database. The logic is that keeping the logs with their database might seem like a good idea, but like keeping all your eggs in one basket, if a problem afflicts the disk holding the database, the same problem will strike down the transaction logs, and you will lose data. This advice

still holds for databases that are not replicated, but it becomes less valuable the more database copies you have within a DAG. If you only have two database copies, database and log isolation across different disks is still a good idea because your tolerance for failure is limited to one copy. However, if you have three or more database copies, you can collocate databases and transaction logs because any failure will still result in the availability of at least two databases.

The question of circular logging

Circular logging means that the Store generates a limited number of transaction logs by reusing logs as their contents are committed into the database. This feature was originally introduced into Exchange as a mechanism to restrict disk usage on low-end servers because storage was an expensive commodity in 1996. Circular logging is also common on test servers and edge transport servers, where data are transient and don't usually need to be captured for a permanent record. From an operation perspective, the Store uses a relatively small set of between four and eight logs to capture transactions at low periods of demand. However, this figure depends on system load and how quickly the Store can commit transactions to the database—and whether the database is replicated. It is quite common to see thousands of transaction logs created for a database that is under heavy load (for example, when many mailboxes are being moved into the database), and these logs might persist for some time, especially if they are being replicated to other servers. Eventually, as system load decreases and replication occurs, the set of logs used for circular logging is truncated back to a minimal set and kept at this level until system load grows again.

In the past, circular logging was never enabled for production databases. The introduction of the DAG and the ability to have multiple copies for databases challenged the need to retain transaction logs until backups were complete. Circular logging within a DAG is explored in more detail in Chapter 9.

Note

By default, new mailbox databases do not use circular logging, so if you want to enable circular logging to save disk space, you have to update the database properties after it is created. The new logging behavior becomes effective the next time the database is mounted. Don't enable circular logging if you plan to replicate the database within a DAG because you'll have to disable it before you can transition the database to have copies.

To enable circular logging on a mailbox database, select the database with EAC, view its maintenance property page, and then select the Enable Circular Logging check box. The equivalent EMS command is:

```
Set-MailboxDatabase –Identity 'DB1' –CircularLoggingEnabled $True
```

After you enable circular logging for a nonreplicated database, you must dismount and mount the database to make the setting effective. These operations enable the Store to clear its cache of transactions for the database and close off the current transaction log stream in preparation for switching mode. Use the following commands to dismount and mount a database in EMS:

```
Dismount-Database –Identity DB1 | Mount-Database DB1
```

To dismount and mount a database from EAC, make sure that the correct database is selected, and then click the ellipses (...) button to reveal the option to change database mount status.

Circular logging will not prevent disk space exhaustion, and disks still need to be monitored carefully to ensure that sufficient storage is available to accommodate logs even if they have to be retained for an unexpected period.

When circular logging is disabled, the Store continuously creates new transaction logs as it fills the current log with data. The Store performs the following steps to create a new log and switch it to become the current log. First, the Store advances the checkpoint in the checkpoint file to indicate that it has committed the transactions in the oldest log into the database. Next, the Store creates a temporary file called *<database prefix>tmp.log*. For example, E00tmp.log is created for the first database created on a server. The Store switches this file into use to become the current log when it closes off the current log to ensure that transactions can be continually written into a log without pausing.

Note

If the Store fails to create the temporary file, it usually means that no more disk space is available on the disk that holds the transaction logs. In this case, the Store proceeds to an orderly shutdown of the Information Store service and uses the two reserved log files to hold transaction data that it flushes from its cache during the shutdown.

When the temporary log file is created, the Store initializes its log header with the generation number, database signature, and timestamp information. When the current log file is full, the Store stops writing transactional data and renames the current log file to incorporate the generation number in its file name (for example, E000007E71.log). The Store then renames the temporary log file to be E00.log (or whatever the prefix is for the database)

to make it the current transaction log and resumes writing transaction data into the current log file, flushing any transactions that have accumulated in the short time required to switch logs.

The Store continues with the process of creating the temporary log file and switching it to take the place of the current log file until the Information Store service shuts down. The number of logs created on a server varies according to message traffic and other activity, such as mailbox moves. Busy and large servers can generate tens of gigabytes of transaction logs daily, so this is obviously an important factor to take into account when you size storage for servers. It's also important to include a healthy margin to account for peaks in demand, including those that might occur due to software bugs such as the extreme growth in transaction logs caused by malfunctioning Apple iOS ActiveSync clients in 2012. The space occupied by transaction logs is released when the files are truncated following a successful full backup. The logic here is that you do not need the transaction logs anymore if you have a full backup. The database is consistent after the backup, and you have a backup copy if a problem arises. The backup copy also contains the transaction logs that are necessary for the Store to make the database consistent if you need to restore it.

Reserved logs

Reserved logs provide the Store with some emergency storage in case it is unable to create new transaction logs for any reason. The reserved logs are called databasePrefixres00001.jrs through databasePrefixres0000A.jrs. For example, E00res00001.jrs is the first reserved log file for the first database created on a server.

The only time I have seen these files used is when space is exhausted on the disk that holds the transaction logs. Running out of disk space was a reasonably common occurrence in the early days of Exchange because disks were small and expensive, but such an event is much less common today. Most administrators now monitor free disk space on all disks Exchange uses and take action to release space if less than 1 GB is available.

You should never get close to the point at which the reserved logs are called into play. First, you should ensure that all the disks that hold mailbox databases or transaction logs are sufficiently large to accommodate normal operation and any expected growth over the next year or so. The exact size of the drives you need to use differs from organization to organization and should be dictated by actual data for database growth over time. Second, you should monitor available space on all relevant disks on an ongoing basis. Third, if your monitoring fails or you haven't provided sufficient space to account for an expected spurt in growth, Exchange includes a self-protection mechanism (backpressure) that prevents the transport service from delivering new messages to the mailboxes in a database if the available space on a disk that holds a mailbox database or transaction log falls below 1 GB. (For instance, a database and its set of transaction logs might grow considerably if you move many mailboxes to the database.) This step prevents new messages from causing a

problem, but it might not stop other procedures (such as mailbox moves) from grabbing some more space. Exchange releases the block on new message deliveries when the available space on the disk grows past 1.5 GB. Exchange could release the additional space as a result of removing transaction logs after their contents are committed into the database or through administrator intervention.

INSIDE OUT **Less than 1 GB of free space requires action**

The fact that free space is reduced to less than 1 GB is a danger signal that warrants immediate and proactive attention from the administrator to take the necessary steps to free space on the disk so the situation cannot recur.

Background maintenance

All databases have internal structures that require some degree of maintenance to ensure logical consistency, and Exchange is no different. You can take a database offline to rebuild it with the ESEUTIL utility, but a database rebuild requires you to deprive users of access to their mailboxes. Microsoft designed Exchange to be highly available with as little downtime as possible, and online maintenance is necessary to keep the databases in good order while allowing users to continue working.

The maintenance Exchange performs can be grouped into the following categories:

- **Miscellaneous content maintenance** These tasks are performed during the maintenance schedule assigned as a property of mailbox databases, typically from 1 A.M. to 4 A.M. nightly. Given the speed of modern servers, these tasks usually complete quickly.
- **Checksum scans** A checksum is determined for every page in mailbox databases to detect possible physical corruption in an Online Database Scanning process. This process usually occurs on a 24/7 basis but can be assigned a specific schedule.
- **Defragmentation** Exchange attempts to reorder pages in mailbox databases so that data that belong together are stored sequentially to make I/O more efficient. This process runs on a 24/7 basis.
- **Compaction** Exchange attempts to free space within mailbox databases by rearranging data into as few pages as possible. This process runs on a 24/7 basis.

- **Page zeroing** Exchange writes zeros into deleted pages to obscure the data these pages once held. This process runs on a 24/7 basis.

Exchange 2013 uses two work cycle–based agents called StoreMaintenance and StoreDirectoryServiceMaintenance to perform Store maintenance operations in the background. These operations happen constantly and are not constrained by an assigned maintenance window.

Database checksums

To ensure that no physical page corruption exists in its databases, Exchange reads sequential 256-KB chunks of data at a rate of approximately 1 MB per second (per database) and then verifies the checksum for each of the eight pages in the chunk. This process is called Online Database or ESE Scanning, and its goal is to make sure that every page in every database (active and passive) is scanned on a server at least once weekly. If the scan takes longer than seven days to complete, ESE logs event 733 in the Application Event Log to notify that processing is taking longer than expected and then logs event 735 when the scan completes.

The default for Exchange 2013 is to run online scans on a 24/7 basis. On most systems, online scanning deals with pages at a rate of roughly 3.5 GB/hour (depending on server load, speed, and the storage system), so it is relatively easy for Exchange to go through every page in a database at least once every three or four days. These values are throttled back in Exchange 2013 from the previous rates used in Exchange 2010 (roughly 5 MB/sec) to improve the I/O profile of the Mailbox server.

If you look at the properties of a mailbox database (Figure 8-13), you see that the check box to enable background maintenance is selected. In fact, "background maintenance" is a misnomer because this check box only controls how Exchange performs checksum scans. The property can also be set through EMS by running the Set-MailboxDatabase command:

```
Set-MailboxDatabase –Identity 'DB3' –BackgroundDatabaseMaintenance $True
```

If you opt not to enable 24/7 scanning, Exchange performs checksum scans as the last stage in all the other background tasks that it performs to keep databases healthy, such as removing deleted mailboxes when their retention period expires. These tasks are executed during the nominated maintenance window, which you can also see in Figure 8-13.

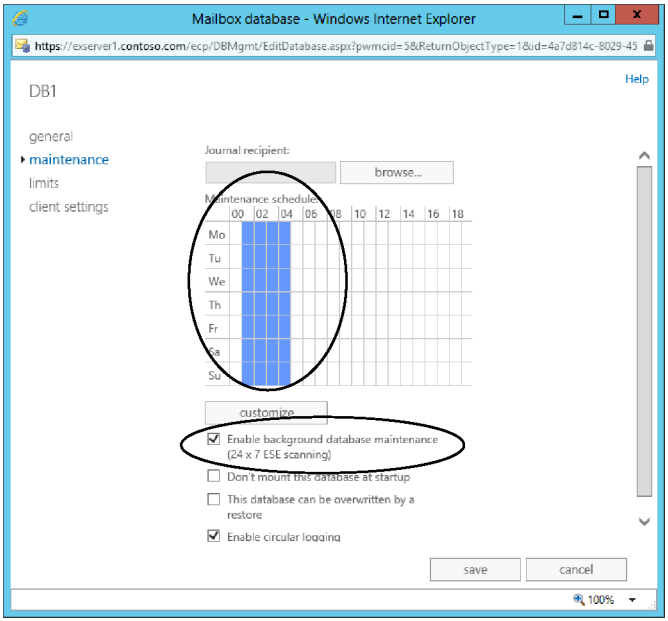


Figure 8-13 Controlling background database maintenance

You can also use the Get-MailboxDatabase command to retrieve the current maintenance schedule for a database. For example:

```
Get-MailboxDatabase -Identity 'DB1' | Format-Table Name, MaintenanceSchedule
-AutoSize
```

Given the format of the schedule, manipulating it by running the Set-MailboxDatabase command could be tricky. It's usually much better to make changes through the graphical interface of EAC to set the maintenance schedule on a database.

If you clear the Enable Background Database Maintenance check box in Figure 8-13, you constrain checksum scans for active databases to whatever maintenance period is assigned to the database (1 A.M. to 4 A.M. in the example shown). Given that other work needs to be done during this time, it's likely that checksum scans would only be performed in the last 150 minutes of the period. This is sufficient for small to medium databases but not recommended for any database that is larger than 500 GB due to the risk that scanning cannot process every page in the database in a reasonable time and so might not discover a corrupt page quickly enough for patching to be possible. Note that online scanning occurs continuously for passive database copies, even if you set up a customized schedule for a database.

All things considered, it is best to leave the default arrangement in place and let Exchange throttle operations to meet the goal of scanning every page in every database at least once weekly.

If a corrupt page is discovered during scanning, Exchange can patch the page from another copy of the database. However, this can only work when multiple copies are available, so it is not a feature available to databases located on servers outside a DAG. See Chapter 9 for more details of how page patching is performed.

INSIDE OUT **So many pages, so little time**

The reason for introducing 24/7 checking is simple. Confining the work to a set period at night is not a problem with small databases because you can be reasonably confident that Exchange can process all the pages in a database during the maintenance period. However, as databases become larger and larger, more pages must be processed. The Store processes pages sequentially from the start of the database to the end and, as the number of pages to be validated grows, it becomes more difficult for the Store to complete the task within the allocated maintenance window, hence the need for Exchange to accommodate a 24/7 processing cycle.

Database defragmentation

Online defragmentation (known as *OLD* or *B+ tree defragmentation*) reorganizes pages within the database so that they are in the most efficient structure. The current method of online defragmentation was first introduced in Exchange 2010 to replace an older and less efficient version and operates on a 24/7 basis. You cannot change a setting to influence how Exchange performs defragmentation because the Store is designed to monitor database operations with an eye to optimization. If the Store detects that requests coming in to read information are for records that are not held sequentially, it defragments that portion of the database to reorganize data into pages so that future requests can be dealt with sequentially. The intention is to minimize the number of random I/O requests the Store has to process and gradually improve the efficiency of the database. In addition, the Store monitors all database tables for free pages, and if it detects that a table has too many free pages, it moves those pages back to the root and makes the pages available for reuse.

You can track the work done to defragment databases with Performance Monitor by using the MsExchange Database/Defragmentation Tasks counters.

Database compaction

Because the nature of email is for information to accumulate steadily in mailboxes over time, it's reasonable to expect that an Exchange mailbox database will grow and grow, which is exactly what happens. An Exchange database will never shrink and return space to the Windows file system unless the database is taken offline and rebuilt using ESEUTIL.

Given that the tendency is for databases to swell over time, it's important for the internal structures of the database to be as efficient as possible in terms of how pages are used to store information. If information is stuffed into pages any old way, databases would grow even faster as disk space is continually requested from Windows to store new items.

Database compaction helps by moving information around within a database so that data are held within the smallest number of pages possible. At the same time, compaction attempts to keep data ordered contiguously so that all the data required to satisfy requests can be retrieved using sequential reads. For example, all the pages needed to store information about item headers might be moved to sequential pages so that Exchange can fetch all the headers for items in a folder using one sequential I/O rather than having to issue multiple smaller I/Os to different points within the database.

Page zeroing

Exchange overwrites the contents of deleted database pages with zeros to ensure that information previously held in those pages cannot be easily retrieved. Many companies that are concerned about the possibility that a rogue administrator might compromise the information held in a mailbox database request this security measure. Given that a rogue administrator has easier methods available to her to access user mailboxes if she desires, it's hard to see her making a copy of a database and then rummaging around in its internals to uncover deep and dark secrets, but it is possible.

In any case, Exchange 2013 overwrites pages with zeros as pages are freed because of items being hard deleted from the database. This process happens continuously and cannot be controlled by an administrator. In addition, during online scanning, Exchange checks for deleted pages that have not yet been zeroed for some reason, such as the Store being overloaded when a page was freed or a bug interfering with normal processing. Online scanning also takes care of zeroing the pages used for deleted tables and indexes because these are not zeroed when they are freed.

Content maintenance tasks

The maintenance tasks discussed so far have focused on raw database structures such as pages. Some of the information held in the database also needs some tender loving care. Clients such as Outlook make extensive use of the Store's ability to generate indexes or views on a dynamic basis. For example, if you decide that you want to view your inbox by

author rather than by date, Outlook asks the Store for this view. If the Store has previously generated the view and has it cached, the response is very quick, but the Store can process a request for a brand-new view quickly, too. Over time, the Store accumulates large numbers of views; each folder in every mailbox can have several views.

Expiry dates control the accumulation of views in the database

It is not desirable to retain all the views because each view occupies space within the database. In addition, users might not require a view after its first use. The Store assigns an expiry time to each view and monitors these data in an internal table called the index-aging table. When background maintenance runs, the Store scans the index-aging table to discover views that are older than 40 days and removes any view that has expired. Of course, if a client accesses a view, the Store resets its expiry time.

Another task is to perform *tombstone maintenance*. The Store maintains a list of deleted messages for each folder and a list of tombstones to indicate that a message was deleted. After successful replication, the Store can clear the entries from the tombstone list. Background maintenance ensures that the lists are accurate.

During background maintenance, the Store examines every message that has the deleted flag set to determine whether its retention period has expired. (If enough time is available in the maintenance window, the Store processes the entire contents of the deleted items cache.) If this is true, the Store removes the message from the Store (a hard delete) and makes the pages that hold the message available for reuse. You can set retention periods on a per-mailbox, per-database, or per-public-folder basis and can control these settings through EAC or EMS. The Store also checks for deleted mailboxes and removes any that have expired.

Corrupt item detection and isolation

Bad items or messages that contain some corruption such as a malformed header have a nasty habit of causing extreme difficulty for an Exchange administrator. Exchange 2007 introduced the ability to detect and isolate corrupt messages in the transport system. Essentially, if the transport system thinks that a message is corrupt and likely to cause problems such as provoking a software crash, it considers the message poison and isolates it in a queue the administrator can examine to decide whether the message really is a problem. Exchange can identify and then isolate or quarantine problematic mailboxes to address when a mailbox that contains one or more corrupt items can cause the Information Store process to terminate abnormally when a client attempts to access a corrupt item.

As you know, the Information Store process is multithreaded. The threads are connected to mailboxes to do work, and the Store detects a problem mailbox when:

- The Store has more than five threads connected to a mailbox that have been frozen for 60 seconds or more.
- One of the threads connected to a mailbox crashes abnormally. The nature of software is that crashes do occur, so Exchange allows for up to three crashes within a two-hour period before it regards a mailbox to have exceeded the acceptable threshold for crashes.

When the Store identifies a problem mailbox, it writes details about the mailbox into the system registry. These details include the mailbox globally unique identifier (GUID) to identify the mailbox, the time a crash occurred, and the number of times the problem has occurred. This information is stored under the root for the mailbox database that contains the mailbox in two entries written for a problem mailbox. The first entry captures the last crash time, the second the count of crashes.

- HKLM\System\CurrentControlSet\Services\MSExchangeIS\Server Name\Private-{Database GUID}\QuarantinedMailboxes\{Mailbox GUID}\LastCrashTime
- HKLM\System\CurrentControlSet\Services\MSExchangeIS\Server Name\Private-{Database GUID}\QuarantinedMailboxes\{Mailbox GUID}\CrashCount

When a mailbox exceeds the crash threshold or the Store considers that threads are being blocked by corrupt content, the Store puts the mailbox into quarantine. This means that normal client interaction, including access by Exchange mailbox assistants, is blocked. If you check the Application Event Log, you'll find a 10018 event from MSExchangeIS logged for each problem mailbox. The text of the event tells you the distinguished name of the problem mailbox and looks like this:

```
The mailbox for user /o=TonyR/ou=Exchange Administrative Group (FYDIBOHF23SPDLT)/
cn=Recipients/cn=TRedmond has been quarantined. Access to this mailbox will be
restricted to administrative logons for the next 6 hours.
```

A quarantined mailbox cannot be logged on to with Outlook or Outlook Web App, even by an administrator who has Full Access permission for the mailbox. In addition, Exchange ceases delivering messages to it. (The messages remain on the transport queues.) Apart from opening it with MFCMAPI, the only operation you can perform against a quarantined mailbox is to move it to another database. By default, Exchange maintains a mailbox in quarantine for up to six hours (21,600 seconds). You can change the quarantine period and the threshold for crashes by entering two new registry keys. These are specific to a mailbox database and are read when the database is mounted. The first key is in seconds and sets the quarantine period; the second sets the threshold for thread crashes Exchange will tolerate before considering a mailbox to contain some corrupt items.

- HKLM\System\CurrentControlSet\Services\MSExchangeIS\Server Name\Private-{Database GUID}\QuarantinedMailboxes}\MailboxQuarantineDurationInSeconds
- HKLM\System\CurrentControlSet\Services\MSExchangeIS\Server Name\Private-{Database GUID}\QuarantinedMailboxes}\MailboxQuarantineCrashThreshold

INSIDE OUT **A cmdlet for identifying a quarantined mailbox**

An administrator will probably not be aware of a quarantined mailbox until a user complains that he cannot access his mailbox, in which case, a check against the registry might reveal that the mailbox is in quarantine. The Get-MailboxStatistics cmdlet can be used to check for quarantined mailboxes. This command lists all mailboxes currently in quarantine:

```
Get-MailboxStatistics | Where {$_.IsQuarantine -eq $True}
```


You can access a quarantined mailbox with a utility such as MFCMAPI and use it to remove any corrupt items you detect. However, determining that an item is corrupt requires a deep knowledge of MAPI and how it expects data to be organized and is probably far past the capabilities of the typical Exchange administrator. To open the problematic mailbox with MFCMAPI, you need to:

1. Create a MAPI profile that points to the mailbox.
2. Run MFCMAPI, select Logon from the Session menu, and select the MAPI profile you created to force MFCMAPI to connect to the mailbox.
3. Select Default Store and then choose Open Default Message Store from the MDB menu. This forces MFCMAPI to display a dialog box to enable you to select the flags you want to use for the session. In this case, you want to use administrative privilege to open the mailbox, so it's sufficient to input OPENSTORE_USE_ADMIN_PRIVILEGE or 0x0000001 as shown in Figure 8-14.

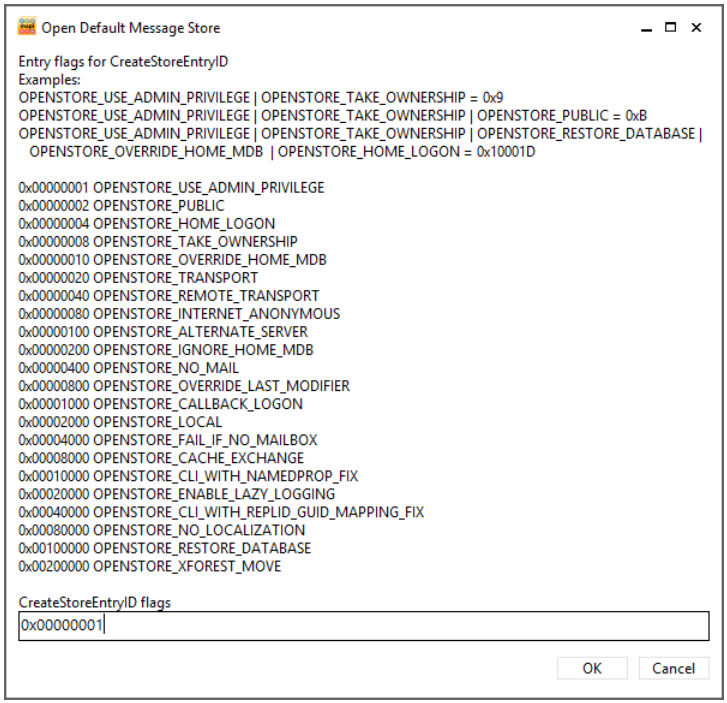


Figure 8-14 Specifying flags to use with an MFCMAPI session

4. Click OK to continue. MFCMAPI opens the folder tree so you can now navigate within the mailbox.

Even after gaining access to a mailbox with MFCMAPI, it will be difficult to find the items about which the Store is complaining. It's often a more pragmatic and therefore better solution to move quarantined mailboxes to a different database. Before you attempt to move a quarantined mailbox, you should run the Enable-QuarantineMailbox cmdlet to release it from quarantine. Because the condition that caused the mailbox to be quarantined probably still exists, you should not allow the user to access his mailbox after it is released. Instead, immediately move the mailbox either individually by running the New-MoveRequest cmdlet or in a group by including the mailbox in a migration batch (which could be a single-mailbox migration batch). To release the mailbox before the move, use:

```
Enable-QuarantineMailbox -Identity 'Tony Redmond'
```

Two good things can result from moving the mailbox. First, you can isolate the mailbox to a database where crashing will not affect service to other users. Second, the act of moving the mailbox can remove potential problems such as consolidating the named properties items in the mailbox use or dropping corrupt (bad) items as the Mailbox Replication service

(MRS) transfers items from the source to the target mailbox. See Chapter 10 for more information about how mailbox moves are performed.

The Store keeps the problem mailbox in quarantine until the quarantine duration elapses and then releases it by removing the registry keys for last crash time and crash count. The Store also removes the registry keys if they are more than two hours old when a database is mounted. The logic here is that dismounting and remounting the database might well have cleared any lurking issue that caused problems with Store threads. Of course, releasing a mailbox that contains corrupt data from quarantine could set off a new cycle of crashes, leading to further quarantine.

TROUBLESHOOTING

How to deal with a mailbox stuck in a quarantine cycle

If a mailbox goes into a cycle of constant quarantining, it's an indication that some deep problem exists that you might be able to resolve by moving the mailbox to another database. In this instance, you instruct Exchange to ignore any corrupt items it detects during the move operation so that the mailbox is as clean as possible after the move. If this doesn't work, and the mailbox continues to cause problems, the only route open to you is to export as much information as possible to a PST and then delete the mailbox. If the export operation fails because of corruption, you might be able to recover data through a database restore.

The Store periodically monitors mailboxes that have caused thread crashes. If no further crashes occur in a two-hour period, the Store concludes that whatever problem existed was transient and removes the mailbox from the list of quarantined mailboxes in the system registry. Obviously, an administrator can decide to remove a mailbox from the list, too, by manually deleting the entry in the system registry. This is not recommended unless you know that the underlying problem no longer exists or you want to restore client access to a mailbox to enable the user to recover as much data as she can from the mailbox before you delete and re-create the mailbox.

If you don't see the QuarantinedMailboxes entry under a database root in the registry, you know that the Store has never detected a problem mailbox for that database. Note that Exchange replicates mailbox quarantine data from the registry to other servers in a DAG through the failover cluster service to ensure that a failover retains knowledge of problem mailboxes.

Protection against high latency

Store databases can suffer from high latency for a number of reasons. A disk might be failing and responding to requests in seconds rather than milliseconds, or the server might be under extreme demand for CPU or suffering from memory shortage. Then it's possible that a mailbox could be causing an issue from a client-side bug or other problem.

Exchange includes a script, Troubleshoot-DatabaseLatency.ps1, which you can run if a server seems to be under strain to check whether it can determine the cause of excessive latency. For example, to check the latency of a database called DB1, the syntax of the script is:

```
Troubleshoot-DatabaseLatency.ps1 –MailboxDatabaseName "DB1" -LatencyThreshold 60
-TimeInServerThreshold 10 –Quarantine:$True –MonitoringContext:$True
```

The parameters that can be supplied to the script are as follows:

- *LatencyThreshold* is the threshold for latency and is 70 milliseconds by default. Normally, you won't need to amend this threshold unless you want to experiment to see what results are returned in your environment under different conditions or you are advised to do so by Microsoft PSS as part of a troubleshooting exercise.

- *TimeInServerThreshold* indicates the number of seconds of work per minute that can be performed on behalf of a single mailbox before the mailbox is considered hazardous to the health of the database. The number of seconds of work is measured by aggregating the time spent inside the Store (CPU, waiting for I/O and other operations) by all threads working on behalf of the mailbox over the period reported by the Get-StoreUsageStatistics command. The number of seconds of work per minute is calculated by dividing the aggregate number by the measurement period the Get-StoreUsageStatistics command uses (10 minutes). By default, the threshold is 60 seconds.

- *Quarantine* should be set to $True if you want mailboxes that are observed to be the cause of high server demand to be placed into quarantine.

- *MonitoringContext* determines whether the script is running on a server that is monitored by a system such as Microsoft System Center Operations Manager (SCOM). If this is the case, set the property to $True to force the script to write events to the Application Event Log.

If the problem is caused by a mailbox that occupies more than one CPU thread over a 10-minute period, Exchange puts the mailbox in quarantine and keeps it there for the default quarantine period unless it is moved to a different database. The same mechanism is used as for corrupt-item detection (see earlier) with the crash count set to three to force immediate quarantine. A mailbox is also put into quarantine if it peaks close to the 1 thread/10 minute threshold three times in a two-hour period.

Protection against excessive database or log growth

Many situations have been observed over the years when a Mailbox server seems to grow the size of a database excessively or generate a very large number of transaction logs. A database usually grows at a rate consistent with the growth of the mailboxes the database holds. The same is true of transaction logs, although you can see a spurt in transaction log generation with no increase in database size if a lot of white space is available in the database that can be filled by new transactions or if other activities, such as deleting a large number of mailboxes, cause a spike in transaction log growth.

For instance, if a database grows at the rate of 100 MB/month and suddenly expands by 1 GB overnight with the attendant increase in transaction logs, it might be due to legitimate activity or a software bug. Legitimate activity includes the movement of many mailboxes into a database or the import of data from PSTs. Known bugs in the past include the submission of messages by Outlook that go into a loop and continually generate new transactions. The problem therefore is to determine whether database growth is due to a problem or to administrative activity. The former might not go away and could need a patch to fix a bug; the latter will subside over time.

Microsoft provides a troubleshooter script called Troubleshoot-DatabaseSpace.ps1 that you can run against a Mailbox server that is currently experiencing excessive database or log growth. The script takes a reasonably hands-off approach to problems in that it doesn't rush to conclusions. Instead, it lets the potential problem go until available space on the disk that hosts the database or the transaction logs falls below a threshold at which it will be exhausted within a configurable time period at the current rate of growth. At this point, the script determines the mailboxes that are responsible for the growth and quarantines them to throttle back growth. If the current rate of growth will not exhaust available space within the period, Exchange logs an event that describes the potential for disk exhaustion due to database or log growth in the Application Event Log or, if SCOM is installed, a SCOM alert is raised.

The script is run as follows:

```
Troubleshoot-DatabaseSpace.ps1 –MailboxDatabaseName DB1
-PercentEdbFreeSpaceThreshold 29 -PercentLogFreeSpaceThreshold 20 –HourThreshold 6
-Quarantine:$True –MonitoringContext:$True
```

The parameters are the following:

- *PercentEdbFreeSpaceThreshold* sets the threshold for the percentage of free space available on the disk that holds the database. Exchange takes available white space in the database into account when it calculates how much free space exists on the disk.
- *PercentLogFreeSpaceThreshold* sets the threshold for the percentage of free space available on the disk that holds the database's transaction logs.

- *HourThreshold* sets the threshold for the number of hours of free space that must be available if the database continues to grow its size or transaction logs. In this example, the troubleshooter script responds (and quarantines mailboxes determined to be causing excessive growth) if disk space will be exhausted in six hours or less based on the current rate of consumption. If sufficient space exists to allow the database to grow for the specified period, the script logs an event and leaves it to the administrator to take the necessary action to restrict further growth.
- *Quarantine* determines whether the troubleshooter is allowed to quarantine mailboxes. If set to $False, the script will not attempt to quarantine problematic mailboxes.

Debugging swelling databases

It is possible for a database suddenly to begin to swell at an abnormal rate. For example, a database that normally grows at 50 MB daily that suddenly starts to expand by 2 GB daily without an obvious reason (such as a large-scale movement of mailboxes into the database) becomes a concern because user activity cannot explain the 40-times increase in growth. Another indication that heightens suspicions that all is not well is when a database reports a large proportion of white (unused) space. For example, if a database is 60 GB and the sum of the space occupied by mailboxes in the database is much lower than this amount, you should ask yourself why this might be. Again, moving many mailboxes out of a database creates white space as pages are freed up by the now-moved mailboxes, but a database that has more than 10 to 15 percent white space is suspect.

You can run the Get-MailboxDatabase cmdlet to check the size on disk and available white space for all databases on a server. Normally, to avoid any performance impact, a call to this cmdlet only reports the static information about a database that is held in Active Directory. In this case, you need to force EMS to make a call to the Store to retrieve internal database data, so pass the –Status parameter to retrieve the necessary data:

```
Get-MailboxDatabase –Server ExServer1 –Status | Format-Table Name, DatabaseSize,
AvailableNewMailboxSpace
```

Comparing the overall database size against the available white space is an easy way to determine whether any database has an unexpectedly high ratio of white space.

The white space within a database is reported as the available amount of space Exchange uses to hold new mailboxes. The same space is also used to expand existing mailboxes as users add new items. More white space will exist in databases immediately after mailboxes are moved to other databases, but this white space will be reused over time because the Store always prefers to use pages that exist inside a database rather than extending the database on disk.

The Exchange 2013 Store includes routines to take action, such as quarantining suspect mailboxes, when problems are detected. However, it's always possible for a new software bug to be exposed through interaction with a particular type of item stored in a mailbox or through the way a client attempts to interact with an item and, in turn, the bug might result in an unexpected growth in the database. One way to deal with quarantined mailboxes is to move them to another database to see whether MRS detects any bad items that are suppressed en route. Even if no bad items are found, the rebuilding of the mailbox in a different database might be sufficient to resolve the problem.

Experience shows that calendar items often are the cause of a swelling database, perhaps because many clients interact with user calendars, using different protocols. You can run the Calendar Checking Tool for Outlook (Calcheck), also downloadable from the Microsoft website, to validate that the calendar items in a suspect mailbox are properly formulated and intact.

Finally, if moving the mailbox or Calcheck cannot cure the problem, you can resort to the examination of transaction logs in an attempt to identify any patterns that show you where the problem lies (for example, the subject of a message). There is no officially supported method to examine Exchange transaction logs because these files are intended to be used for data replication and recovery purposes only. However, an interesting blog from a senior Microsoft support engineer at *http://blogs.msdn.com/b/scottos/archive/2007/07/12/rough-and-tough-guide-to-identifying-patterns-in-ese-transaction-log-files.aspx* explains how you can use a set of utilities to examine transaction logs. No guarantee is given that this approach will turn up any clues, and this is not an activity you should undertake without some planning and preparation. For instance, don't just start reviewing the live transaction logs used to replicate databases within a DAG. Always take copies of logs to work with, preferably from a time when some database swelling was observed, and use them rather than live data.

At the end of the day, none of these methods might turn up any results if you are dealing with a new and previously unknown bug. However, these methods have been successful in the past in identifying problematic client activity caused by corrupt data, so they are worth considering if you have to deal with a swelling database.

Online repair cmdlets

Those who have worked with Exchange for many years know of ISINTEG, the Information Store Integrity maintenance utility Microsoft provided to address problems that occurred in the logical structure of Exchange databases. (By comparison, ESEUTIL handles much lower-level problems that cause physical corruption.) ISINTEG is no more because its function in repairing logical corruptions is now performed through a set of online repair cmdlets.

Prior to Exchange 2010, the database schema featured many tables maintained at database level. A corruption that occurred in a table can therefore affect all the mailboxes in a database. The current schema is more focused on mailbox tables, so the chance of a logical corruption within a database that affects the entire database is remote. Although the likelihood of corruptions is greatly lowered by the new schema, it's also true that new mailbox-level corruptions are possible. However, because most data are now stored in mailbox tables, the simple act of moving a mailbox from one database to another is sufficient to clean out many problems that might exist in these tables. Bad items are dropped en route from the source to target database, and problems that might exist in views and item counts are fixed as the new mailbox is built in the target database.

The mailbox repair cmdlets enable administrators to create online repair requests for mailboxes that address the most common causes of corruption for views and item counts. These are the following:

- Search folder corruptions
- Incorrect aggregate counts on folders
- Incorrect contents returned by folder views
- Incorrect link to a parent folder

The repair cmdlets use roughly the same model as mailbox move, import, and export requests in that an administrator creates a repair request that the Store queues for processing, which then performs whatever repairs are required asynchronously with the database online, so there's no need for the user to log out of her mailbox while the Store examines and adjusts internal mailbox structures. The New-MailboxRepairRequest cmdlet creates a repair request for a mailbox. For example:

```
New-MailboxRepairRequest –Mailbox JSmith –CorruptionType FolderView
```

If you add the –DetectOnly parameter to the request, Exchange reports any corruption it finds but doesn't repair it. The other corruption types that can be fixed in a mailbox are SearchFolder, AggregateCounts, and ProvisionedFolder. These repairs fix problems with search folders, counts on folders, and provisioned fields. You can perform several repairs at one time by specifying a list of the fixes you want to make. For example:

```
New-MailboxRepairRequest –Mailbox JSmith –CorruptionType FolderView,
AggregrateCounts, ProvisionedFolder
```

By default, the Store only repairs the primary mailbox. If you want to include the mailbox's personal archive in the repair, you must specify the –Archive parameter:

```
New-MailboxRepairRequest –Mailbox JSmith –CorruptionType FolderView, AggregrateCounts
-Archive
```

You can also scan all the mailboxes in a database at one time to fix any corruption that is found in any mailbox. For example:

```
New-MailboxRepairRequest -Database DB1 -CorruptionType FolderView, SearchFolder,
AggregateCounts
```

When you submit a new repair request, Exchange responds with a task identifier and the name of the server that will handle the request. This is the Mailbox server that currently hosts the active copy of the database. The only evidence that Exchange has processed the repair request exists in the Event Log, which captures an event (Figure 8-15) when a mailbox repair request is completed. Events are logged on the server that processes the repair request.

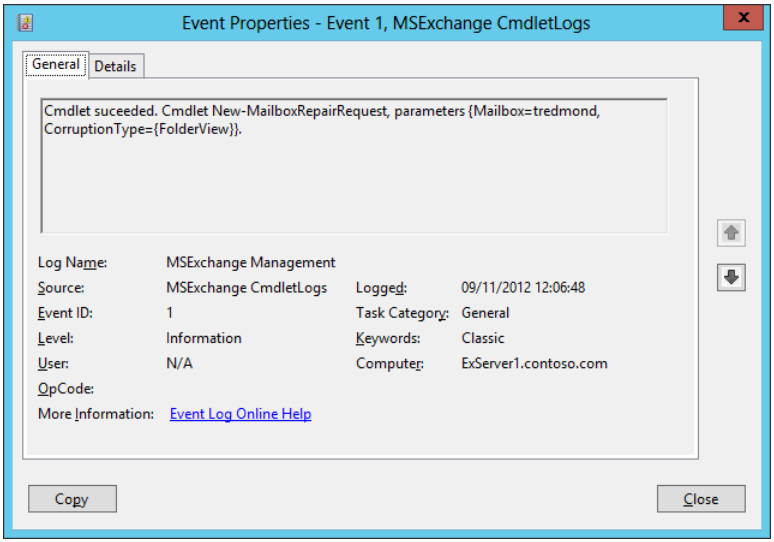


Figure 8-15 Viewing details of a mailbox repair request logged in the event log

INSIDE OUT Canceling a repair job

You can't cancel or review the current status of a repair job. This functionality might be added by Microsoft in a future release. For now, the only way to terminate a repair job is to dismount a database or move the database to another server (or if the database crashes due to a software bug). These actions clear out any pending or active repair jobs within the database.

To ensure that performance is not affected, you can only run a single repair against a complete database on a server at one time. However, you can run up to 100 individual mailbox repairs concurrently on a server (spread across multiple databases).

If the database has copies within a DAG, the results of any repairs made to fix problems found in the tables within the mailbox are replicated along with other transactions to the database copies and are logged as events in the Application Event Log on the server on which the repair is performed.

Rebuilding a database

As you know, Exchange performs ongoing online defragmentation to achieve maximum page contiguity and space usage within the database. The overall size of the database is not affected by online defragmentation because pages freed by processing are recycled and made available to the Store to hold new items. Offline defragmentation or a database rebuild is accomplished by taking the database offline and then running the ESEUTIL utility in /D mode to build a new copy of the database by reading all the pages out of the old database and writing them into the new. Depending on system performance and the size of the database, an offline rebuild can take a long time, and the only thing it accomplishes is the return of unused space to the file system. In the days of Exchange 5.5 and Exchange 2000, when background defragmentation was inefficient and ineffective, it was deemed best practice to perform an offline rebuild for databases on a regular basis to prevent them from swelling to obscene sizes.

Microsoft has improved the effectiveness of background defragmentation significantly, and you should never have to take a database offline to rebuild it except in extraordinary circumstances. For example, mailboxes allocated to students while they attend universities are more transient than mailboxes used in corporate email systems because there is typically a very large turnover of the student body annually. If you had to delete 10,000 student mailboxes from a database, performing a rebuild would shrink the database dramatically and make backups faster. However, given this scenario, it would be better to create a new database, move all the remaining mailboxes to the new database, and then delete the old database. Users aren't disturbed by this operation because the Mailbox Replication service (MRS) moves their mailboxes in the background.

A rebuild might also be required in previous versions of Exchange when the window assigned for background maintenance operation is insufficient for defragmentation to complete, so over time, the database bloats with unused pages. However, given that Exchange performs online defragmentation on an ongoing basis, this argument to justify an offline rebuild does not hold water now.

Unlike previous versions of Exchange, you won't see events logged to indicate when background maintenance has started and finished online defragmentation, and you will not see

any reports of the white space that's available in a database following a defragmentation pass. This is quite logical because the continuous nature of Exchange online defragmentation means that it starts when the Information Store mounts a database and finishes when the Store service shuts down. The loss of events to log the start and finish of defragmentation does not mean that you cannot see evidence that defragmentation is proceeding. Microsoft has added a new set of counters to the system Performance Monitor, so you can start the PerfMon utility and add the MSExchange Database/Instances/DB Maintenance IO Reads/sec counter to the set being monitored. The level of activity varies with system load, and you should see more activity when the system is lightly loaded and less when users are active.

As noted in the "Debugging swelling databases" section earlier in this chapter, you can still gain a sense of the amount of available white space in a database with the Get-MailboxDatabase cmdlet. You might decide to check the databases in use within the organization every six months or so to decide whether any are candidates to be taken offline for a rebuild. In reality, given the efficiency of Exchange databases today, it is unlikely that you will discover anything that requires intervention, so this is something that is done for peace of mind rather than an imperative operational requirement. The command shown below can take some time to complete because it requires EMS to check the status for each database on every server. The data is sorted by database size.

```
Get-MailboxDatabase -Status | Sort-Object DatabaseSize -Descending | Format-Table
Name, DatabaseSize, AvailableNewMailboxSpace
```

```
Name            DatabaseSize                      AvailableNewMailboxSpace
-------         ----------------                  -------------------------
MBDatabase3     1.57 GB (1,686,175,744 bytes)     121.1 MB (126,943,232 bytes)
MBDatabase1     1.195 GB (1,283,522,560 bytes)    69.44 MB (72,810,496 bytes)
IT Department   936.1 MB (981,532,672 bytes)      6.063 MB (6,356,992 bytes)
```

The reported database size should be pretty close to the actual file size on disk. The figure for available new mailbox space gives an indication of the available white space in the database and, as you can see, the percentage of free space varies from database to database, depending on recent activity in the database. For example, a database that hosts a set of mailboxes that has remained constant over time will have a lower amount of free space than a database that has had some mailbox moved out to another database. This is obvious when you compare the data reported for the IT Department (0.64 percent free) and MBDatabase2 (85.3 percent free) databases. Over time, the percentages for the two databases will come much closer together as user activity consumes space. If more than 15 percent of space is available in a database, it might be due to incomplete maintenance or another reason such as the recent deletion of many mailboxes.

The available white space reported by Get-MailboxDatabase only tells you how much free space is available in the root tree of the database. It does not include any free space that is available within mailbox or index tables that might be recovered and eventually returned to the root tree by database defragmentation. To get a 100 percent accurate picture of how much white space actually exists within a database, you have to dismount the database, run ESEUTIL /MS, and then calculate the white space by multiplying the number of free pages by 32 KB (because Exchange 2013 uses a 32 KB page size). The article posted at *http://technet.microsoft.com/en-us/library/aa996139(v=EXCHG.65).aspx* explains how to run ESEUTIL /MS, but remember that this was written in 2006 when Exchange 2003 used a 4 KB page size. Most people aren't interested in taking databases offline simply to find out how much free white space exists and are happy to use the data Get-MailboxDatabase reports instead.

The available new mailbox space value is a fair approximation

The value reported for available new mailbox space is not an accurate indication of available white space because it includes only pages that are immediately available to be assigned to new mailboxes, which use space from the root tree within the database. It does not include free pages that exist in mailbox or index tables. However, it's a good approximation of how much white space exists in the database.

Moving away from internal structures, at the level of the database file, Exchange uses a 128-MB default file extension whenever it requests more disk space from Windows to grow an active or passive database.

Using ESEUTIL

ESEUTIL is a useful but dangerous utility. It's useful because it can fix some low-level problems in a database. (However, it cannot fix fundamental corruption caused by hardware failures.) It's dangerous because some administrators and other commentators consider it to be a good thing to run ESEUTIL against Exchange databases on a regular basis just in case. Such an attitude is firmly entrenched in the 1990s when, to be blunt, the Exchange database wasn't as robust and reliable as it is today—and has been since Exchange 2003.

In the early days, the predominant reason for running ESEUTIL was to recover disk space from the database. Exchange was like a swollen pig when it came to growing its database and wasn't very good at reusing database pages. A 10-GB database (big in those days) might only contain 7 GB of mailbox data; you had to rebuild the database with ESEUTIL to return the unused 3 GB storage to the system. Of course, disks were expensive and small, so

recovering 3 GB was a big thing, making the exercise usually worthwhile, even if you had to take the database offline over the weekend to perform the rebuild.

It's a different situation today. Disks are larger, they are radically less expensive, and Exchange is much better at recovering and reusing pages within its databases. It's true that software bugs occasionally cause an excessive growth in the size of a database, but the software bugs are being closed off one by one, and additional safeguards are being added to Exchange to prevent a database from suddenly swelling.

I don't cover the correct use of ESEUTIL in detail in this book. The utility has been around since the earliest days of Exchange and is well covered in other books, blogs, and TechNet. Instead, I refer to ESEUTIL when it is needed, such as validating a recovery database before it is mounted.

INSIDE OUT **Don't use ESEUTIL to rebuild a database**

You should not have to rebuild a database with ESEUTIL unless Microsoft Support advises you to do so. If anyone else tells you that running ESEUTIL to rebuild databases regularly is good for Exchange, you should give him the same look you'd give someone who told you to have a monthly colonic irrigation to improve your complexion.

Those considering using ESEUTIL to rebuild a database just to return some disk space to Windows need to remember that a database rebuild invalidates the current set of its transaction logs. Essentially, the rebuild transforms the internal structure of the database, so any of the transactions in the logs cannot be applied if the need occurs to recover transactions through log replay. This situation is not a problem on a standard server because you can make a full backup immediately after the rebuild to establish a new secure baseline for the database. It's different when a DAG is involved because the transaction logs provide the replication mechanism for the database copies; if you rebuild an active database, you essentially reset the database, and you will have to reseed all the database copies after the rebuild is complete. Not recommended!

Database usage statistics

The Get-StoreUsageStatistics cmdlet is designed to be a diagnostic aid for administrators who are concerned about Store performance for a particular mailbox, database, or server. The cmdlet retrieves data about Store activity for an individual mailbox or for all the mailboxes in a database or server to report the amount of server time executed for Store operations over the past 10 minutes. The cmdlet analyzes the mailboxes with the highest

TimeInServer value over the past 10 minutes, using performance data sampled every second.

The following example retrieves data for the activity in a specific database. In this instance, you see that the time of highest demand was during the "3" sample (three minutes ago) because activity for three mailboxes is noted during this time. The TimeInServer field provides a relative measurement of the demand that a mailbox exerts on the server. It is calculated from the total time spent processing synchronous and asynchronous requests sent to the Store for a mailbox, so it is a pretty good catchall metric that loosely encapsulates the performance indicated by the other metrics.

For example, if the activity for a mailbox is I/O intensive, the latencies for that mailbox tend to be higher, and this increases the TimeInServer figure. Heavy demand for the CPU also increases the TimeInServer figure because the Store has to spend more time processing requests on behalf of the mailbox. By looking at the following data, you can also see that the user who generated the most activity was the EMEA Help Desk in the fourth sample; the TimeInServer rating of 1163 was over twice as high as any other mailbox during the 10-minute duration of the sample.

```
Get-StoreUsageStatistics -Database 'VIP'
```

```
DigestCategory       SampleId       DisplayName                    TimeInServer
------------------   ---------      ------------                   -------------
TimeInServer          0             Mailbox - Akers,Kim            485
TimeInServer          1             Mailbox - Smith, John ...      32
TimeInServer          1             Mailbox - Online Archi...      0
TimeInServer          2             Mailbox - EMEA Help Desk.      1163
TimeInServer          3             Mailbox - Ruth, Andy           16
```

To access statistics for all the databases on a server, you use a command like this:

```
Get-StoreUsageStatistics -Server 'ExServer1'
```

Note that Exchange returns statistics only for mounted databases. Database copies are noted, but you won't see any statistics. This is quite logical because no mailboxes can connect to these database copies. An individual mailbox can appear multiple times in the output for a database or a server. To drill down on the activity for a mailbox, you can retrieve more detailed statistics for a specific mailbox by passing its name to the cmdlet.

```
Get-StoreUsageStatistics -Identity 'Akers, Kim'
```

```
DigestCategory    : TimeInServer
SampleId          : 1
SampleTime        : 5/23/2010 2:39:10 AM
DisplayName       : Mailbox - Akers, Kim
```

```
TimeInServer      : 424
TimeInCPU         : 79
ROPCount          : 222
PageRead          : 0
PagePreread       : 0
LogRecordCount    : 975
LogRecordBytes    : 467303
LdapReads         : 1
LdapSearches      : 0
ServerName        : ExServer1
DatabaseName      : VIP Data
```


Apart from the TimeInServer metric, the data includes:

- **TimeInCPU** The number of milliseconds of CPU time used for operations for this mailbox. Samples are taken on a one-minute basis; if the server has multiple processors, this number can exceed 60 seconds. The timer is very simple and is triggered whenever the CPU dedicates time to the thread handling operations for the mailbox. This information is useful when tracking down problematic clients that are imposing a high load on the server.
- **ROPCount** The number of remote (client) operations performed on the mailbox during the sample period.
- **PageRead** The number of noncached page reads required by the mailbox.
- **PagePreRead** The number of pre-read pages required by the mailbox.
- **LogRecordCount** The number of log records written out for this mailbox during the sample period. Together with TimeInServer, this is a good indication of the amount of activity the Store performs to handle mailbox operations.
- **LogRecordBytes** LogRecordCount converted to bytes.
- **LDAPReads** The number of asynchronous LDAP reads required by mailbox operations.
- **LDAPSearches** The number of LDAP searches performed on behalf of the mailbox.

The measurement data are cleared out after 10 minutes. This means that Get-StoreUsageStatistics returns blank results if you look for data 10 minutes after the last mailbox activity was recorded. If fewer than 25 users have been active during the past 10 minutes, only these users will be listed in the summary.

Mailbox assistants

Exchange includes a number of mailbox assistants to perform automated processing on different aspects of the data held in mailboxes. An assistant is just another name for a thread that executes within the Exchange System Attendant process on a Mailbox server. Exchange 2013 uses two types of assistants:

- **Event-based assistants** These respond to events as they occur. For example, the resource booking assistant monitors requests to book rooms and lets users know whether the room is unavailable.
- **Throttle-based assistants** These operate all the time to process data and are subject to throttling to ensure that their workload does not affect the responsiveness of a server. The Managed Folder Assistant is the best known of these assistants.

Among the better-known assistants are the following:

- **Calendar assistant** Checks mailboxes for incoming meeting requests and processes the requests according to mailbox settings
- **Resource booking assistant** Processes meeting requests for room and equipment mailboxes
- **Scheduling assistant** Scans attendee calendars for suitable meeting slots and suggests them to meeting organizers
- **Junk email options assistant** Copies the safelist data from user mailboxes to their Active Directory accounts so the anti-spam agents can use it
- **Managed folder assistant** Processes retention policies for managed folders and applies retention policy tags to items in mailboxes

These assistants are relatively well known because they exist in previous versions of Exchange.Exchange 2013 includes a new test cmdlet to enable administrators to check that the mailbox assistants are functioning properly on a server and to recover from any problem the test cmdlet detects. For example, this command runs the cmdlet to test that the mailbox assistants are functioning properly on server Exserver1 and then reports the results in list format:

```
Test-AssistantHealth -Server ExServer2 -ResolveProblems | Format-List
```

Exchange uses a work cycle model to define how its assistants perform work while remaining active in the background to accomplish goals over a set period. For example, you might define that the Managed Folder Assistant must process every mailbox on a server at least once every two days. Given the constraints determined in a work cycle, the assistant will

work out how much work it has to do over the period and create an internal schedule to do the work on a phased basis so that a smooth and predictable demand is generated on the server rather than the peaks created by the older model. The assistant monitors system conditions on an ongoing basis to make any required adjustments to perform well. If the server is under high demand for a period due to user activity, the assistant can back off its work and then speed up when server load drops.

The Get-MailboxServer and Set-MailboxServer cmdlets retrieve and set work cycle information for assistant processes. First, find out what the current work cycles are for the assistants that run on a multirole server:

```
Get-MailboxServer –Identity ExServer2 | Format-List *WorkCycle*
```

The majority of the assistant processes process the data for which they are responsible at least once daily because the work cycle period is set to one day. In effect, the goal for the assistant is to process all relevant data for all mailboxes on the server in a 24-hour period, assuming that system resources allow it and the assistants are not throttled back to allow more important processes to proceed. The checkpoint for an assistant tells it how often it should check for new objects that need to be processed, such as looking to see whether new mailboxes have been added to the databases on a server. In most cases, the checkpoint is equivalent to the work cycle period. It makes sense to have the same work cycle settings applied to all Mailbox servers in the organization. This is easily done by running the Get-MailboxServer command to fetch a collection of all Mailbox servers and then using Set-MailboxServer to set the properties.

And now for something completely different

Much of the technology discussed so far has existed in one form or another for at least several versions of Exchange. The Store is powered by a database engine, and its physical instantiation is a set of databases on disk. The interesting subject is to discuss just what can be done with those databases using the high-availability technology included in Exchange 2013, so that is where this book goes next.

CHAPTER 9

The Database Availability Group

Changes in high availability in Exchange 2013 457
Database portability. 459
Migrating an Exchange 2010 DAG. 464
Active Manager. 465
Transaction log replay: The foundation for DAG replication 479
Changes in message submission within a DAG 489
Day-to-day DAG management and operations. 489
Datacenter Activation Coordination 551
Approaching DAG designs 552
Server maintenance 562
Stressed servers. 565
On to protecting data 566

For most companies, the email system is one of the mission-critical applications that everyone immediately notices if the service misses a heartbeat. Microsoft has invested heavily over the years to provide Exchange Server with the ability to resist different types of failure and deliver a highly available service. The investment has spanned all parts of the Exchange code base, but perhaps the biggest single technical step made in Exchange 2010 was the introduction of the Database Availability Group (DAG) to enable customers to build highly available Exchange infrastructures without having to invest in expensive, third-party, add-on products. Having a solid set of high-availability features built into the product and administered through the standard management interfaces increases the attractiveness of Exchange as a platform, removes complexity, and avoids cost for customers in the small to medium segment and for many enterprise customers.

The DAG enables you to deploy highly available servers in an incremental fashion so you can deploy standalone Exchange Mailbox servers first and then decide to include those servers in a DAG as the need arises to incorporate more high availability into the environment. You can also gradually expand the DAG to include more servers or more database copies to add resilience against different failure scenarios as time, money, and hardware allow.

Changes in high availability in Exchange 2013

Many Database Availability Groups have been deployed to date. Inevitably, as is the case after new technology is introduced, some problems were encountered. Some of these problems were due to a lack of experience with the technology, whereas the remainder pointed to some areas that Microsoft could improve in Exchange 2013. These include the following:

- **I/O improvements** Although on a practical level it is impossible to say whether a customer deployment can ever achieve the degree of improvement that Microsoft

claims, it is fair to say that the storage characteristics of Exchange 2013 have continued to improve the I/O characteristics of the product and that Exchange 2013 can be deployed on all classes of available drives, including low-cost SATA. Care must still be taken to ensure that a rush to embrace low-cost storage does not compromise data integrity, but it is undisputable that storage is much cheaper than ever before, so it is possible to deploy more database copies than ever before and achieve data resilience in that manner—the entire purpose behind the DAG.

- **Autoreseed** Exchange 2013 enables you to configure sets of volumes to support automatic reseeding of failed databases. The Active Manager periodically scans for failed copies, and, if such a copy is detected, the Active Manager can allocate space, reseed the failed database(s) from the active copy, and then bring the newly seeded copy online. Previously, this work would have required manual intervention to detect the problem, fix the failed disk, and reseed the copy. You had to configure spare disks and still fix the failed disk!

- **Multiple databases now supported on a single volume** With the further reduction of I/O operations generated by database activity, it is now possible for Exchange 2013 to support multiple databases on a single volume. In the past, the advice was to place databases and their transaction logs on different volumes (Exchange 2007) or to keep a database and its transaction logs on one volume (Exchange 2010). The changing advice reflects improvements in software, storage trends, and experience. However, as in any issue having to do with hardware, you have to put these trends into the context of the operational environment within which you work and decide how you want to deploy databases, often after considering operational backups (for instance, whether hardware-based Volume Shadow Copy Services [VSS] backups are used) or following advice from your storage vendor.

- **Automatic network management** DAGs depend on network settings to know how to exchange information between member servers. Although Exchange 2010 did a good job of setting up DAG network information automatically to take account of the network interface cards (NICs) installed on servers, in multi-subnet configurations the automatic enumeration of networks often resulted in a profusion of defined networks within the DAG, some of which were never used. In turn, this required administrators to intervene manually to edit the network settings and collapse (remove) unwanted networks. Using the experience gathered from multiple real-life deployments, Exchange 2013 is more intelligent when it learns about network conditions from installed NICs and then transforms this information into settings used for the DAG; you should not have to intervene to retune DAG networks in multi-subnet configurations. It's still possible for administrators to exert manual control over DAG network settings, but it's usually best to let Exchange do the work for you.

- **Replacement for the Transport Dumpster** The Transport Dumpster served as a method to capture copies of messages sent to users on replicated databases as they passed through the transport system so that messages could be recovered if required to bring a database completely up to date during a recovery process. Exchange 2013 uses Safety Net, a replacement for the Transport Dumpster. Both mechanisms hold messages in a special queue within the transport database, with a separate queue maintained for every active mailbox database on a Mailbox server. You can control how long Safety Net holds copies of messages before the copies expire and are automatically removed. Within a DAG, the Microsoft Exchange Replication service can use the copies of messages held in Safety Net to redeliver messages to mailboxes if it needs to update a database during a recovery process. This approach extends the shadow redundancy mechanism introduced in Exchange 2010, which continues to capture copies of messages in transit. You do not need to run a DAG to use Safety Net; Exchange 2013 automatically stores copies of delivered messages on other Mailbox servers within the same Active Directory site. See Chapter 2, "The Exchange transport system," in *Microsoft Exchange Server 2013 Inside Out: Connectivity, Clients, and UM* for more information on how the Safety Net feature works.

With this overview of some of the changes made in the area of high availability for Exchange 2013, this chapter explores the details of how DAGs work.

Database portability

Exchange Server 2007 introduced the concept of database portability. You can mount a mailbox database from one server on another server. Exchange 2010 and Exchange 2013 both support database mobility. The major difference between portability and mobility is that you are not moving a database from server to server. Instead, you move the active focus for client connections and workload between copies of a database. All the copies of a database share the same globally unique identifier (GUID) or identity, so each can function as the active master no matter on which server it is currently located. Database copies also must have the same path on the server for database and log files. The ability to move databases around servers within a DAG is fundamental to the ability to manage database-level failures to achieve high availability within Exchange, without recourse to third-party software as previously required.

Exchange 2010 and Exchange 2013 treat a mailbox database as a unit of failover in that it can be moved between servers in a DAG as problems occur. In addition, databases are now a top-level object within the organization and treated with the same importance as a server in terms of the overall capabilities of the organization. A server now exists to host databases, whereas before the DAG appeared, databases were subservient to servers, and each database belonged to a server.

Of course, you do not need to define DAGs within your organization, and you can run Exchange as before with databases that never move off their host server. If you elect to deploy a DAG, servers become members of the DAG and can host mailbox databases that belong to the DAG. All the services that run on a Mailbox server (the Exchange Information Store service and so on) still exist and operate on the mailbox databases that a server currently hosts, including running the Replication service to process incoming copies of transaction logs to update passive copies of mailbox databases whose active copies exist on other servers elsewhere in the DAG.

INSIDE OUT The advantages of multiple passive database copies

Only one active copy of a database exists at a given time, but you can create multiple passive copies up to the number of available servers in the DAG. Assuming that the design of the DAG is in line with best practice, the more copies that exist for a database, the more likely it is that you can quickly recover from an outage that affects a server or some storage attached to a server, and the less likely it is that your infrastructure has a single point of failure. Lower disk I/O and the ability to use cheaper disk technology to host databases mean that you can afford to maintain more database copies. Of course, you have to maintain a reasonable balance here to ensure that you don't create more database copies than you really require.

For example, in a DAG that spans 10 servers, a database can be active on one server, and you can have its contents replicated to passive copies that are managed on the nine other servers in the DAG. This is a rather extreme example, and it's more likely that databases will have three passive copies to achieve a good balance between the ability to recover from different outage scenarios and the amount of data replication that is required to keep the passive copies updated. When the DAG stretches across multiple data centers or a need exists for a lagged copy, the number of passive copies might be increased to four. The point is that system designers now have great flexibility in terms of the way they protect data in different circumstances.

As you would expect, the active copy of a database can be mounted or dismounted. If mounted, the database is generating transactions that Exchange replicates to the target servers that host copies of the database. Nothing much happens for dismounted databases.

Apart from its mounted state, Exchange defines a database to be either the source for replication or the target for replication. A database copy can act as the source or target but cannot function as both at the same time. In much the same manner, a database copy can be active, meaning that it is available to service incoming connections from email clients, or

passive, meaning that it can be switched into active mode to take over service, but it cannot be both active and passive at the same time. Only one copy of a database can be active within the DAG at any time, and a server cannot host more than one copy of a database. All this is quite logical and provides the framework within which replication and database transition from active to passive and back again occurs.

Database Availability Group basics

Fundamentally, a DAG is a collection of databases and copies that are shared across up to 16 servers. A server can host only one copy of a database. Although 16 might seem an arbitrary figure to use as the limit for the number of servers a DAG can support, in fact, the limit was imposed by Windows Failover Clustering, which can support only 16 nodes in its clusters when running on Microsoft Windows 2008 R2 servers. Windows Failover Clustering supports 64-node clusters in Windows Server 2012, but Exchange 2013 respects the lower limit because it can be deployed on both Windows 2008 R2 SP1 and Windows 2012. In addition, it can be argued that no great demand exists for growing DAGs to larger than 16 servers; such a mega-DAG might be very difficult to manage, and current DAGs are capable of supporting 100,000 mailboxes, providing that appropriate hardware and designs are used.

In any case, 16 seems like a number that should be sufficient for most deployments and is certainly enough to meet the needs of most deployments. If you need to support more than 16 servers, you can just split them across multiple DAGs. A combination of technologies constitutes a DAG, including the following:

- Log replication (the technology to implement transfer and replay plus the network load to support replication)
- Networks
- Windows Failover Clustering
- Monitoring and management tools
- Server and storage hardware

The DAG implements the concept of an active (or primary) database—the one to which users currently connect—and its copies on other servers that can be swapped into place to become the active database. The database copies are kept updated through log replication and replay. If a problem occurs on a server that renders the databases running on the server inaccessible, the DAG can activate a passive copy and make it the active copy. By reference to Active Manager, which is responsible for controlling the activation of database copies, the Client Access Server (CAS) can redirect client connections seamlessly to the newly activated copy.

DAG member servers can be multirole servers, but each DAG member must have the Mailbox role installed because it has to be able to host a mailbox database. The servers in a DAG can be on different subnets and span different Active Directory sites as long as the underlying network infrastructure supports sufficient bandwidth to transfer the expected volume of transaction log files and content indexing traffic between the different servers. Equally, a stretched DAG can exist within a single Active Directory site, so a great deal of flexibility exists in how you lay out DAGs within an existing Active Directory forest.

To ensure smooth operation, Microsoft recommends that Mailbox servers in a DAG be connected with a network that accommodates a round-trip latency of 500 milliseconds or less. The recommendation has to be placed in context. If you run a small DAG that generates a few hundred transaction logs daily, the network requirements are far less stringent than when operating a large DAG in which thousands of logs are generated hourly. In the first case, a 500-millisecond latency will cope with the load without any difficulty whatsoever. In the second, a round-trip latency of 250 milliseconds will struggle under the load, and a higher capacity and more capable network will probably be required.

Microsoft also recommends that you block cross-network traffic between the data centers to avoid excessive heartbeat traffic across the cluster and so conserve available bandwidth for more important activities such as log replication. There are other issues to consider when a DAG stretches across two data centers, including the assignment of IP addresses for the networks, the use of appropriate Domain Name System (DNS) time to live (TTL) settings to ensure that clients pick up network changes quickly in the event of switchovers, and the provision of suitable names for all the services Exchange offers to clients from both data centers.

Tip

An in-depth discussion of these issues is impossible in the context of a book that attempts to cover the wide horizon Exchange presents. If you are considering a stretched DAG, you should sit down with an Exchange consultant who has considerable experience with DAGs to chart out the current situation, your business needs, and a suitable design for the DAG.

The original versions of Exchange 2013 restricted the Enterprise version to mounting a maximum of 50 databases (including databases that are active and servicing clients and those that are passive and processing replicated transactions from active databases). This was a reduction from the previous 100-database limit available in Exchange 2010. Experience with actual deployments indicated that few customers actually ran DAG member servers with more than 50 databases. This fact, together with the greater overhead

imposed by the additional processing Exchange 2013 Mailbox servers perform and some extra overhead imposed by the transition to Search Foundation, convinced the Exchange developers that it was best to impose a maximum of 50 mounted databases.

Each database mounted on a server incurs an overhead as the database structures are mapped into memory. Remember that the Exchange developers have long been in the business of trading memory for I/O to drive down the overall I/O profile of the server and enable the use of lower-cost storage solutions. So if memory is being assigned to new processes, it is reasonable to conclude that less memory is available to mount databases, and that's exactly the position reached in Exchange 2013. The performance data within Microsoft reveal that Mailbox servers have no problems in terms of I/O, but they can become memory-constrained and CPU-bound as the number of mounted databases grows.

Customer feedback often exerts a powerful influence on engineering groups. Those who ran large DAGs did not welcome the reduction because this limited a DAG to 800 databases spread across the 16 available members and therefore posed some interesting problems when the time came to upgrade any Exchange 2010 DAGs that supported more than 800 databases to Exchange 2013. Fortunately, Microsoft was able to do the work necessary to improve memory management and therefore restored the maximum number of mounted databases to 100 from Exchange 2013 RTM CU2 onward.

The dependency on Windows clustering

In the background, the DAG uses Windows Failover Clustering technology to manage server membership within the DAG, to monitor server heartbeats to know which servers in the DAG are communicating within the cluster, and to maintain a quorum. Exchange has no practical dependency on the network name for the DAG (it is used for internal security purposes) because Exchange reverts to server names if the cluster name is unavailable. However, a third-party product might have a dependency on the DAG name to function properly, and you should understand when this is the case.

In fact, you never need to manage cluster resources such as nodes, network, or storage by using the Windows Failover Cluster Manager. Everything is managed through Exchange, and unless you have very good reason to manipulate the cluster through the Failover Cluster Manager and are willing to manage the DAG on that basis, you might change the cluster settings with the Failover Cluster Manager and run the risk of breaking something on which Exchange depends. In effect, Exchange provides a blanket that hides the complexity of cluster technology from system administrators.

INSIDE OUT **Operating system requirements**

Even though Exchange makes minimum use of cluster technology, the dependency on Windows clustering means that you can only add Mailbox servers to a DAG if they are running on Microsoft Windows Server 2008 R2 SP1 Enterprise edition or Windows 2012 Standard or Datacenter edition. It also means that all the DAG member servers must be part of the same domain. You should also run the same version of the operating system on all the DAG member servers; you definitely cannot mix Windows Server 2008 R2 SP1 and Windows Server 2012, and it just makes good sense to keep the servers at the same software level.

Despite its name, apart from telling Exchange that a cluster node is unavailable, Windows Failover Clustering takes no part in the failover of Exchange mailbox databases. This functionality is provided by a system management component within Exchange called the Active Manager, which maintains visibility of server conditions and the current state of databases and is responsible for instructing servers to move database copies from active to passive and passive to active as required. Even better, an Exchange administrator doesn't have to be concerned with the complexities of Windows clustering because Exchange configures the limited clustering features it needs (cluster heartbeat and quorum) when it adds the first Mailbox server to a DAG. Some of the information relating to the DAG is held in Active Directory. This information, such as the name of the DAG, doesn't change very often. Other information that is more dynamic and prone to change quickly, such as database mount status (active or passive), is held in the cluster database.

Migrating an Exchange 2010 DAG

You have a DAG running Exchange 2010, and now you want to introduce Exchange 2013 and upgrade the DAG so that it runs Exchange 2013. The question is how best to accomplish the task. The answer is pretty simple—you cannot run mixed operating systems within a DAG, a DAG does not support mixed versions of Exchange, and you can't update a server after Exchange is installed. All these factors mean that you can't just introduce an Exchange 2013 Mailbox server into a DAG that is currently based on Exchange 2010.

The approach that must be taken is therefore to build a new DAG based on Exchange 2013 and then migrate mailboxes to the new Exchange 2013 DAG after it is fully operational. A general outline of the process is as follows:

1. Create the new Exchange 2013 DAG and add Exchange 2013 Mailbox servers to the DAG.

2. Create mailbox database copies within the Exchange 2013 DAG as required.
3. Move mailboxes from databases in the Exchange 2010 DAG to the Exchange 2013 DAG.
4. As mailboxes are removed from Exchange 2010 databases and the databases empty, remove these databases and their copies from the Exchange 2010 DAG.

   As the workload reduces in the Exchange 2010 DAG, you can consolidate it on a smaller number of members and remove the Exchange 2010 DAG members that are no longer required.
5. After removing a Mailbox server from the Exchange 2010 DAG, you can run Setup to uninstall Exchange 2010 and remove the server from the organization.
6. After all mailboxes have been migrated to the Exchange 2013 DAG, remove the last Mailbox server from the DAG and then run the Remove-DatabaseAvailabilityGroup cmdlet to remove the now-obsolete Exchange 2010 DAG from the organization.

As you remove Exchange 2010 Mailbox servers, they can be reused as the basis of a new Exchange 2013 server. Don't bother attempting to upgrade anything because it's simpler and usually much faster to use the "strip to bare metal and rebuild" approach to create a brand-new server running Windows Server 2012 (preferably) and Exchange 2013. When the new server is ready, it can become a member of the Exchange 2013 DAG.

Another point to consider is that you cannot upgrade a DAG member from Windows Server 2008 R2 SP1 to Windows 2012. The server has to be removed from the DAG before Exchange is removed, the operating system is updated, and Exchange is reinstalled.

All this sounds like a lot of bother, but when you think about the situation, it's really not dissimilar from the approach taken for regular Exchange servers, which you cannot upgrade to run a new version of Exchange. Given that this approach is now three versions old and users have become used to installing new versions of Exchange on new (or reused) servers, it does make sense to extend the rule to a DAG. Build a new DAG, migrate mailboxes, and move forward.

Active Manager

Active Manager runs as part of the Microsoft Exchange Replication service (MSExchangeRepl, not to be confused with MRS, the Mailbox Replication service) on every Mailbox server. Both the Replication service and Active Manager are present on Mailbox servers that don't belong to a DAG, in which they are essentially in a standby role waiting for the server to join a DAG (which it might never do). Conceptually, Active Manager is the orchestrator for native-mode high availability in Exchange because it decides which

database copies are active and which are passive, taking into account administrator preferences such as the database activation preference order. You can regard Active Manager as the successor of the resource management model previous iterations of Exchange clustering technology used.

Within a DAG, Active Manager can operate in two roles. One server in the DAG takes on the Primary Active Manager (PAM) role, and the others operate in a Standby Active Manager (SAM) role. Whether in PAM or SAM mode, servers continually monitor databases at both the Information Store and Extensible Storage Engine (ESE) levels to detect failures. However, it is the PAM that determines which database copies are currently active and which are passive; the SAM concentrates on monitoring the health of the Information Store process and the databases that run on the local server. When it detects a failure of the Information Store or a database, the SAM on that server asks the PAM to initiate a failover, providing that the databases affected by the failure are replicated and a viable, healthy, passive copy exists. A failure might be caused by a storage failure that takes one or more disks offline or a problem that causes the ESE database engine to consider a disk unresponsive. If the server hosting the PAM is still online, it initiates the failover. If this server has been taken offline by a failure, another server in the DAG assumes the PAM role following arbitration by the cluster and begins to bring any necessary database copies online to restore service.

The server that holds the PAM role is responsible for processing topology changes that occur within the DAG and making decisions about how to react to server failures, such as deciding to perform an automatic transition of a passive copy of a database to become active because the server that currently hosts the active copy is unavailable for some reason. The server that hosts the PAM role is determined by the server owning the default cluster resource group, but there's no real operational need for you to know which server is currently acting as the PAM. If the server that owns the cluster resource group fails, the PAM role moves to the server that takes ownership of the cluster resource group. The criteria the PAM uses to select the database copy to activate is discussed in the "Best copy and server selection" section later in this chapter. When a new database copy has been successfully mounted, the Active Manager client that runs inside the Replication service queries the PAM to learn which server hosts the newly activated copy so that client connections can be directed to the correct server.

Database copies can be moved from active to passive state through a *switchover*, which is a change initiated by an administrator, perhaps in preparation to take a server offline to apply a service pack or other software upgrade, or a *failover*, which is the result of a hardware or software outage (or administrator accident) that prevents a database from functioning properly. In either case, Active Manager is responsible for selecting and enabling a new copy to accept incoming client traffic and is the definitive source of information about which server has mounted the currently active copy of a database, where passive copies are mounted, and the current database copy state. Configuration information is written back into the cluster database and updated there by Active Manager, but transient information

relating to current database status is held in memory. You can view information about a DAG, including information about the names of the servers in the DAG and the current Active Manager, with the Get-DatabaseAvailabilityGroup cmdlet.

If the PAM fails, its functions automatically pass to another server in the DAG that takes over ownership of the cluster resource group. You can also move the cluster resource group to another server to transfer the PAM role if you want to take the server that holds the role offline for some reason, such as to apply a software upgrade or update.

Automatic database transitions

Figure 9-1 illustrates an example of a typical DAG configuration that is designed to deliver high availability after a storage or server failure. The DAG contains four member servers, each hosting two active databases. Each of the databases is replicated to two other servers to provide the desired level of protection against an outage. If server ExServer1 fails, which affects the active copies of databases DB1 and DB2, the Active Manager process, running as part of the Replication service, activates the best possible copies of these databases. The passive copy of DB1 might be activated on server ExServer2 alongside the passive copy of DB2 on ExServer4. Following a successful activation, Active Manager updates the RPC Client Access service to reroute user connections to pick up the copies of the databases on ExServer2 and ExServer4. Users connected to DB1 will be redirected to ExServer2, and those connected to DB2 will go to ExServer4.

Similarly, if the problem is a disk failure rather than a complete server outage and only the disk holding DB1 on ExServer1 becomes inoperative, Active Manager detects the problem, selects the best copy of DB1 from those available on ExServer2 and ExServer3, activates the chosen copy, and reroutes the workload to that copy.

DAGs support deployment in more complex environments, such as when a DAG is extended across more than one data center. When multiple data centers are involved, it is common to replicate database copies in a way that helps the DAG survive and continue operations even if a complete data center is shut down. Obviously, the design exercise to establish the layout of servers, databases, and copies is far more complex than the work necessary to create a relatively simple DAG like the one illustrated in Figure 9-1.

The scenario shown in Figure 9-1 uses two passive copies for each database. It is possible to run a small DAG in which just one passive copy exists for each active database. Cost of additional hardware and software and your own experience might lead you to decide that the probability of more than one server ever failing at one time is negligible, so it is sufficient to rely on the single additional copy. This is acceptable as long as everyone involved in the decision, including the business owner of the email service, recognizes that this type of high-availability configuration is capable of handling only certain failure scenarios.

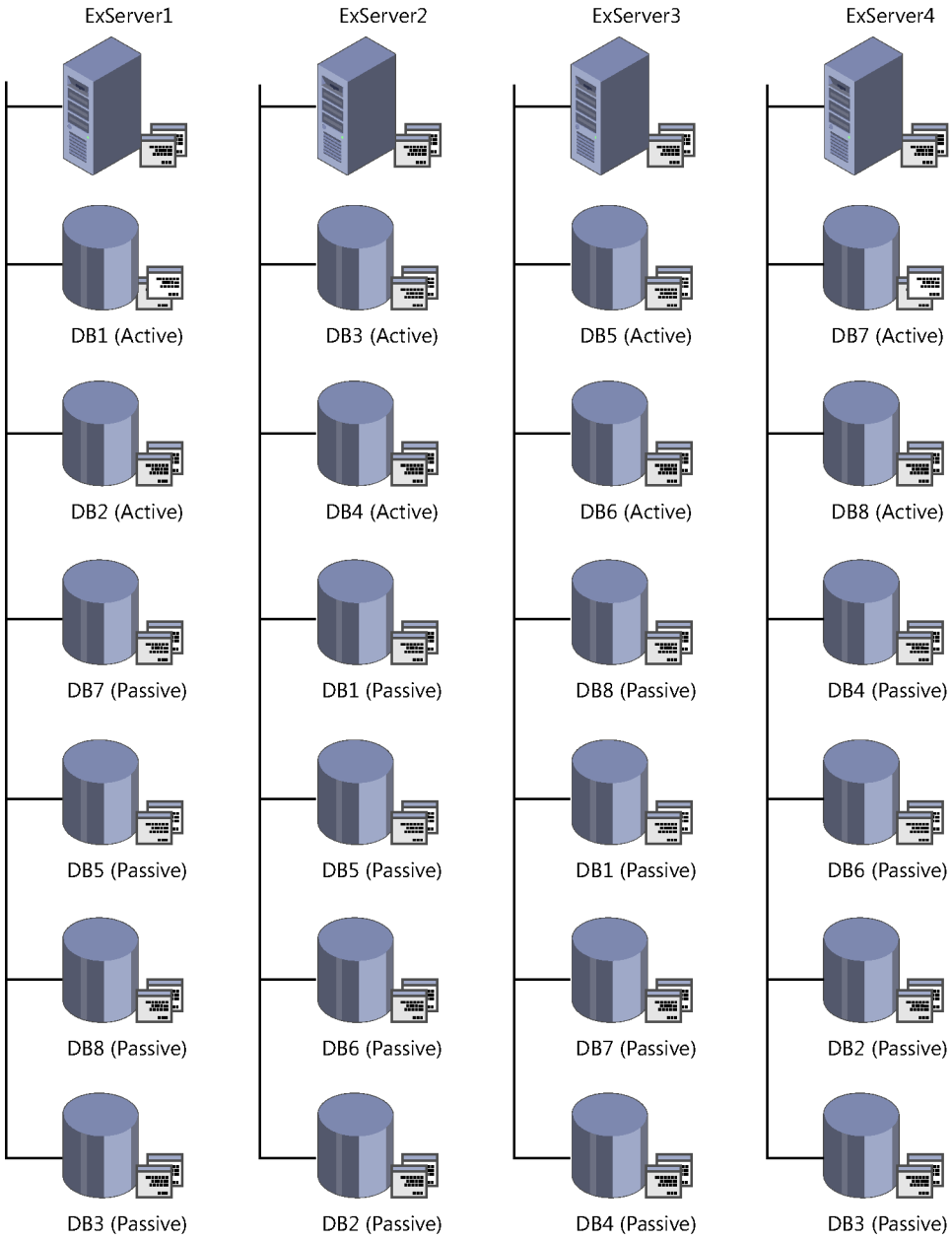


Figure 9-1 Databases and copies within a DAG

Two copies of a database (one active, one passive) provide the minimum protection possible within a DAG. Three copies are so much better because they allow protection to be maintained during common operational scenarios such as a server being taken offline for maintenance. The number of copies you can create for an individual database is limited only by the number of available servers in the DAG, disk space, and available bandwidth. The high-capacity bandwidth available within a data center means that the availability of sufficient disk space to hold replicated databases and transaction logs is likely to be more of an issue. This issue is somewhat negated by the ability to deploy databases on low-cost drives, providing there is sufficient rack space, power, and cooling within the data center to support the disks.

INSIDE OUT **Dismounting an active database is not a failover**

Before discussing the details of how automatic database transitions occur, it's worth making the point that Exchange does not invoke a transition when an administrator dismounts an active database. Although this action creates an immediate outage for users whose mailboxes are in the database, it is deemed an action the administrator has taken deliberately, possibly to allow some hardware maintenance to proceed. Exchange therefore has no reason to execute an automatic failover because these are designed to occur automatically in the event of change in the operating environment that is not caused by an administrator.

Managed Availability and high availability

Microsoft operates a massive multitenant Exchange environment within Office 365. The sheer size of Office 365 makes it impossible for the classic approach to problem monitoring and detection to work smoothly, if only because billions of events might be generated and need to be analyzed and correlated daily. Exchange 2013 introduces a new approach to the problem in the form of Managed Availability, an automated framework to measure the health of server components, detect when problems exist, and take a carefully planned series of actions potentially up to and including a forced server reboot to solve problems. Managed Availability exerts a pervasive and ongoing influence across Exchange 2013 operations and has a particular importance in terms of ensuring high availability of services to end users. As such, this might be the most important feature introduced in this release.

Managed Availability works on the basis that each server should be able to measure its own health and take actions to solve its own problems. To do this, Exchange 2013 installs a series of probes that run on the server to measure the current state of the many components that

collectively make up a server. Some of the probes generate artificial transactions to measure end-to-end message flow. Some depend on Performance Monitor counters to understand where components are under stress. Some measure the health of a protocol such as Outlook Web App or ActiveSync; others measure the end-to-end experience a user receives when he attempts to perform an action, with the aim of ensuring that the system can detect any degradation as experienced by an end user. Some probes run every 20 seconds or so; the more comprehensive tests run every 20 minutes as a backstop to ensure that any lurking problem is picked up. The probes report their results to a monitoring engine that is capable of recognizing patterns and problems. The fact that problems are addressed on the server on which they are detected ensures that Managed Availability can scale from a single server up to Office 365.

If a monitor detects a problem, it invokes responders in an attempt to solve the problem. Many issues are solved by restarting the underlying service, or, as in the case of protocols, perhaps recycling an application pool within Internet Information Services (IIS). Problems are timed from when they are first detected and, if the initial attempt to solve an issue is unsuccessful, the monitor escalates its action. For instance, if recycling the Outlook Web App application pool fails to enable users to connect to their mailboxes with Outlook Web App, the problem probably does not lie within IIS. The monitor therefore forces a failover of the databases to ensure that users can continue to have access to their mailboxes. Later, if the problem still exists, the monitor might force the server to bugcheck. The ultimate escalation is to a human being through a Microsoft Systems Center Operations Manager (SCOM) alert. If you don't use SCOM, you might have to build your own monitoring tool using the Managed Availability cmdlets to signal problems that cannot be automatically resolved.

It's important to realize that Managed Availability does not attempt to determine the cause of any failure. Instead, its role is to detect problems and take the appropriate action to get a component back online so that service is not disrupted for users. However, all the protocol logs and probe results are retained and available for later investigation if it is necessary to conduct a cause analysis with the aim of preventing future similar problems.

As with all new technology, Managed Availability has had its hiccups. One problem that occurred was with the *ActiveDirectoryConnectivityConfigDCRestart* responder in Exchange 2013 RTM CU2. In single-domain forests this responder worked as designed. However, in multi-domain forests, it occasionally forced servers to bugcheck because it considered that Exchange had lost contact with the domain controller from which it retrieved configuration data. Microsoft quickly advised customers how to disable the responder until a longer term fix became available. Problems like this don't make Managed Availability bad. They are simply indicative of the complexity involved in building automated system monitoring and management intelligence into software products.

INSIDE OUT How can I see the sets of Managed Availability probes, monitors, and responders?

If you open the Event Viewer and navigate to the ActiveMonitoring crimson channel, you'll find all the Managed Availability probes (ProbeDefinition), monitors (MonitorDefinition), and responders (ResponderDefinition) defined there. You can extract the information about these objects using Microsoft Windows PowerShell in a form that makes it easier to browse the information. For example, here's how to extract a list of the monitors defined on server ExServer1 and extract the XML-formatted data for each item. The list is stored in a variable that you can then print or view online.

```
$DefMonitors = (Get-WinEvent -ComputerName EXSERVER1 -LogName Microsoft
-Exchange-ActiveMonitoring/MonitorDefinition | % {[xml]$_.toXml()}).event
.userdata.eventxml
```


Managed Availability provides a more sophisticated monitoring framework for Active Manager to check the health of the databases mounted on each DAG member than was available in Exchange 2010. Analyzing the results returned by probes helps Active Manager understand whether servers and databases are in good health and are capable of handling the load generated by client connections. You can get an idea of the number of probes, monitors, and responders that Exchange 2013 uses to ensure that servers remain highly available by looking at the list reported by this command:

```
Get-MonitoringItemIdentity –Identity DataProtection –Server ExServer1 |
Format-Table Identity, ItemType, HealthSetName, Name –AutoSize
```

A health set is simply a known set of probes, monitors, and responders used by Managed Availability to monitor the health of a component. In this case we specify the DataProtection health set and extract details from the server ExServer1 (run the command against a Mailbox server because a CAS server cares little or nothing about this kind of high availability). You'll find that over 60 items are returned, including probes that check on components such as the Replication service endpoint. Separate probes are created for each database that is mounted on a server.

Exchange reports the results of probes in a health set in a concise manner in the form of a health report. For instance, to get a one-line snapshot of the current high availability health for server ExServer1, run the command:

```
Get-HealthReport –Identity ExServer1 –HealthSet DataProtection |
Format-Table ServerName, AlertValue, LastTransitionTime
```

If all the monitors are healthy, you see Healthy reported in the *AlertValue* field. If any monitor is unhealthy for any reason, the value is Unhealthy, which should prompt an examination to determine why the problem exists. All the databases might be online, and another issue such as a failed content index is causing a monitor to flag a problem.

INSIDE OUT **Tracking Managed Availability actions**

The responders Managed Availability uses take defined actions to solve problems. You can see the records of the problems detected by responders and the actions taken to address the problems by examining the RecoveryActionResults crimson channel log (under Exchange/Managed Availability). Here you will find event 500 logged for conditions that cause a responder to take an action, event 501 logged when a recovery action is successful, and event 502 logged for an unsuccessful recovery. For instance, if a database does not have an available copy, Managed Availability cannot force a failover. In this case, you see a 502 event.

As always, Windows PowerShell is extremely useful when the need arises to make sense of the many events that can accumulate on a busy server. The following code extracts all the responder events from the event log and then formats the information into XML format. Finally, search through the output to locate any instance when a forced server reboot was initiated. You might be surprised by the results!

```
$Events = Get-WinEvent -ComputerName ExServer1 -LogName
Microsoft-Exchange-ManagedAvailability/RecoveryActionResults

$EventsXML = ($Events | Foreach -Process {[XML]$_.toXml()}).event
.userdata.eventXml

$EventsXML | Where {$_.State -eq "Finished" -and $_.ResourceName -eq
"EXSERVER1" -and $_.Id -eq "ForceReboot" -and $_.EndTime -like "2013*"} |
Format-Table StartTime, RequestorName -AutoSize
```

Best copy and server selection

To make the choice to replace a failed copy of a replicated database, Active Manager runs a process designed to identify the best possible copy of a failed database. Originally called *best copy selection* (BCS), the process is modified slightly in Exchange 2013 to take account of target server health, so now it's called *best copy and server selection* (BCSS). The aim of BCSS is to take all possible complications and administrative blocks into account when selecting the right database copy to which to transfer service because it's obviously not good to attempt to transfer service to a database that is failing in its own right. BCSS works

by creating a sorted list of available database copies after ignoring any database copies that are:

- On servers that are currently unreachable for some reason (network failure, maintenance, or other conditions).
- Administratively blocked from activation because either the DatabaseCopyAutoActivationPolicy or DatabaseCopyActivationDisabledAndMoveNow properties of the database have been set to Blocked, using the Set-MailboxServer cmdlet.

It is hoped that the resulting list after exclusions has at least one healthy copy in better shape than the failed copy. If only one healthy database copy is available, Active Manager runs the attempt copy last logs (ACLL) process to bring that database copy up to date. If more than one database copy is available, Active Manager proceeds with the BCSS process by sorting the list according to the copy queue length, with the copy with the fewest outstanding logs to copy at the top of the list. The actual sort is performed using the value of the LastLogInspected property. This property contains the date and time the last transaction log file was inspected for the database copy. The result is that the database copy that needs the least work to update after activation is at the top of the list.

Administrators can indicate a preference for a database copy to be activated in case of failure by setting the ActivationPreference property on a database copy with the Set-MailboxDatabaseCopy cmdlet. For example, in a DAG in which database copies are maintained in two data centers, it would be normal to set the activation preference so that the copies in the local data center are activated ahead of those in the remote data center. In this scenario, you can run this command to assign an activation preference value of 2 to the DB1 database copy on server ExServer1:

```
Set-MailboxDatabaseCopy –Identity 'DB1\ExServer1' –ActivationPreference 2
```

You don't want a database copy that is less preferred for activation to be brought online ahead of a more preferred copy, but the only way Exchange knows your preferences for activation is if you set them. To respect administrator choice, Active Manager sorts the list by activation preference. Strictly speaking, activation preference is only used to break a tie when multiple healthy database copies exist and when a lossless transition is required by the AutoDatabaseMountDial setting for the server (a topic discussed in the next section).

The default settings for Mailbox servers enable Active Manager to perform a transition even if some data is lost (normally due to transaction logs that cannot be copied from the last active server), and in this case, Active Manager sorts the list of available healthy copies by their copy queue length. A copy with a shorter copy queue length is always preferred because less potential for data loss exists.

It's possible that the state of health of the copies might not be as good as you'd like, so Active Manager now reviews the state of health of each database copy more thoroughly, using criteria such as the following:

- **Database status** Healthy is best because it means that the database copy is ready to go. The other status values are *DisconnectedAndHealthy*, *DisconnectedAndResynchronizing*, and *SeedingSource*. All these status values indicate that more work is required to bring the database copy online.
- **Content index** Once again, a Healthy status is best because it indicates that all the content in the database has been indexed. A status of Crawling indicates that Exchange is still indexing the content of the database copy.
- **Copy queue length** It is best when the queue contains 10 transaction logs or fewer, which means the database copy has been copying logs from the active database to stay up to date.
- **Replay queue length** Once again, you want to see a moderate queue length of fewer than 50 logs awaiting replay. More than this number indicates that more work is required to bring the database copy online.

Active Manager goes through its list sorted by activation preference and selects a database copy that is healthy, has a copy queue length of fewer than 10, and has a replay queue length of fewer than 50. If it can't find a database copy that matches these criteria, it reduces the standard for activation and looks for a database copy with a content index of Crawling. If this doesn't work, it reduces its standard further until it can find a database copy that matches. Active Manager also checks the value of the MaximumActiveDatabases property on potential target servers to ensure that it does not attempt to activate a database copy on a server that would exceed the maximum value permitted.

It is possible that Active Manager will have to reduce its selection standard significantly before it can find a matching database copy, and that copy might have a low activation preference and be in the process of reseeding. Therefore, it will take a considerable effort to bring the copy online and restore service. Usually, selection can happen well before Active Manager has to reduce its criteria. When this happens, Active Manager activates the selected database and logs its activity in the crimson High Availability channel (Figure 9-2).

Exchange 2013 uses probes controlled by the Managed Availability system to monitor the overall health of servers. BCSS uses the information available through Managed Availability to make sure that the server Active Manager selects is operating normally; it doesn't make much sense to activate a database copy on a server that is faltering or one that is in a worse overall state than the server that previously hosted a database. Health sets are used in the equation, too; these are collections of health measurements for specific Exchange components that Managed Availability measures so that it has a more precise idea of server

health. For instance, if Outlook Web App experiences a problem that forces Managed Availability to initiate a failover, it does not make much sense for Active Manager to activate a database copy on a server that is also having problems with Outlook Web App.

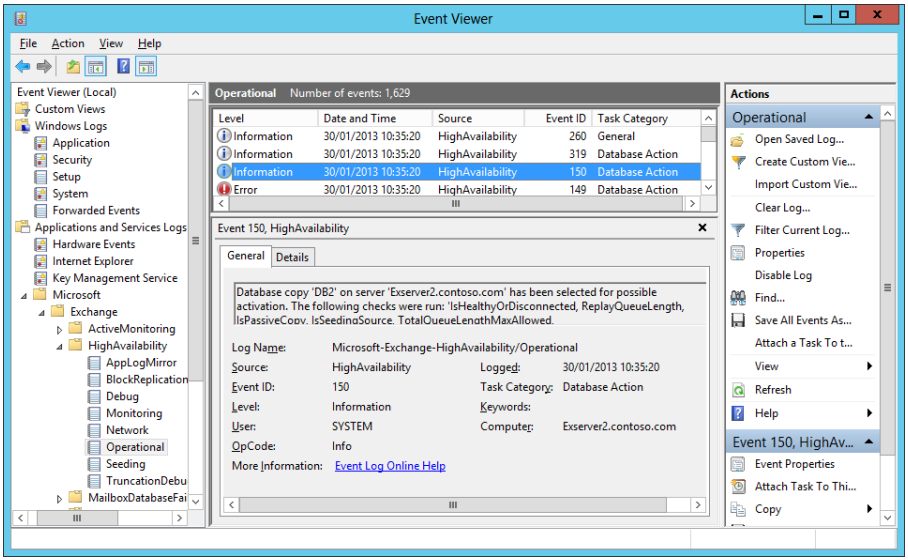


Figure 9-2 Crimson channel event logged when Active Manager activates a database

Accordingly, the BCSS mechanism orders servers that host a copy of the affected database based on the following order of preference:

1. **All Healthy** All monitored health sets on the server are in a healthy state.
2. **Up to Normal Healthy** All the monitored health sets that have Medium priority or higher are in a healthy state.
3. **All better than Source** All the monitored health sets are in a better state than the server that hosts the affected database copy.
4. **Same as Source** All the monitored health sets are in the same state as the server that hosts the affected database.

For example, Figure 9-3 illustrates an instance of event 306 logged in the crimson channel when Active Manager responds to the failure of a Managed Availability responder that uncovers a problem with the Autodiscover component running on a server. (For more information about how Exchange logs important information in the crimson channel, see the "Crimson events" section later in this chapter.) If Autodiscover doesn't work properly,

Microsoft Outlook clients will be unable to connect to Exchange, so Active Manager initiates a move for the database noted in the event (DB2).

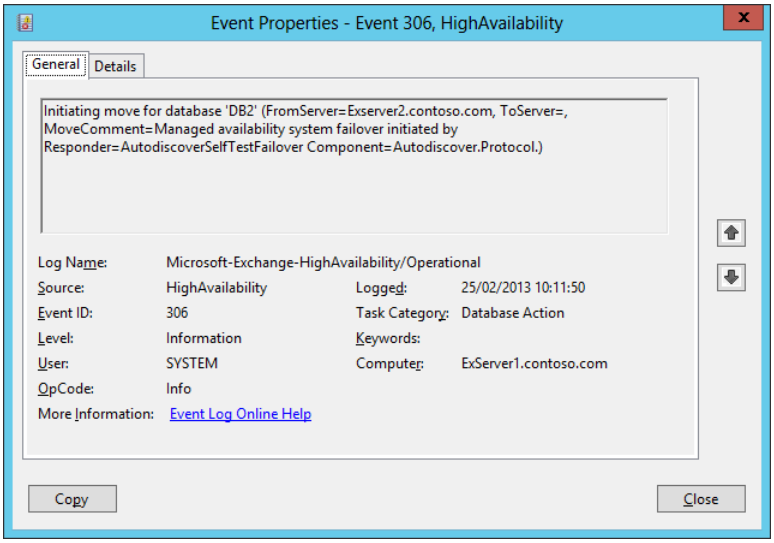


Figure 9-3 Crimson event noted for a component problem

If you continue to examine the events logged in the crimson channel, you should find instances of event 1103 (Figure 9-4), created as Active Manager measures the health of candidate servers that host copies of the database it wants to failover. Each of the protocol stacks running on the server is assessed to determine its health to provide Active Manager with an overview of the server it can feed into its decision process to select the best possible failover target (as described earlier). A separate event 1103 is logged for each member server in the DAG. In this instance, you're looking at the event logged for the server that you know has a problem (event 306 has told you this), so it is no surprise to see that Autodiscover is listed as Unhealthy. You can see some of the other protocols Active Manager assessed, including Outlook, Outlook Web App, Exchange Web Services, ECP (really the Exchange Administration Center), ActiveSync, and DataProtection (high availability). The other protocols all report a Healthy status, so something odd has happened with Autodiscover to force the failover to occur.

One other constraint comes into play if a failover is enforced by Managed Availability because it determines that a failover is necessary due to the failure of a component. The constraint is that Active Manager must select a target server on which the component that failed is healthier than on the failed server.

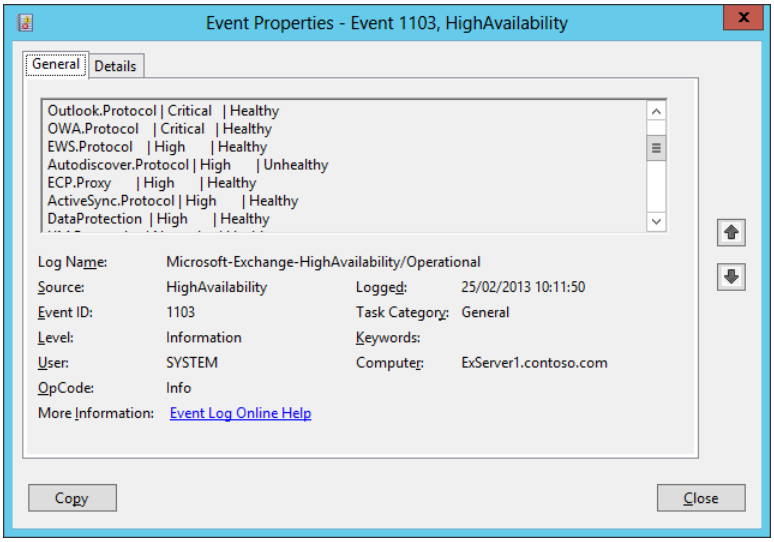


Figure 9-4 Event 1103, showing Active Manager's health measurement for a server

It's possible that no database copy will meet even the least stringent standard for activation. For example, if all the other database copies are offline, Active Manager won't be able to activate any database copy automatically and will report failure. In this case, the administrator has to fix the problem with the original database copy and bring it back online or address the issues that prevent Active Manager from bringing one of the other copies online.

The full set of criteria for the standard for activation is described in "Active manager" at *http://technet.microsoft.com/en-us/library/dd776123.aspx*.

INSIDE OUT Is activation preference ever used?

Active Manager uses a database's activation preference only to break a tie when multiple healthy database copies exist as candidates for activation following a failure and no loss of data is required. As it turns out, experience shows that many failovers occur inside DAGs that have excellent network connectivity between member nodes so that copy queue lengths are usually at or close to zero. In this scenario, multiple healthy database copies are available, and a database's activation preference will be used. It is therefore important to give some consideration to assigning the preferred order of activation to databases.

ACLL: Attempt copy last logs

Assuming that Active Manager can determine a suitable database copy to activate, the Replication service on that server starts to run the ACLL process to copy any missing transaction logs from the server that hosts the failed database copy. The purpose of ACLL is to assemble all available data in the form of transaction logs to enable the Replication service to update the database copy before it is mounted and made available to users. The best outcome is when all outstanding transaction logs can be copied from the server that hosted the failed database copy; the Replication service then replays all the logs, the database copy is completely up to date, and a lossless failover is successful.

Of course, the nature of failure is that a database is most often taken offline because of a storage outage. In this case, the disk holding the transaction logs might also be affected, and the Replication service won't be able to copy any logs. In this scenario, the AutoDatabaseMountDial property of the Mailbox server is consulted to establish the tolerance for data loss caused by missing logs.

CAUTION!

The default value for the AutoDatabaseMountDial setting on an Exchange 2013 Mailbox server is *BestAvailability*, meaning that the Mailbox server is happy to mount a database if up to 12 transaction logs are missing. This represents a data loss of 12 MB and potentially more because transactions for individual messages can extend across logs. Much of these data are probably messages Exchange can recover from Safety Net, so it is acceptable to mount the database and run the risk of a small data loss. Other values are *GoodAvailability*, representing a tolerance of six transaction logs, and *Lossless*, meaning that no data loss is tolerated.

Exchange will not mount a database automatically if the number of missing transaction logs exceeds the limit set by AutoDatabaseMountDial. Two actions can be taken from this point. The administrator can decide that she wants to force the database to mount and accept whatever data loss is incurred, or she can locate the missing transaction logs from a backup (if one is available that has these logs) or from a server that hosts another copy of the database.

If the ACLL process can copy all the necessary transaction logs, Active Manager has a database copy that is up to date and ready to mount. It therefore issues a mount request to bring the selected database copy online and make it available to clients. However, further checks occur to ensure that mounting the database will not exceed the maximum number of active databases configured for the server or that the Mailbox server is suspended for activation. Either of these issues will block activation.

If ACLL can't copy the necessary transaction logs and the database copy remains outside the AutoDatabaseMountDial threshold, Active Manager selects another database copy and starts the process again. This cycle continues until a database copy is successfully activated and mounted or Active Manager reaches the end of the list of available copies and has to declare failure.

Transaction log replay: The foundation for DAG replication

Within a DAG, transaction log data generated by the active database copy are streamed by the Replication service to each of the servers that maintain passive database copies, where the data are then reconstructed, validated, and replayed to update the passive copies. The DAG is the boundary of data replication for transaction logs. In other words, you cannot replicate logs to a server in a different DAG and have Exchange replay the logs into a database replica there. It then follows that before you can create a copy of a database, it must reside in a DAG, and the target server must be part of the same DAG. The DAG is also the boundary for failover insofar as failovers occur within the DAG only.

Exchange uses the Extensible Storage Engine (ESE) streaming backup application programming interface (API) and raw Transmission Control Protocol (TCP) sockets to transfer log data between source and target servers, using one TCP socket per database. The same mechanism is used when a new database copy is seeded. When the seed is complete, the server hosting the active copy of the database streams transaction log data to the servers that host the new passive database copy. When the logs arrive on that server, the Information Store replays the data held in the logs to ensure that the new database is immediately ready to accept a full client load if a transition is required.

The Replication service is composed of a set of components that interact to keep replication and replay working.

- The copier on the active server copies closed log files from the database to the inspector location on each server that hosts a database copy. This process is asynchronous and occurs on a continual basis as new logs are created. A log file is available to be copied when the Store closes off the current log file, renames it, and creates a new current log.

- The inspector on the copy server checks its directory regularly to detect new logs and then verifies that each of the logs is valid. If valid, the inspector moves the log file to the replay directory. If a log fails validation, the inspector copies it again to ensure that temporary conditions (such as a network glitch) do not stop continuous replication from working. If Exchange cannot recopy a log, the passive copy of the database will have to be reseeded.

- The log replayer replays the contents of the copied logs to update the database copy using the same kind of processing that the Store executes for soft database recoveries following an unexpected server outage. The Store replays logs every 60 seconds or after 10 new logs become available.
- The truncate deletor deletes log files the log replayer has successfully used to update the database copy. Full online backups of Exchange databases have traditionally removed log files, but when continuous replication is used, the truncate deletor makes sure that the log files that have not yet been replayed are not deleted.
- The incremental seeder is responsible for ensuring that the database copies are not in a diverged state after a restore has been performed or after a database failover.
- The copy queue is the number of logs that are waiting to be copied from the active server to the one holding the copy. The replay queue is the number of logs that are waiting to be replayed into the copy database.

INSIDE OUT Fluctuations in the number of logs

The number of logs in the replay and copy queues fluctuates in line with the activity generated by clients. Relatively low numbers such as those shown here are of no concern. You should only become worried if you see the number of logs in the queues growing over time and failing to decrease even as the load eases on the server. This kind of situation indicates that something is interfering with log replication or replay or that some source of abnormal load exists. For example, it is possible to generate a very high load on a database when you move mailboxes around. Moving a large 2 GB mailbox generates at least 2,000 transaction logs. The Mailbox Replication Service (MRS) is capable of processing multiple mailbox moves to a target server concurrently, so if you initiate a migration batch containing several large mailboxes to move them to target databases that have copies in a DAG, you immediately generate an enormous load for the CPU on the target server (to process the incoming data, including updating the content index) and the storage subsystem. Exchange's workload management system will throttle back MRS activity if it threatens to cause problems for a server but replication activity is driven by the volume of transaction logs, and large queues are likely.

INSIDE OUT Consequences and how to deal with them

Two consequences usually result from the heavy load generated on the target server. First, all the servers that host copies of the target database will be very busy, and users might experience performance degradation until the mailbox moves taper off toward the end. Second, Exchange might put the mailbox move requests into a StalledDueToHA status and not resume the moves until the server load decreases and the replication queues clear. You can see the StalledDueToHA status by using the Get-MoveRequestStatistics cmdlet to interrogate the current move queue. The stalled status indicates that Exchange has had to back off processing of move requests until the replay queue clears to reduce the number of outstanding transaction logs and ensure the preservation of high availability. Exchange uses a setting called DataMoveReplicationConstraint on mailbox databases to tell MRS the replication state that must exist for the target database before mailbox moves can complete. Usually, the requirement is that at least one of the database copies is healthy and up to date. For these reasons, it is important to schedule mailbox moves at a time when server load is light and the servers can cope with the demand generated by moving data from source to target servers. In addition, you should spread the load of mailbox move requests so that no one database becomes the target of more than three or four concurrent moves.

It's important to realize that the information about replication queue lengths Exchange Administration Center (EAC) displays is a static snapshot that is valid only at the time EAC last fetched information about a database by running the Get-MailboxDatabaseCopyStatus command. Like all the other administrative interfaces in Exchange, EAC contains no business logic or extra intelligence for it to analyze and report on replication activity; EAC depends on the available Windows PowerShell commands to fetch its information. To know what the current activity is, you have to click Refresh to force EAC to execute the Get-MailboxDatabase –Status command and retrieve updated status information about the selected database. Sometimes it is more convenient to see what's happening with replication using the Get-MailboxDatabaseCopyStatus cmdlet.

```
Get-MailboxDatabaseCopyStatus –Identity 'DB1\ExServer2'
```

Note that if you run Get-MailboxDatabaseCopyStatus without passing the name of a database, Exchange returns the current status for all databases on the server on which you run the command. If you use an asterisk (*) instead of a database name, Exchange returns the copy status for all local databases.

Apart from checking that the copy and replay queue lengths are within acceptable limits and that the last inspected log is reasonably recent, the other interesting item Get-MailboxDatabaseCopyStatus returns is whether the content index for the copy is functioning properly. (At times of low user activity, replication activity can be limited to the log files generated because of messages sent by Managed Availability probes.) A Healthy status indicates that items are being properly indexed as transaction logs are replayed into the database copy. It's important for context indexing to work because Active Manager takes the health of the index into account when it considers which copies are candidates for activation following an outage.

Before the Information Store replays transaction logs into a database copy, it verifies the checksum and database signature of the logs to ensure that they are valid. This is equivalent to running the ESEUTIL against the log by using the /K switch. If the checksum fails, Exchange recopies that log. If Exchange fails to copy and validate the log a third time, it is an indication that something very bad is happening within the storage subsystem or network to cause a physical corruption, and you might need to reseed the database.

Many problems will be limited to a single mailbox

It's worth emphasizing at this point that the way Exchange performs continuous background maintenance to fix problematic pages in the database, coupled with the validation of transaction logs during the replication process, does a good job of preventing physical corruption arising from storage problems from infecting multiple database copies. If a page becomes corrupt and cannot be fixed by background maintenance, it will be detected when the replication process attempts to validate the transaction log that contains the corrupted page. It's possible that logical corruption caused by a software bug (not necessarily in Exchange) might sneak through because this might not result in a bad checksum. If this happens, the new database schema makes it likely that the problem will affect only a single mailbox, and you can recover from a backup or a lagged database copy. (In the past, a logical corruption in a table could affect many mailboxes.)

When the Replication service has verified a transaction log, it moves the log into the log directory. The Store then replays the log to update the database copy, using a process similar to the soft recovery process that occurs when Exchange needs to update a database whenever it is mounted. The log replay is also similar to the processing performed when you run ESEUTIL /R to replay logs to a database, but it is faster because Exchange attempts to batch log files to stream transactions into the database and skips the UNDO phrase of ESEUTIL because it's not necessary. A separate instance of ESE that runs inside the Information Store process is responsible for log replay. The Store replays logs in generation order. For example, if the Replication service copies log generation 1876 and the Store is

replaying log generation 1873, it is logical that generation 1876 has to wait until the Store has replayed logs 1874 and 1875. If this does not happen, it might not be possible to replay transactions that span several log generations.

Tip

Transient network conditions could cause the network link between servers to have reduced capacity or be unavailable for a period. When this happens, the queues of transaction logs waiting to be replicated to servers that cannot be reached grow. When the network is restored to good health, normal replication activity clears the backlog, and the database copies on the other servers are brought up to date. There is no need for a reseed in these situations. However, if a database copy is down and the other database copies fill the disk on which their transaction logs are located, the copy is marked as FailedAndSuspended and needs to be reseeded because the Replication service truncates the logs it needs to be updated to keep the other copies online.

The Replication service might detect logs it ignores for updates. For example, for some reason, a copied log file might be too old to apply to the database copy, or it might be corrupt. These logs are moved into the IgnoredLogs directory and can be deleted. The IgnoredLogs directory can have two subdirectories: a time-specific OutOfDate directory (for example, 2013-11-01T10-32-08E00OutOfDate) and an InspectionFailed directory. The first directory holds any log file that is present in the replica directory when a failover occurs. The file was probably created because the replica copy was previously run as the active copy. When a failure occurs, the Store creates a new E00.log file to act as the current transaction log, so if it finds an E00.log file in place, it simply moves it into the E00OutOfDate directory to get the old file out of the way. The InspectionFailed directory is there to hold log files the inspector rejects, perhaps because the Store cannot read the file or believes that the signature in the header is invalid. You won't see these directories on a server until Exchange creates them the first time it activates the passive database copy.

Transaction log compression

From the description so far, it is obvious that Exchange ships a great number of transaction logs around between servers in a DAG to keep all the databases synchronized. To minimize the network impact, Microsoft made some changes to the way data are written to transaction logs. The order of processing is now as follows:

1. ESE compresses all message headers and HTML/text bodies before writing these data to the transaction logs.
2. The transaction logs are compressed as they are replicated to servers within the DAG.

3. The transaction logs are decompressed before they are written to the replication target disk.

4. The compressed message headers and bodies are decompressed when clients request the Store to access the data.

Microsoft believes that this processing reduces transaction log traffic by approximately 30 percent (which might vary across different environments).

Block mode replication

Block mode replication is designed to minimize the delay between a transaction occurring in an active database and being copied and replayed to all the database copies. The idea is that speeding up replication reduces the potential exposure to data loss in the case of failure caused when a transaction log is corrupted between the time it is generated and the time it is copied. In this respect, a transaction log can be considered a potential single point of failure that Exchange needed to address to achieve true high availability.

Exchange depends on the replication and replay of complete transaction logs to keep database copies updated. The current transaction log file is locked, and the data it holds are unavailable when it is in use. Exchange still uses transaction log replication, but the log copier component includes some new code that monitors the state of replication to understand when it is safe enough to begin replication at the block level as data is flushed to disk. In this context, "safe" means that file-level replication is proceeding without any problems in keeping the passive copy up to date, and the database copies are all healthy.

Exchange DAG members use the following approach to control replication modes:

- Replication always begins in file mode. When a server comes online, it contacts the servers that host the active copies of any of its databases to request replication of any log generations it has not yet seen.
- The servers that host the active databases respond with copies of any outstanding logs.
- The server replays the logs and brings its database copies up to date.
- After a database copy is current (in other words, the next transaction log it needs is the current transaction log), Exchange can switch over to block mode.
- In block mode, as transactions are written into the ESE log buffer (the 1 MB cache used to accumulate transactions to be written into a transaction log), the log copier copies the data asynchronously (effectively in parallel) to all the servers that hold a database copy. The data are written into a similar log buffer on these servers.

- When the log buffer on the receiving servers is full, the complete buffer is written out as a transaction log. The new log is inspected to ensure that it is not corrupt, and if this check passes, the log is inserted into the log stream to be replayed into the passive database.
- Replication continues in block mode until a replay queue starts to accumulate on a server. Depending on conditions, the point at which Exchange automatically switches back into file mode is usually when the replay queue reaches four (it could be more or less). Replication continues in file mode until the replay queue is cleared and the next log required is once again the current transaction log. Exchange then transfers back into block mode.

The same data paths are used for block replication as for copying transaction logs, which are still copied to servers that host database copies to ensure that the copies have complete data sets available to them. The advantage of block mode is that transactions are dispatched immediately when they are available to the log buffer of the active database. Transaction data arrive to the servers that hold passive databases as quickly as the server hosting the active database can dispatch data across the network.

Note

Because log file replication continues and the logs are verified on both the sending and receiving servers (a verification failure causes a log rewrite), the previous potential for failure caused by a corrupt transaction log is removed.

An individual transaction might fit into a single 32 KB database page. New data obviously reach the servers that host database copies faster if the log copier only has to wait for and then copy a 32 KB page instead of a complete 1 MB transaction log. The case is less obvious for transactions generated by large multi-megabyte messages because these require the replication of a number of transaction logs. However, sufficient small transactions fit into single pages to make block replication very attractive in terms of its ability to reduce overall replication latency dramatically.

Block mode replication also changes the activation process when Exchange has to bring a passive database copy online. If a copy was using block replication when the failure occurred, Exchange uses whatever partial log content is available to create a complete transaction log that it uses to bring the passive copy up to date as it is activated. This step ensures that the newly activated database has all available data in it when it is brought online. Later, when the failed server comes back online, the incremental reseeding code can resolve any divergence that exists between the data in the fragment used to update the activated database and the data it might have received if replication had been complete.

The normal resolution to any divergence is a request to recopy the complete log generation represented by the fragment that arrived from the database that has all the information. After the missing log is distributed, it can be replayed into the other database copies to make sure that all copies contain the same information.

Transaction log truncation

Busy Mailbox servers generate a lot of transaction logs. If circular logging is not enabled on a database, its transaction logs will continue to accumulate until they are truncated by the Store or manually deleted by an administrator. Log truncation for databases that do not use circular logging typically occurs when a successful backup occurs as the Store removes all the transaction logs that are no longer required because their contents are safely committed within the backed-up copy of the database.

When circular logging is enabled, the Store uses a much smaller set of transaction logs to hold transactions that are pending commitment into the database. After the transactions in a log are committed and the checkpoint has been advanced past the log generation, the Store can delete the log file. This mechanism enables the Store to operate with a set of between 5 and 10 transaction logs, depending on the activity of the database.

Things are slightly different when circular logging is used for databases within a DAG. Several copies of a database can exist within a DAG, and it's important for the Store to take no action that might expose a database to data loss. Deleting a transaction log on one server cannot proceed without being sure that the log might not be required by another database copy. For example, one copy of the database might be offline because its server is down for maintenance. To handle circumstances like this, the Store keeps the transaction logs until it is quite sure they will never be required again. The following checklist is used.

Truncation occurs for transaction logs for databases without lagged copies. (Where lagged database copies are useful is discussed in the "Using a lagged database copy" section later in this chapter.)

- Has the log file been backed up, or is circular logging enabled?
- Is the log file generation less than the database's checkpoint?
- Has the log file been replayed successfully for all but one other database copy?
- Has the log file been inspected by servers that host lagged copies?

If the database has a lagged copy, it uses this variation:

- Is the log file generation less than the database's checkpoint?

- Is the log file older than the replay lag time and truncation lag time configured for the lagged copies?
- Has the log file been deleted on the active copy?

After all checks are met, the Replication service knows that it no longer needs a transaction log, and it deletes it. If all checks are not met, the log is retained until all conditions are satisfied. A lagged copy creates the need to ensure that sufficient storage is available to hold all the logs generated in this period. If you set the lagged copy to keep logs for 14 days (the maximum), you might have to set aside several gigabytes of storage for this purpose.

The set of transaction logs maintained for any copy of a database depends on how quickly the transaction logs are replicated within the DAG and then replayed into all copies of a database. You shouldn't be concerned if you see a temporary accumulation of transaction logs for a database copy because the database might be waiting for an update from the other copies to allow it to truncate the log set, which might not happen for several hours. However, you should investigate to determine the underlying reason if you see the log set steadily grow over time, especially if no reduction occurs during off-peak times when fewer logs are being generated. Truncation actions are logged in the TruncationDebug crimson channel, where you can find the events recorded as logs that are validated and removed.

Incremental resynchronization

Although you'd like to imagine that things will always progress smoothly when transaction logs are copied between servers and replayed to update database copies, the potential for some interruption always exists. Exchange includes a feature called incremental resynchronization to correct divergences in database copies automatically. However, an automatic correction can only occur under certain conditions when Exchange can be reasonably sure that it has access to enough data to enable it to correct matters. These are as follows:

- Following an automatic failover of a database. Exchange checks all the copies of a database to ensure that they are consistent.
- If you create a new copy of a database and some files (database and/or log files) already exist at the copy location on the target server. This might happen if you enable and disable a database copy and stops you from having to reseed the database each time.
- Following resumption of replication activity if you stop the Replication service for any reason.

Whenever the Store detects a divergence condition between an active database and one of its copies, it looks in the available transaction logs to determine where the divergence took place. It then locates the relevant pages in the problem database copy, reads the changed

pages from the active database, and copies any necessary log files from the active server. The update information for the problem pages is then applied from the copied transaction logs into the diverged database copy to synchronize it with the active copy.

Exchange also includes the ability to update individual pages within a database. In previous versions of Exchange, especially those prior to Exchange 2007, administrators hated seeing error -1018 or error -1022 logged in the application event log because these events indicate that the Store had detected a page-level corruption in a database. Corruptions usually occurred as the result of some hardware problem, often in storage controllers, and although a database would continue to run with the corruption, the problem was that the only way to fix a database that contained a corruption was to restore from the last good backup. Given that some databases were well over 100 GB, this wasn't a popular option with administrators.

A page corruption in a single-copy database is a severe problem. Replicating the corruption to many database copies creates a problem of an entirely new dimension. For this reason, Exchange detects and fixes page-level corruptions that occur in active or passive database copies.

If the Store detects a problem page in the active database, it places a marker in the log stream (in the current transaction log) that acts as a request for a valid copy of the corrupted page. The request is sent to all database copies, where it is inspected and processed along with other log content. When the Information Store replays data for the passive copy, it notices the marker and responds to the request by invoking a Replication service callback to ship a copy of the page to the server that hosts the active database. When this server receives the replicated page, the Store patches it back into the active database to remove the corruption. Other servers that host passive copies might also respond with pages, but these are ignored after the active database has been restored to good health.

The process to fix a corrupted page in a passive database copy is slightly different. In this case, the server that hosts the passive copy immediately pauses log replay. Log copying continues to ensure that all the transaction logs that will eventually be required to bring the database completely up to date are available on the server. The server then requests a copy of the corrupted page from the server that hosts the active database, using the internal ESE seeding mechanism. The active server responds with the page data. The passive server then waits until all the log files necessary to bring it up to date past the point at which the active server provided the page (as indicated by the maximum required generation) have been copied and inspected. When it is sure that all the required data is available, the passive server then restores the corrupt page and resumes log replay to clear the backlog of transaction logs that have accumulated since the corruption was first detected.

Seeding a database

Seeding a database is the process Exchange uses to create an initial database copy or to re-create a database copy if the current copy cannot be updated through log replay. During a seed, Exchange copies the database from source to target and then updates the copy by replaying the transaction logs that have accrued since the copy operation started.

The time required to seed a database depends on the network speed between the two servers and the current workload each server is under. Given the wide variation in environments, it is hard to give a definitive figure for the expected throughput, but you should expect to achieve a copy rate of up to 50 GB/hour if you use a 100 Mbps network. Therefore, if you have a 1 TB database that you need to reseed, you can expect this operation to take more than 20 hours to complete. You might consider this an unacceptable length of time to reseed a database, but the only real solution is to increase network capacity or select a different (perhaps better connected) server to act as the source. Reseeding a 1 TB database is much faster on a 1-gigabit network. Even though Microsoft has attempted to minimize the need for reseeding, the potential always exists for this to be necessary, so if you plan to use very large databases, you also need to consider fast replication networks.

Changes in message submission within a DAG

Transport processing changes slightly within a DAG to achieve better high availability for message transport. Normally, the Mailbox Transport Submission service submits new outgoing messages to the Transport Service running on the same server. Within a DAG, the Mailbox Transport Submission service round-robins among the other DAG member servers and will failback to a local instance if no remote servers are available. In addition, the transport system attempts to route inbound messages by a server other than the one that hosts the active mailbox database.

Day-to-day DAG management and operations

Setting up a new DAG is neither something to do on a whim nor something you should attempt unless you have sufficient knowledge of all the technologies that are involved. These include the following:

- Exchange 2013
- Windows Failover Clustering
- Networks

Exchange hides most of Windows Failover Clustering when it builds and maintains a DAG. In the normal course of events, you should never have to look at the cluster details unless a

problem occurs and you have to carry out some investigation to retrieve information, probably under the direction of Microsoft Support. Even so, it's still a good idea to know the nature of the beast you are dealing with, if only at a reasonably high level.

In addition, the account you use needs to have sufficient permissions to install all the necessary components or to make changes to the Exchange organization or the Windows system. Setting up a new DAG for a test environment is an interesting exercise in the application of a combination of computer software and hardware to solve a problem, and it's something you need to go through several times before you deploy a DAG into production. When that time comes, you need to understand what form the DAG will take and what purpose it will serve. For example, important considerations include which servers will participate in the DAG, how the servers will communicate to replicate transaction logs, which mailbox databases will be replicated to which servers, where the file share witness (FSW) for the DAG is located, and how the overall design will operate under different circumstances such as an individual or multiserver outage. These details should be written down and validated before any step is taken to create a production-quality DAG.

Assuming that all the up-front work is done, you can create a new DAG through EAC or by running the New-DatabaseAvailabilityGroup cmdlet through Exchange Management Shell (EMS). Either approach works well. Remember that you have to pre-stage the Cluster Name Object (CNO) before beginning to create a new DAG if you use Windows 2012 servers for the member nodes. (See the "Pre-staging the Cluster Name Object" section later in this chapter.)

To create a DAG through EAC, first move to the Servers node in EAC and then to the Database Availability Group tab. Click the + (plus) sign to begin creating the new DAG. You can then enter the name of the new DAG, which can be up to 15 characters in length and must be unique across the forest. You have the opportunity to provide details of the computer that will act as the witness server and the path for the witness directory. You can leave these values blank if you want Exchange to create them automatically, which it will do by searching for a suitable CAS that does not have the Mailbox role installed. If Exchange can't find a suitable server, you can update the DAG properties afterward to specify a server you choose.

The witness server must be in the same forest as the DAG. Microsoft has experimented with using a cloud-based Azure File Server as the witness server. However, at the time of writing, some technical restrictions exist that make such a configuration impracticable. Any Windows 2008 R2 or Windows 2012 server can act as a witness server for a DAG, as long as the File Server feature is installed on the computer. A file or print server is fine. Do not use a computer that might become part of the DAG in the future; however, if you do, you can amend the DAG properties to select a new witness server. As you can see in Figure 9-5, you can also assign IP addresses to the DAG. If you leave this section blank, Exchange will use Dynamic Host Configuration Protocol (DHCP) to assign addresses. If necessary, you can amend the DAG properties to assign static IP addresses afterward, but it is a good idea to add all the

necessary IP addresses (one for each subnet) to the DAG up front so you don't forget to do this later.

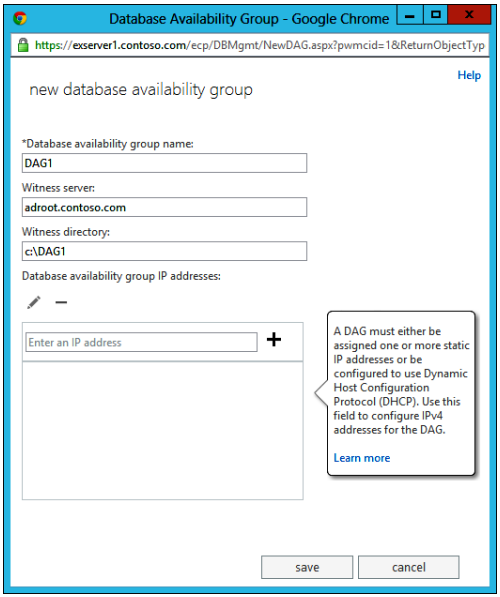


Figure 9-5 Creating a new DAG

You do not have to take any other step to enable Exchange to manage the FSW; some commentators have blogged that you should add the machine account for the server that hosts the FSW to the Exchange Trusted Subsystem group. Such advice should be ignored because you have to take extreme care when you allow access to the Exchange Trusted Subsystem group. This group permits access to any Exchange object in Active Directory, and you need to keep it as restricted as possible. Failure to allow Exchange to manage the witness directory will mean that the DAG might not be able to maintain quorum if member servers fail, with resulting loss of service to users.

Note

A single computer can act as the witness server for multiple DAGs as long as each DAG is assigned its own unique directory. Microsoft recommends that the easiest way to meet these requirements is to select a CAS in the same Active Directory site that hosts the DAG. (A DAG can span multiple Active Directory sites, so in this instance, you select a hub transport server in one of the sites to act as the witness server.) One good reason for choosing an Exchange server to act as the witness server is that you can then be sure that an Exchange administrator can manage all the DAG components, including monitoring the status of the witness server.

It is also entirely possible for an administrator to remove or wipe a server that is acting as the witness server for one or more DAGs with no complaint from Exchange until the next time a DAG needs to use the witness server to maintain quorum. At that time, Exchange will discover that the server is no longer available, and quorum might be lost. Fortunately, the simple fix is to edit the DAG properties with EAC or use the Set-DatabaseAvailabilityGroup cmdlet to select another server to act as the witness. For example:

```
Set-DatabaseAvailabilityGroup –Identity 'DAG1' –WitnessServer 'Computer1.contoso.com'
–WitnessDirectory  'C:\DAG1'
```

In scenarios that stretch a DAG across multiple data centers, you should select a server so that a network interruption between the primary and secondary data centers doesn't trigger a failover or halt DAG operations. One approach is to place the witness server in the primary data center on the basis that it is just that—the primary. Another is to place the witness server in a third data center that has good network connectivity with both the primary and secondary data centers.

INSIDE OUT Create a firewall rule for remote management

Firewalls can get in the way of communications—even legitimate communications—so if Windows Firewall is enabled on the target server, you should create a firewall rule to allow remote management to avoid remote procedure call (RPC) access errors during DAG communications. A rule such as the one shown here is appropriate:

```
NetshAdvfirewall Firewall set rule group="Remote Administration" new enable=yes
```

You can also create a DAG through EMS. The following example creates a new DAG called DAG2 and assigns a static IP address to the DAG. You can also see how the witness server details are specified as parameters to the command. As discussed earlier, you don't have to register an IP address at this point because DHCP can be used. (Microsoft does not support Automatic Private IP Addressing [APIPA] for DAG networks.) Exchange 2013 supports IPv6 addresses for DAGs only if IPv6 is implemented on all member nodes and if IPv4 is also enabled. Exchange 2013 also allows you to specify both IPv4 and IPv6 addresses for the DAG, and if you specify IPv4 addresses when member nodes run IPv6, Exchange automatically assigns an equivalent number of IPv6 addresses.

Windows Failover Clustering registers the IP address as a network resource for the Windows cluster resource that underpins the DAG. Essentially, you can think of the IP resource as a pointer that identifies the DAG in DNS. As you'll see when DAG networks are discussed, one IP address is enough for a DAG that exists on a single subnet. If you want to add servers to the DAG that span different subnets, you must update the DAG object with the

Set-DatabaseAvailabilityGroup cmdlet to add an IP address for each of the subnets you want to use. Most companies prefer to use static IP addresses for server resources:

```
New-DatabaseAvailabilityGroup -Name 'DAG2' -DatabaseAvailabilityGroupIPAddresses
192.165.1.8 -WitnessServer FileServer1 -WitnessDirectory "C:\DAG2"
```

Note

Get-DatabaseAvailabilityGroup only reports the static properties that are stored in Active Directory. If you want to see the full properties for the DAG, including the properties Exchange maintains, use Get-DatabaseAvailabilityGroup –Status.

DAGs use Majority Node Set clusters, so at least half the votes that exist within the cluster must be available for the cluster to run. Each server node has a vote. The cluster adjusts itself automatically to maintain quorum as member servers join and leave. After the new DAG is created, it is an empty Active Directory object with no member servers, and no cluster has yet been created. Adding the first member server to the DAG also creates the cluster and begins the process of building out the DAG. As you add Mailbox servers to the DAG, the cluster automatically changes to use the appropriate quorum model, and the DAG begins to use the witness server to maintain quorum. Inside a fully formed DAG whose member servers are all online, the witness server is not required to maintain quorum. However, if member servers fail or are taken offline, the witness server becomes more important. For example, in a four-node DAG, the quorum is three (50 percent plus one of four member servers, rounded to the nearest whole number). If two of the member servers fail, the witness server is needed to maintain quorum and to keep the cluster online.

When active users are spread across two data centers, it is sometimes better to run two DAGs rather than attempt to run a single DAG with servers in both data centers. The logic is that a DAG has a single witness server, so if you operate a single DAG, its witness server (or the majority of DAG member servers for DAGs with an odd number of members) must be located in one of the data centers, which in turn creates a potential single point of failure for the users located on servers in the other data center. Any network outage that removes access to the witness server from the other data center creates a condition in which the witness server is inaccessible and is therefore unavailable if it's needed to maintain quorum. You are concerned about network connectivity between the two data centers, it is usually better to create two DAGs and locate a witness server for each DAG in each data center.

The Microsoft preferred approach is to use Exchange's continuous replication technology to maintain database copies within a DAG. It also supports a replication API to enable other vendors to build DAG solutions with their own feature set. I have no experience of using third-party replication solutions with a DAG and suggest that careful testing of these solutions should be done before they are placed into production with Exchange 2013.

INSIDE OUT **Dynamic quorums**

Windows Server 2012 introduces the ability to prevent a node from voting in the quorum process. This allows for a feature called *dynamic quorums* by which an administrator can change quorum voting by using Windows PowerShell to force the cluster to continue working in situations such as if the working members drop below 50 percent of the cluster membership. The feature has been back-ported to Windows 2008 R2 SP1. Manipulating quorum voting is an interesting and worthwhile capability to have but not something to try on a live cluster unless you have no other choice. It is a better idea to experiment with the feature in a test environment and decide whether it offers any value for your production setup. See *http://technet.microsoft.com/en-us/library/hh831414.aspx* for more information.

Building the DAG

Immediately after creation, a new DAG is an empty container that is waiting to be filled with Mailbox servers and their databases that bring functionality to the DAG. Being able to construct a DAG gradually over time is one of the advantages the implementation brings to Exchange. The alternative, which you saw in the implementation of clustered Mailbox servers in Exchange 2007, is to install all the servers in the cluster at one time, which is almost always impractical when configuring a DAG spanning more than two or three servers. After the DAG is created and its properties verified, you can start to add Mailbox servers to the DAG before creating new database copies that will be managed by the servers. This process doesn't have to begin immediately, and you can keep an empty DAG in the organization for as long as you need to before you begin to add servers.

Servers are added to and removed from the DAG by using EAC or the Add-DatabaseAvailabilityGroupServer and Remove-DatabaseAvailabilityGroupServer cmdlets. To begin to manage DAG membership with EAC, in the Database Availability Group section of Servers, click the Manage Database Availability Group Membership icon. You can then select the new member from the set of available Mailbox servers. You can select several servers to add in one operation; however, it's usually more manageable to add new members individually, taking the time afterward to ensure that everything is working properly before you add the next member.

All the servers in a DAG must use the same version of the operating system. If you start with a server running Windows Server 2012, you can only add new servers to the DAG if they also run Windows Server 2012 (it is currently unknown if this situation changes with the release of Windows Server 2012 R2). If the Mailbox server you want to add to the DAG

does not already have the Windows Failover Clustering feature installed, you can add the Windows Failover Clustering feature on the local server or have Exchange install it when it adds the Mailbox server to the DAG (Figure 9-6). Exchange cannot install Windows Failover Clustering on a remote server because the feature must be installed locally.

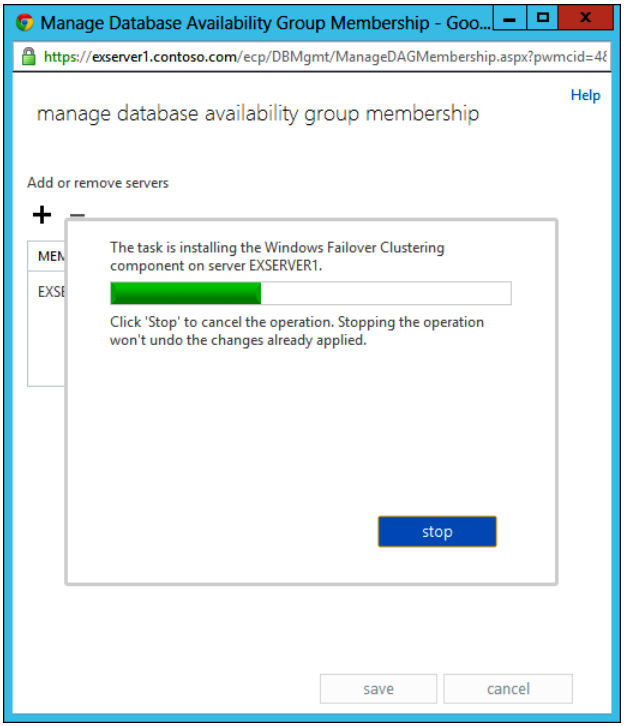


Figure 9-6 Windows Failover Clustering installed as a server joins a DAG

CAUTION!

Microsoft recommends against installing Exchange on a domain controller. It therefore follows that it's not particularly excited if you add a Mailbox server that happens to run on a domain controller to a DAG. Exchange doesn't block this action, and you can use a domain controller if you really must; hopefully, you do this only in a test configuration and not in production use, if only because it creates a situation Microsoft doesn't support.

The following steps occur to instantiate a DAG with the addition of the first Mailbox server. Exchange has to do quite a lot of work to bring a server into a DAG, especially if it has to install Windows Failover Clustering, so this process takes a little time.

1. Exchange validates that the server has the Mailbox server role installed and does not host the witness server for the DAG.
2. If not present, Exchange installs Windows Failover Clustering on the Mailbox server.
3. A failover cluster is created using the name of the DAG.
4. A Cluster Name Object (CNO) is created in the Computers organizational unit (OU) in Active Directory. However, if you use Windows 2012 for your member servers, you have to pre-stage the CNO before you begin creating the DAG.
5. The name and IP address of the DAG are registered in DNS as a Host (A) record.
6. The Mailbox server is linked to the DAG object by populating the MSExchMDBAvailabilityGroupLink property on the server object in Active Directory.
7. The cluster database is updated with information about the databases the newly added server hosts. These databases remain as standalone active copies until you create additional passive copies through replication to other servers in the DAG.

Although it is easiest to add servers using EAC, you can also do this through EMS. For example:

```
Add-DatabaseAvailabilityGroupServer –Identity 'DAG1' –MailboxServer 'ExServer2'
```

When you add a Mailbox server to the DAG, Exchange does the following:

- Validates that the Mailbox role is installed on the server.
- Joins the server to the cluster.
- Adjusts the quorum model. A node majority model is used for DAGs with an odd number of members, whereas a node and file share majority model is used for DAGs with an even number of members. Exchange automatically adjusts the quorum model as servers join and leave the DAG. The adjustment occurs in the background and does not require any administrator intervention.
- Links the server to the DAG object in Active Directory.
- Updates the cluster database with information about the databases the newly added server hosts.

Pre-staging the Cluster Name Object

All DAGs are built on top of Windows failover clusters. All Windows failover clusters use a Cluster Name Object (CNO), an Active Directory computer object that essentially gives the cluster (and therefore the DAG) its identity. The CNO is also used for all communications within the cluster. You don't have to know anything about the CNO to work with the DAG because this object is used for internal purposes within the cluster. However, you do have to create the CNO manually before you add any Mailbox servers to the DAG if the servers run Windows 2012. This isn't the case when you work with Windows 2008 R2 SP1 because Exchange can create the CNO when it builds out the DAG. However, Windows 2012 made a change that restricts the creation of computer accounts, and this change had an unfortunate knock-on effect of stopping Exchange 2013 from creating the CNO, hence the need to pre-stage a CNO before you build the DAG. Indeed, some experts recommended that you always pre-stage the CNO no matter what operating system is used because this removes a potential failure point from the DAG creation process.

The steps to pre-stage a CNO for a DAG are straightforward.

1. Use the Active Directory Users And Computers console to create a new computer object in the desired organizational unit (OU). Often, this is the default Computers OU that is present in every Active Directory, but there is nothing to prevent you from using a different OU, such as a new OU called Exchange DAGs. For guidance on how to redirect the default creation of computer accounts in Active Directory, see *http://support.microsoft.com/kb/324949*.

2. The name assigned to the CNO will be the name of the DAG. Do not use a name such as Exchange or Email because the potential exists for these names to be confused with an application now or in the future. Select a name that clearly indicates the purpose of the object, such as DAG1 or DAG-DC1.

3. After you create the new computer object, right-click it and disable the account. The account must be disabled; otherwise, Windows Failover Clustering cannot use it as a CNO, and you'll get Active Manager errors reporting that the DAG object cannot be found when you attempt to add the first server to the DAG.

4. You now need to assign the necessary permissions to the new computer object. First, make sure that Advanced Features are enabled from the View menu in the Active Directory Users And Computers console. Select Computer, click Properties, and then click Security to reveal the current security settings for the object.

5. Exchange 2013 uses its membership in the Exchange Trusted Subsystem security group to manage the DAG. Therefore, the Exchange Trusted Subsystem must be added to the list of objects that have Full Control access. Click Add, type **Exchange**

Trusted Subsystem as the object name to which permissions are to be assigned as shown in Figure 9-7, and then select Full Control in the Allow column. Click OK to complete the assignment.

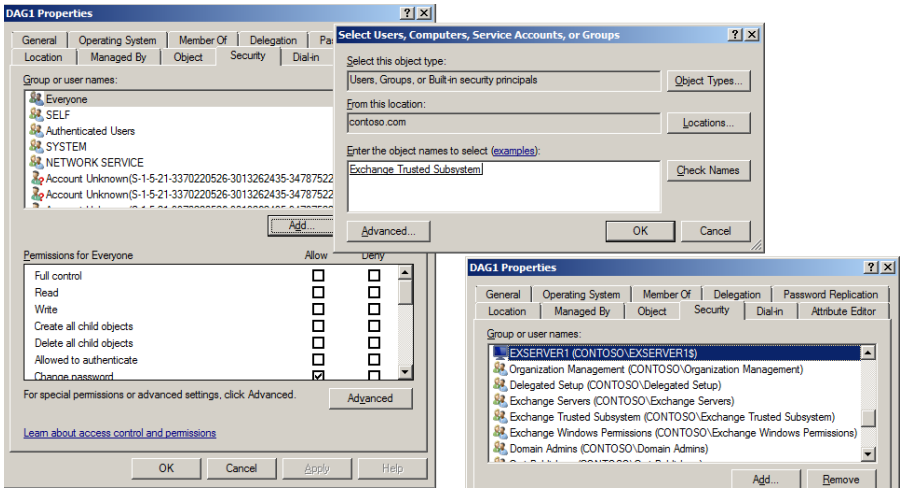


Figure 9-7 Pre-staging a CNO for a DAG

6. After the permissions have been assigned, you can create the DAG through EAC or by running the New-DatabaseAvailabilityGroup cmdlet. You do not have to assign Full Control access to the other member nodes that are subsequently added to the DAG after the first member node is in place.

INSIDE OUT DAG resources and Failover Cluster Manager

Although a DAG is visible to the Failover Cluster Manager, you should never attempt to manage any of the resources the DAG uses through this console unless you have very good reason to do so, such as adjusting node weight to force quorum. Don't expect much sympathy from Microsoft Support if one of your changes compromises the integrity of the DAG. Exchange stores many important properties for a DAG in Active Directory, and the only way you can manipulate DAG settings properly is through EAC or EMS. If necessary, the underlying code in the DAG management cmdlets will update the settings of the Windows cluster.

Before a server can join a DAG, it must be able to communicate with the cluster service running on every other member server that is currently in the DAG. In other words, you cannot expect to populate a DAG if some servers are offline or experiencing network problems. There are a number of reasons communication might not be possible, including the following:

- Servers are powered off or otherwise unavailable.
- The cluster service is not running on a server.
- Firewall rules on a server are blocking communication to the cluster service.
- The DNS service is unavailable.
- Authentication problems (Kerberos, Active Directory, or NTLM) are interfering with secure server-to-server communications.

Production servers should have at least two network interface cards (NICs) to enable them to isolate Messaging Application Programming Interface (MAPI) traffic (interaction with other servers including Active Directory) and replication traffic (log shipping and database seeding). Ideally, the NICs are connected to separate networks. Using multiple NICs and separate networks is not a difficult technical requirement from an Exchange perspective because it is perfectly feasible to have all traffic routed across a single NIC, assuming the NIC has sufficient capacity to handle the network traffic. (Microsoft recommends Gigabit Ethernet for single NICs.) However, Windows Failover Clustering also has a dependency on a solid network, and because dependable replication is so important to the smooth operation of the DAG, it's really best if you deploy servers equipped with multiple NICs for DAGs. Additional replication networks can be added as required, and you can take advantage of techniques such as NIC teaming to improve overall network resilience against failure. (See the "DAG networks" section later in this chapter.) You can also consult TechNet to determine how best to configure network settings for DAG operations.

The role of the FSW

A file share witness (FSW) exists to help the failover cluster that underpins a DAG remain operational. It is a network file share created on the witness server that is a cluster resource. The role of the FSW is to act as a tiebreaker when quorum cannot be established by the available member nodes.

All the member nodes in the DAG have a vote. For instance, four votes are available in a four-member DAG. The DAG can continue to function as long as a majority of the votes are available, so as long as the cluster knows of three available votes, it can operate. If two servers are unavailable, the cluster uses the FSW to maintain quorum. Two votes are available and two votes are not. The cluster knows that one of the surviving members (the server

that owns the default cluster group) has a server message block (SMB) lock on the FSW and therefore assigns that member a weighted vote of 2. The vote is now 3 (2 for the weighted vote and 1 other member) against 2 (the two failed members), so the cluster can maintain quorum and remain operational. In effect, the FSW is only used when the cluster needs to determine quorum. It has no other function and is not used during normal DAG operations.

You nominate a witness server and a witness directory as part of the properties of a DAG. The witness directory holds the FSW. These properties can viewed as follows:

```
Get-DatabaseAvailabilityGroup -Identity DAG1 | Select *Witness* | Format-List
WitnessServer             : adroot.contoso.com
WitnessDirectory          : c:\DAGFSW
AlternateWitnessServer    :
AlternateWitnessDirectory :
```

These properties tell you that the DAG1 Database Availability Group uses a witness server called adroot.contoso.com and that the FSW is located in the network file share in the c:\DAGFSW directory. Exchange creates this directory when the first Mailbox server is added to the DAG. Exchange uses its privileges gained through the Exchange Trusted Subsystem security group to create the directory and establish the file share, which is why you need to make sure the Exchange Trusted Subsystem is added to the local administrators group on the server you want to host an FSW unless Exchange is installed on that server. Figure 9-8 shows the sharing properties of the FSW directory, in which you can see that the file share that's been created is known as \\ADROOT\DAG1.contoso.com.

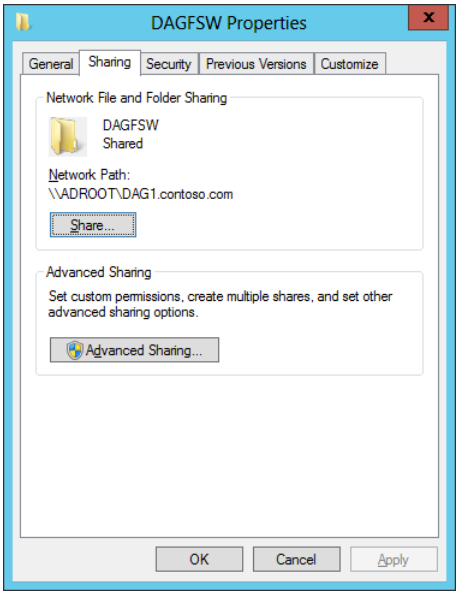


Figure 9-8 The file share witness for a DAG

The AlternateWitnessServer and AlternateWitnessDirectory properties are not populated for DAG1. These properties are used only in multi–data center scenarios when you want the DAG to use a different witness server in a different data center following a data center switchover. The basic witness server properties can also be viewed using EAC, but EAC does not show the alternate witness server properties. These have to be accessed and managed through EMS.

Exchange 2010 provided the first opportunity to learn about the operational issues around DAG management. As such, most people were conservative in their designs and selected a witness server on the same site as the DAG members. For stretched DAGs that had members in two sites, the witness server was typically in a data center in one site, and an alternate witness server was in the data center in the other site. However, the closely coupled nature of the CAS and mailbox roles in Exchange 2010 meant that it was sometimes not easy to coordinate DNS switchovers and database failovers across the two sites. The situation is different with Exchange 2013; users have more experience with DAG operations, and Microsoft has deliberately decoupled the server roles to enable the creation of a single unified namespace that spans multiple sites.

With this in mind, it is now possible to consider how the witness server might be placed so that it is always available to DAG members in multiple sites following the failure of a network link. For example, consider when DAG members are split across two data centers, each of which is a separate Active Directory site. The witness server is in a third site. All the sites are connected with a resilient high-speed network. All the servers are running Exchange 2013. Assume that the wide area network (WAN) forms a redundant mesh topology so that a single WAN failure cannot make a site lose connectivity to more than one other site and cannot affect any site seeing any other site. In this instance, having the witness server on a completely different site from the sites that host the DAG members is perfectly acceptable, but only for Exchange 2013.

It's also possible to envisage a situation in which a single server on one site hosts the witness server for separate DAGs deployed on other sites. The point is that DAG design and practice evolves continually. A statement of "You cannot do that" might be true at one point in time but completely outdated soon afterward because of new knowledge or a software update Microsoft provides. For this reason, it's wise to keep an eye on websites that discuss Exchange and the proceedings of conferences at which Exchange topics are discussed, or just ask a question in the TechNet forums to see the advice others might offer. As always, before you put any advanced technique into production, it's wise to test first, especially for failure conditions.

DAG task logging

All the activities DAG maintenance operations perform are captured in log files stored in the \ExchangeSetupLogs\Dagtasks directory. Each log file is named DagTask with the date and time when the task was performed, so it's easy to find the relevant file.

The system event log is also a useful source of information if you need to track down the source of a problem. In addition, the crimson events Exchange logs in the high-availability channel are often revealing about what happens in terms of Active Manager. More detailed cluster-related logging is captured in the Windows\Cluster\Reports directory. This information is pretty verbose but could hold the key to some of the more esoteric issues that occur from time to time. One way to limit the mass of data in the logs you have to review is to attempt to correlate it with any event log entries that seem to be related to the problem.

Crimson events

If you've been a Windows administrator for any period, it's likely that you are very familiar with the event log. Two sets of logs are available on Windows. The Windows logs are those that are most often used and therefore most familiar. Many Exchange events are recorded in the Application event log, whereas other interesting events about the operation of the system are recorded in the Security and System logs. Collectively, these logs are intended to capture events that apply to the entire system. A separate category of logs called Application and Services Logs capture *crimson channel* or application-level events. The intention is for applications to be able to use the crimson channel to record important events in a way that prevents their loss in the large number of general events that accumulate on a production Windows server.

One way of thinking about the difference between the two sets of event logs is that the crimson channel tracks important events about specific parts of applications, whereas the general logs record the day-to-day functioning of the overall application. For example, Exchange routinely records information about the global catalog and domain controller that a server uses to access Active Directory in the application event log. This is interesting and worthwhile background information to know but not quite at the level or importance of high-availability events captured in the crimson channel.

The crimson channel first became available with Windows 2008, so Exchange 2010 was the first version of the application to begin using the crimson channel. Active Manager is both a publisher to and a subscriber for the crimson channel, monitoring it to detect problems that ESE and other components report. If a problem is detected, Active Manager then takes the appropriate action. For example, if ESE reports a corrupt page in a database, Active Manager initiates the single page patching process to recover good copies of the affected pages from other copies of the database. As explained earlier in the "Managed Availability

and high availability" section of this chapter, the Managed Availability system also makes extensive use of the crimson channel to record information about its activities.

Exchange 2013 captures more information in more detail, so crimson events really are a rich source of information about exactly what happens on a DAG member server during database transitions and other events. In addition to events relating to high availability, Exchange also records events about Managed Availability and operations concerning the troubleshooting scripts in the crimson channel.

INSIDE OUT The Database Availability Group management service

Exchange 2013 RTM CU2 introduced the Database Availability Group management service (MsExchangeDAGMgmt.exe). The new service was created by moving the non-critical code used to analyze and report on DAG events out of the Replication service with the intention of streamlining the Replication service and focusing it on the task of replicating information between database copies. Even though the DAG management service now reports events to the crimson channel, to maintain backward compatibility the events still seem like they originate from the Replication service.

You've already seen how Exchange 2013 logs crimson channel events in the "Best copy and server selection" section earlier in this chapter. To give another example, Figure 9-9 shows a typical event captured in the *Operational* channel. In this instance, the Incremental Reseed process that runs when a database copy is brought online (possibly after a failure or maintenance operation) has determined the set of transaction logs it has to replay to bring the copy of the DB2 database on node ExServer1 up to date. You can see that 238 transaction logs are required, which either must be already present on the server or must be copied from the server that hosts the active database copy to allow the reseed to happen. Other interesting events that you'll see here are requests to Safety Net to replay messages after a database transition or the number of logs that the ACLL process determines have been lost when a database copy is activated following a failure.

Exchange also uses the *HighAvailability* channel to capture startup and shutdown events for the Replication service and the various components that run within this service such as Active Manager, replication operations, the TCP listener, and the VSS writer used for backups. This channel also captures events relating to database health and transitions and anything to do with a DAG's underlying cluster. However, the *MailboxDatabaseFailureItems* channel captures information about failures that affect normal operation of a replicated mailbox database. Collectively, these events provide a great deal of information about the

operations of a DAG and can be of enormous assistance in figuring out what went wrong—if something does!

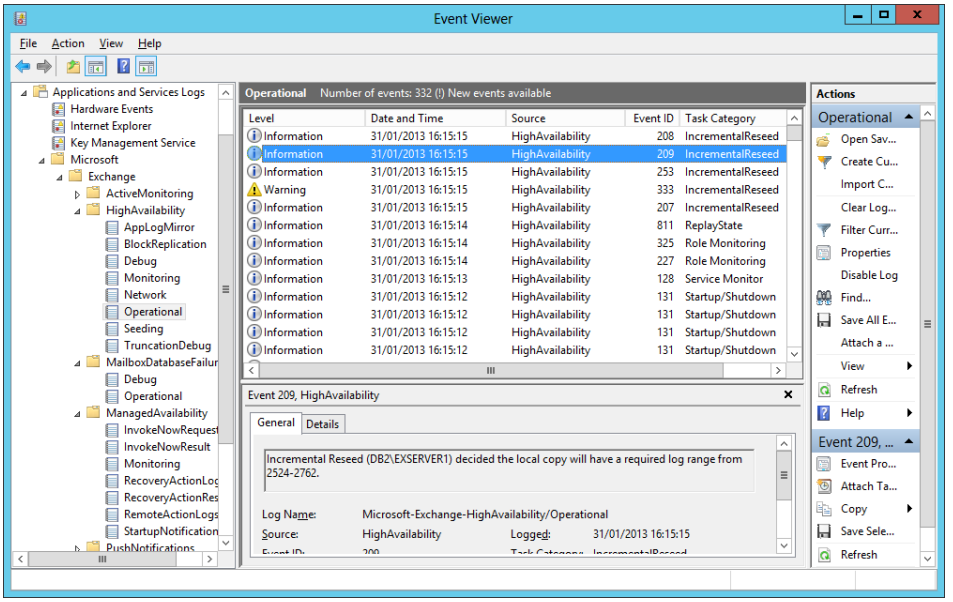


Figure 9-9 Crimson channel events

INSIDE OUT This isn't a starting place

These events probably won't be the first place you look if you have to resolve a database failure. More obvious issues such as a storage failure or a network outage are more likely to seize your attention. However, they are an interesting and valuable source of background data that you might be asked to consult if you work with Microsoft Support to resolve a particularly complex issue with a DAG.

Managing DAG properties

After the DAG is created, you can alter its properties to meet operational requirements through EAC or EMS. You need to be a member of the Organization Management role group to modify DAG properties, which are listed in Table 9-1.

TABLE 9-1 DAG properties

Property	Use
FSW share (WitnessServer)	Path for the share used by the FSW.
FSW directory (WitnessDirectory)	Name of the directory used by the FSW.
Alternate FSW share (AlternateWitnessServer)	Path for an alternate FSW.
Alternate FSW directory (AlternateWitnessDirectory)	Name of an alternate directory used by the FSW.
Network encryption (NetworkEncryption)	Disabled: No encryption of data on the DAG network. Enabled: Encryption used for replication and seeding. InterSubnetOnly: Encryption used on DAG networks in the same subnet. SeedOnly: Encryption used only for database seeding.
Network compression (NetworkCompression)	Disabled: No compression. Enabled: Compression used for replication and seeding. InterSubnetOnly: Compression used for DAG networks that use different subnets. SeedOnly: Compression only used for database seeding.
Replication port (ReplicationPort)	TCP port used for replication activity; if you change this value from the default (64327), make sure that you also make the necessary change on any firewall that is in place to allow replication traffic to pass through the open port.
IP addresses (DatabaseAvailabilityGroup-IpAddresses)	IP addresses for the subnets used by the Windows failover cluster that underpins the DAG.

You can use the Set-DatabaseAvailabilityGroup cmdlet to update DAG properties through EMS. (Only some of these properties are available through EAC.) For example, this command manipulates a number of DAG properties:

```
Set-DatabaseAvailabilityGroup –Identity 'DAG1' –WitnessServer 'adroot.contoso.com'
–WitnessDirectory 'D:\FSWDAG1' –NetworkEncryption SeedOnly –NetworkCompression
Disabled –DatacenterActivationMode Disabled
```

By default, DAG network traffic is not encrypted or compressed unless it is replicating across different subnets. The NetworkCompression and NetworkEncryption properties of the DAG are therefore set to *InterSubNetOnly*. It's easy to change these values because they are both properties of the DAG. As such, the values you assign apply to all traffic between all members and cannot be altered for individual servers. Table 9-1 includes the available options you can set for DAG encryption. For example, this command enables compression and encryption for all traffic within a DAG:

```
Set-DatabaseAvailabilityGroup –Identity 'DAG1' –NetworkCompression Enabled
–NetworkEncryption Enabled
```

Exchange uses existing code for intra-DAG communications

Exchange uses the Windows Kerberos security support provider (SSP) to encrypt inter-DAG communications. This module contains routines that enable applications to encrypt, decrypt, sign, and verify the data contained in messages sent between application components such as DAG members. When two DAG members communicate, they each act as an endpoint and begin the connection with a handshake and key exchange. After the connection is set up, encrypted traffic that can be decrypted by both members flows across it.

Exchange also supports compression for intra-DAG communications based on the LZ77 algorithm. Once again, a familiar algorithm is reused because LZ77 (otherwise known as Microsoft XPRESS) is deployed to protect Messaging Application Programming Interface (MAPI) RPCs flowing between Outlook and Exchange.

You can use the Get-DatabaseAvailabilityGroup cmdlet to review the properties of a DAG. For example:

```
Get-DatabaseAvailabilityGroup –Identity 'DAG1' | Format-List
```

At some point, you might want to change the location of the witness server. Perhaps the server is unavailable for some reason, or perhaps it is more productive to use another server. In this case, you can use the Set-DatabaseAvailabilityGroup server to configure a new witness server for the DAG. This example command selects a server called ExServer22 (which cannot be a member of the DAG) and specifies a directory on that server to host the resource:

```
Set-DatabaseAvailabilityGroup –Identity 'DAG1' –WitnessServer
'ExServer22.contoso.com' -WitnessDirectory 'C:\FSW'
```

Changing the witness server should only ever be attempted when the DAG is healthy and the witness server is not actively in use to maintain quorum.

DAG networks

A simple DAG composed of two or three Mailbox servers, such as the one discussed so far, has relatively basic network requirements. The DAG has to be able to replicate transaction logs between the Mailbox servers, and the network has to be able to carry the replication load and any other network load that other activities generate such as client connections, Active Directory and DNS lookups, message transport, and traffic from other applications. A single NIC installed in each network server is capable of handling the load, albeit with the understanding that a single NIC also represents a potential single point of failure for the DAG. However, for test and other environments that do not need to operate at the level required for large production systems, a single NIC is often sufficient. At one point in the

development of Exchange 2010, Microsoft considered making multiple NICs mandatory for DAG member servers because this was then required by Windows Failover Clustering. Under pressure from the Exchange team, the Windows group pulled back from this position because it meant that low-end servers might not be able to participate in a DAG. Instead, the position is that you can deploy servers equipped with a single NIC as long as you are satisfied that the solution is robust enough for your purpose.

The preferred approach for larger DAGs is to split network traffic across two networks:

- The MAPI network takes care of client connections but can also be used for database seeding and transaction log replication if a replication network is unavailable. There is always a MAPI network within a DAG. Logically, because it services client connections, the MAPI network is registered in DNS and, in a multi-subnet environment, it has a default gateway.
- The replication network is used for database seeding and transaction log replication. You don't have to run a replication network. However, high-end Exchange Mailbox servers that have to handle a substantial replication load can be configured with multiple replication networks. Replication networks do not have to be registered in DNS.

You can also use the Get-DatabaseAvailabilityGroupNetwork cmdlet to view the network information for a DAG. For example:

```
Get-DatabaseAvailabilityGroupNetwork –Identity 'DAG-Dublin' | Format-List
```

When two NICs are available for a server, the MAPI network is typically assigned to one NIC, and the replication network is assigned to the other. As you add more NICs, you increase the network capability for the replication network. When a server is configured with more than one replication network, Exchange attempts to direct traffic across the least used network to maximize available network resources. Microsoft recommends using Gigabit Ethernet NICs in production Exchange Mailbox servers. It's also worth noting that the single biggest cause of network problems within DAGs is outdated drivers. For this reason, it is a good idea to check the NIC drivers on all production servers regularly.

INSIDE OUT **Rubbish in will mean a bad DAG network out**

Code cannot work except on the basis of input data. If that data is suspect, then the code is unlikely to produce good results. For DAG network auto-configuration to work properly, it's essential for all the NICs on DAG members to be configured correctly. For instance, a NIC intended for use as a MAPI network should have a default gateway set, but one to be used for a replication network should not. The Exchange auto-configuration code is good, but it cannot make sense of badly configured NICs.

If you elect to use multiple networks, the same configuration must be used for every server in the DAG. Using multiple NICs provides better resilience against failure; if the replication network becomes unavailable for some reason, Exchange automatically reverts to using the MAPI network for both sets of network traffic, and you won't have to deal with a server outage. When the replication network is restored to health, Exchange starts to use it again if you suspend and then resume replication. However, the server won't be able to function as a member of the DAG if the MAPI network fails and, in this case, Exchange will failover the active databases from the server to other members of the DAG.

As previously discussed, when the DAG was created with the New-DatabaseAvailabilityGroup cmdlet, an IP address was specified to identify the Windows cluster resource that underpins the DAG. (Alternatively, you could have leased an address using DHCP.) This IP address is registered in DNS, as is the address the NIC uses for the MAPI network. Assigning a single IP address to the DAG is sufficient as long as all the Mailbox servers within the DAG share the same subnet. However, if the DAG spans several IP subnets, you must specify an IP address in each subnet for the cluster resource to enable servers in that subnet to participate in the DAG.

INSIDE OUT A note about latency

The existence of multiple subnets often implies different geographic locations, and the sole requirement is that the round-trip network latency between member servers in the DAG must be less than 500 milliseconds. This isn't a hard requirement in that your DAG will immediately fail when latency exceeds the limit and, indeed, the original recommendation was to deploy DAGs only when the network supported a latency of 250 milliseconds, so the recommended limit has evolved in line with experience. If your latency exceeds 500 milliseconds, you can expect queues of transaction logs to accumulate during peak production hours, and any network outage or other operation that generates a lot of network activity will result in even larger queues that Exchange has to clear to bring database copies up to date. In some instances, large queues can build up even when latency is less than 500 milliseconds, and you might need to increase network capacity so that logs are replicated quickly. The same latency requirement exists for DAGs that use a single subnet.

Multiple subnets are also required if you deploy a DAG across multiple Active Directory sites. This is not an issue if the network requirements are met and Active Directory replication is functioning normally. Exchange 2013 is more intelligent than Exchange 2010 is when it comes to configuring DAG networks in multisite environments. As long as all the NICs installed in member servers are correctly configured in advance, the process of adding

servers to the DAG ensures that network data is updated correctly. You should not, for instance, have to go through the process of collapsing (merging) unwanted logical DAG networks into a single logical network, as is often the case with Exchange 2010. Automatic configuration can't work if underlying network problems exist, such as missing static routes for secondary networks, but as long as the network is in good health, Exchange can automate the network configuration for a DAG.

If necessary, you can exert manual control over DAG networks. To do this, use EAC to update DAG network control or run the Set-DatabaseAvailabilityGroup command to set the ManualDAGNetworkConfiguration property to $True. For instance:

```
Set-DatabaseAvailabilityGroup –Identity DAG1 –ManualDAGNetworkConfiguration:$True
```

After running this command, you can perform tasks such as adding or removing DAG networks or disabling replication on a MAPI network.

You can use the Get-DatabaseAvailabilityGroupNetwork cmdlet to view the networks that are available to a DAG. If, as in the previous example, you had added servers from multiple subnets to the DAG, you would see the IP addresses specified for those subnets in the information returned here. Use the Set-DatabaseAvailabilityGroupNetwork cmdlet to change parameters for a DAG network. In this case, you change the properties of the network to enable it for log replication.

```
Set-DatabaseAvailabilityGroupNetwork –Identity 'DAG1\DAGNetwork01'
–ReplicationEnabled:$True
```

The IgnoreNetwork parameter is available to instruct a DAG not to use a network, something that is often done to exclude Internet Small Computer Interface System (iSCSI) networks from being used by a DAG on a permanent basis. This need exists because when a Mailbox server is configured to join a DAG, all the available networks on the server are discovered and included as a DAG network. You do not want to use iSCSI networks within a DAG because these networks are intended to link storage, not to carry general-purpose traffic. For example:

```
Set-DatabaseAvailabilityGroupNetwork –Identity 'DAG1\iSCINetwork'
–IgnoreNetwork:$True
```

The Remove-DatabaseAvailabilityGroupNetwork cmdlet is available to remove a network from a DAG. If you remove a network, traffic will flow across the remaining networks that are available to the DAG. You can't run this cmdlet to remove the last network in a DAG.

```
Remove-DatabaseAvailabilityGroupNetwork –Identity 'DAG-Dublin\DAG-Dublin Network02'
```

Cleaning up before creating database copies

The Exchange setup program creates a default mailbox database whenever it installs the Mailbox role on a new server. Unless you use command-line qualifiers to name the mailbox database and place it in a specific directory, Exchange will generate a unique name for the database and install it in a directory under the Exchange root. Although the name of the database isn't so much of a problem for DAG operations, it is unlikely that you'll want to store every copy of the database under the Exchange root, if only because it is generally held to be good practice to separate binaries from data.

For these reasons, you might want to remove the default mailbox database from a server before you join it to a DAG. This is best done immediately after installing the server because the only mailboxes that will be present in the database are the two health-monitoring mailboxes Exchange creates to use with Managed Availability probes. These commands select the health mailboxes from all databases (hopefully there will be only one database) on the ExServer3 server and then remove them. You will be asked to confirm that it's okay to remove the mailboxes.

```
Get-MailboxDatabase -Server ExServer3 | Get-Mailbox -Monitoring | Remove-Mailbox
```

After the mailboxes are removed, you can remove the mailbox database. Once again, you will be asked to confirm that the operation can proceed:

```
Get-MailboxDatabase -Server ExServer3 | Remove-MailboxDatabase
```

Removing a database only removes it from Exchange configuration data; you have to delete the files from the server to complete the cleanup. The Mailbox server now has no databases, but it is a member of a DAG, so it's ripe for hosting some database copies.

Using circular logging inside a DAG

Best practice for versions of Exchange prior to Exchange 2010 dictates that circular logging is never enabled for production databases. However, the logic that underpinned this recommendation is now outdated and does not take into account the introduction of the DAG and multiple database copies. Microsoft now recommends that you enable circular logging for databases that are protected by at least three copies because the availability of multiple copies ensures that adequate protection exists against failure. The Microsoft recommendation does not mean that you always have to have three copies of every database in a DAG; three copies are only required if determined by your DAG design. For instance, you might decide to use low-end disks that are potentially more prone to failure and thus require more protection through additional copies. In other words, you should understand how each copy is used in production and be able to justify it on that basis. Enabling circular logging is interesting from a technical perspective, but it's probably an insufficient reason to create an additional copy if that's the only reason to have three database copies.

Circular logging has been part of Exchange since its first release. The introduction of DAG technology meant that it had to evolve to account for new circumstances. Two distinct modes of circular logging can be used for mailbox databases. The first mode is traditional circular logging, much the same as has been used since Exchange 4.0 and the only type that can be used with the older type of public folder databases. The second is referred to as continuous replication circular logging, or CRCL, the method the Replication service uses when databases operate within a DAG. Why the difference? It's simply because a DAG is a far more complex environment than a standalone server, and the system has to accommodate configurations such as lagged database copies when transaction logs are retained for a set period before they are replayed to update a database copy. In addition, traditional circular logging is a function of the JET database engine, whereas the Replication service manages CRCL.

Log truncation, the process whereby unwanted transaction logs are deleted or marked for reuse, requires additional management, too, because the Replication service has to ensure that it retains transaction logs until no possibility exists that they might be needed to bring a database copy up to date. Within a DAG, servers update each other by using the Replication service with the current status of transaction logs and so can determine when it is safe to delete logs because the logs are no longer required for replication. For example, the server holding the active copy of a database won't truncate logs if the server(s) that host passive copies are down for some reason to ensure that every log that might possibly be required to bring a database up to date is available.

Even though enabling circular logging seems to be a matter of setting a simple property, the switch between traditional circular logging and CRCL requires manual intervention by an administrator. However, this is only required when the first database copy is created in a DAG and when the last copy of a database is removed because these are the events that mark the transition into and out of database redundancy. If you want to create a copy of a database on which traditional circular logging is enabled, you must first disable circular logging, dismount the database, mount it again, and then create the database copy. This process enables Exchange to ensure that all available data is available to create the new copy and that log truncation does not occur while a database copy is being seeded. (A small database will seed quickly; a larger database might take several hours, and all logs must be maintained until the seeding process is complete.) Later, after the new database copy has been fully seeded, you can switch the database back to circular logging by updating database properties. At this point CRCL is used rather than traditional circular logging. The interesting thing is that if you now create another database copy in the DAG, you don't have to change the circular logging settings because CRCL is in use, and the Replication service is managing the database accordingly.

The basic steps to enable circular logging for database copies within a DAG are first to create the database, then set up its copies, and finally enable circular logging. Exchange signals an error (Figure 9-10) if circular logging is enabled when you attempt to add the first

passive database copy. The need to switch from one form of circular logging is necessary because of some subtle differences in the way circular logging works for databases that are not in a DAG and those that are in a DAG. As you'll recall, inside a DAG, transaction logs are shipped between servers that host database copies, so turning circular logging on and off could confuse matters, especially for lagged copies. (See the "Using a lagged database copy" section later in this chapter for more information.) In addition, the behavior of circular logging differs when a database has one copy and when it has more than one copy. The upshot is that you should decide whether to use circular logging for databases in a DAG early in the deployment and stay with that configuration from that point onward.

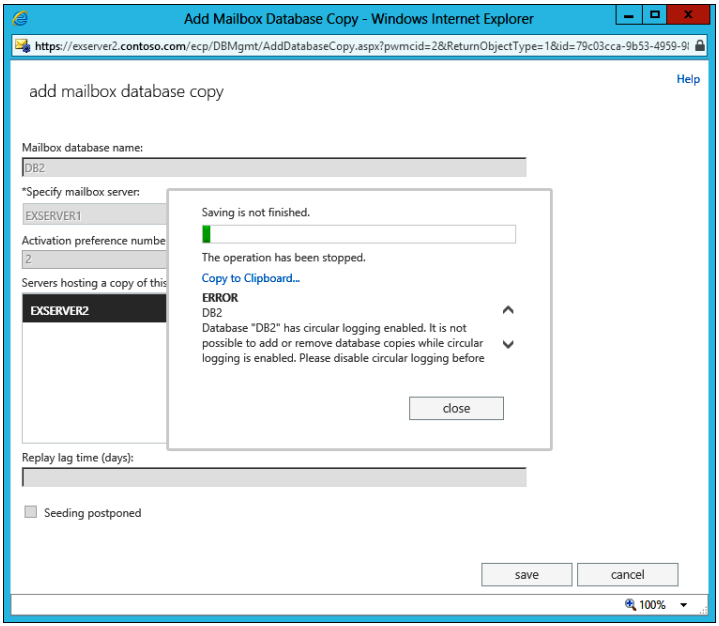


Figure 9-10 Error flagged when circular logging is enabled for a database

To disable circular logging from EAC to allow the new database copy to be created, select the database, click Properties, and select Maintenance to expose the option to enable or disable Circular Logging for the database. Remember to dismount and remount the database to make the setting effective. The EMS commands to disable circular logging and then dismount and remount a selected database are:

```
Set-MailboxDatabase -Identity 'DB2' -CircularLoggingEnabled $False
Dismount-Database -Identity 'DB2'
Mount-Database -Identity 'DB2'
```

After the new database copy is created, you can reverse the process and enable circular logging again.

The nature of circular logging is that transaction logs are discarded regularly as transactions are committed to the database, but when you create new database copies in a DAG, you want to be sure that all possible transaction logs that might be required are available, so enabling circular logging is best left until the database copy is up and running. In addition, as discussed in the "Removing database copies" section later in this chapter, you cannot remove database copies from a DAG while circular logging is enabled because you have to switch the database back into noncircular logging mode first, so you need to take this requirement into account as you approach database management.

INSIDE OUT More logs than you expect?

As you know, when you enable circular logging for a database that does not have any copies in a DAG, the Store truncates transaction logs after their contents are committed to the database. Depending on its usage patterns, a database might have fewer than 20 logs and, during periods of low user activity, the number of logs will reduce to five or so. Things are slightly different within a DAG, where different criteria are applied to decide when the Store or Replication service can remove log files. The big difference is that a log file is only ever removed when it is below the checkpoint. In other words, there is no possible chance that the log file will be required for recovery purposes. And because Exchange 2013 uses a 100 MB checkpoint for both active and passive databases, it is reasonable to expect that at least 100 transaction logs will be present at all times for each database. This happens even when a database is essentially quiet and subject to low user demand (Figure 9-11), when Exchange uses a process to close transaction logs on a regular basis to keep replication going. Of course, much activity might occur under the cover of a seemingly quiet database because Managed Availability generates synthetic transactions, and the Managed Folder Assistant processes retention settings for mailboxes.

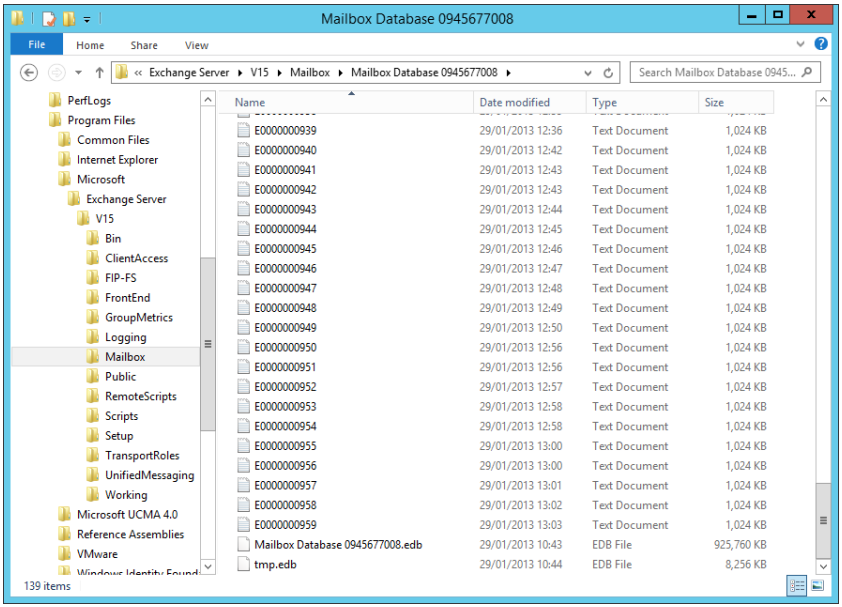


Figure 9-11 Transaction logs for a quiet database copy

Adding new database copies to a DAG

You cannot create a new database copy on the same Mailbox server that hosts the active copy. In other words, you cannot have two copies (active and passive) of a database on the same server. You also cannot have two passive copies of a database hosted by the same server. This restriction aside, you can create a new copy with EAC or by using the Add-MailboxDatabaseCopy cmdlet. Either method requires you to determine some settings in advance:

- The database that you want to copy. Remember that you can only make a copy of a database that exists within the DAG.
- The Mailbox server in the DAG that will host the new database copy.
- The activation preference number (or preferred list sequence number) for the new copy. Each copy in a DAG has a number that Active Manager takes into account to determine the order in which Exchange will attempt to use database copies when lossless failovers are used. The active database has preference 1, and the database copy with preference 2 will be used before the copy with preference 3, and so on. When you add a new copy, Exchange sets its activation number one higher than the current highest activation value. Accept this value unless you have good reason not to. If you make a mistake with the activation order, you can change its value

by selecting the copy from the list of database copies and updating its properties. Exchange then reorders the other database copies to fit in with the new activation number you assign.

To begin the process of creating a new database copy with EAC, select the database you want to copy listed under Databases in the Servers section and then choose Add Database Copy from the drop-down menu. EAC then invites you to select the Mailbox server to host the new copy (Figure 9-12). You can browse to select from a list of the available Mailbox servers within the DAG. When you have selected the target server, click Save, and Exchange will begin to seed the new copy on that server by copying the database unless you select the option to postpone seeding of the new copy. The time taken to seed the new copy depends on the size of the database files (Exchange database [EDB] and transaction logs) and the network speed between the two servers. It is a good idea to schedule the seeding of large database copies (more than 100 GB) for times of low network demand such as outside normal working hours.

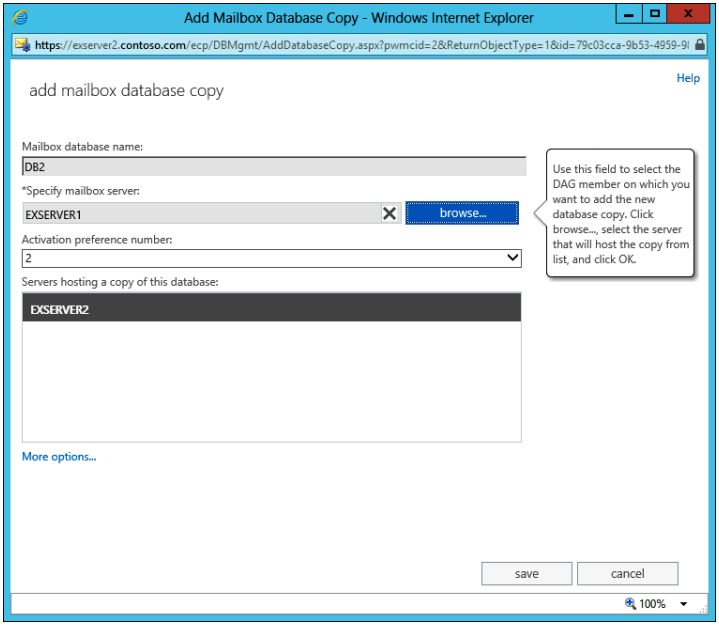


Figure 9-12 Selecting a server to host a database copy

Figure 9-12 includes a More Options link. If you click this link, EAC reveals options to:

- Set a lag time for the new database copy. You can use this option to create a passive database copy that Exchange keeps updated to a certain point (the lag interval)

behind the active database copy. For example, you could create a database copy that lags one week behind the active copy. The maximum lag interval is 14 days.

- Postpone database seeding. You create a database copy in Exchange configuration data but leave the actual act of seeding to later. This option is often used when the databases are too large to seed over an extended network connection or when you want to seed from a different source than the active database. The new database is seeded by sending a physical copy of the database to the server's location, where it is then loaded and synchronized by using the Update-MailboxDatabaseCopy command to bring the new copy fully up to date. This technique is sometimes called FedEx seeding. Because every transaction log created since the physical copy was taken will be required to bring the copy up to date, you cannot use this technique if circular logging is enabled for a database.

After the new database copy is seeded, the Store begins the process of resynchronizing it with the active copy. EAC updates you on its progress as the seeing proceeds. Exchange copies the EDB and transaction log files from the server that hosts the active copy, and you should see the copy status change to Healthy when seeding is complete. During this period, the copy queue length decreases first, followed by a decrease in the replay queue length as Exchange synchronizes the new database copy. Eventually, the two queues reduce to zero, and the new database copy is fully up to date. At this point, if you compare the transaction logs on the servers that host copies of a database, you should see that the same generations are present on each server. This is easily seen by comparing the end of the file name for the transaction logs—if E1000009A9.log exists on the server hosting the active copy, a log ending in 09A9 should exist on the servers that hold the passive copies.

Databases that do not use circular logging could have thousands of log files the Replication service on the receiving server has to copy, inspect, and potentially hand to the Store to replay to synchronize the new server fully. If you monitor the replication status of the database copy with the Get-MailboxDatabaseCopyStatus command, you see two queues reported. The copy queue length is the number of transaction log files that have yet to be copied from the server hosting the active copy. The replay queue length is the number of transaction logs that have been copied to the server and inspected successfully but still have not been replayed into the database copy. As you monitor replication, you see that the copy queue length initially increases as the logs are replicated from the active copy to the new copy and then decreases after all the current logs have been copied. You then see an increase in the replay queue followed by a reduction to zero as the database copy becomes fully resynchronized and moves into a healthy state.

INSIDE OUT **Where's my database mounted?**

After you begin to create copies of databases within a DAG, it's easy to lose track of where any individual database is mounted. After all, Active Manager could have mounted a database on another server because of a failed Managed Availability probe related to a problem that you didn't notice. The fastest way to check where databases are mounted is to run the Get-MailboxDatabase cmdlet. The important thing is to include the Status parameter because this tells Exchange that you need more than static information. For instance, the following command does the trick:

```
Get-MailboxDatabase -Status | Format-Table Name, MountedOnServer, Mounted
```

Monitoring database copies

On an ongoing basis, you can browse the set of databases with EAC to check that all databases and their copies are in good health. Unlike the Exchange 2010 console, EAC does not display the copy queue length and replay queue length for database copies in a dashboard format. EAC should not attempt to display constantly updated replication activity for a DAG for two obvious reasons. First, the queue lengths are very dynamic and change over time as network and server load flex. To have any chance of being accurate, a console would have to refresh its information about queue length very frequently, say every second or so. This would provide a reasonably accurate snapshot of replication activity at the expense of creating additional load on the servers. In addition, there is a chance that all the servers in a widespread DAG would not be able to respond to the console's poll quickly enough to make the information usable. The second reason is the potential for confusion. Exchange 2010 certainly reported queue lengths, but sometimes the reports were off the chart. The largest value reported for a copy queue length was 9,223,372,036,854,775,766. (As explained in *http://blogs.technet.com/b/timmcmic/*, such a value occurs for the very good reason that a divergence has occurred between the information of log files as known to the cluster and to the server hosting the copy.)

When you look at database copy status through EAC, you see an indication of which servers host a database copy (depending on the screen size of your workstation, you might not see all the server names here) as part of the data reported in the database list and some information about the current copies for the selected database in the details pane. Figure 9-13 shows the situation after a new database copy has been created for the DB2 database. The information in the details pane tells you which copy is active and which copies are passive. In this case, you see that the copy mounted on server ExServer2 is active and that its content index is healthy (meaning that all items in the database have been indexed for search).

Naturally (or confusingly), because this is the active copy, its copy queue length is zero. You can also see that the passive copy on server ExServer1 is healthy.

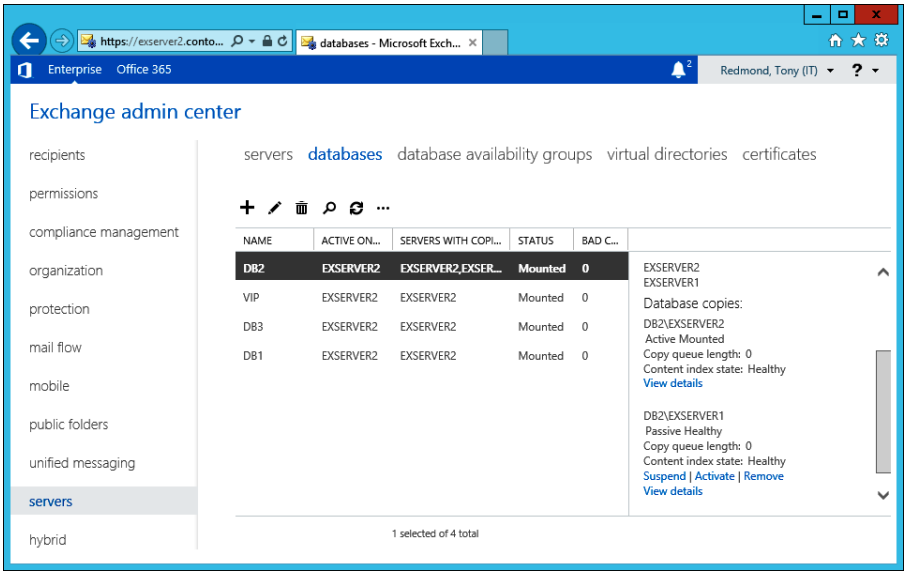


Figure 9-13 How EAC displays information about database copies

Options are available under each database copy to enable you to suspend (replication), activate the copy, or remove the copy. Clicking the View Details link reveals more information about the replication status for the copy (Figure 9-14). Again, you see that the copy queue length is zero, but now you also see that the replay queue length is zero. No errors have been reported. The time stamps for the last logs processed show that the same log was made available on the server hosting the active copy, copied, inspected, and replayed, all of which is the expected state of affairs when the copy and replay queue lengths are zero. The time stamps displayed are taken from values in the transaction log header.

Scrolling through database copies to review information about replication status is an activity best left for a wet Thursday afternoon in March. And, as discussed earlier, the information EAC displays is static and is not updated on an ongoing basis. Thus, the EAC display rapidly goes stale and is not indicative of the current replication status. To get a better and more accurate idea of what's happening with replication, you can either use EMS to report replication status through the Get-MailboxDatabaseCopyStatus command or use the Windows Performance Monitor (PerfMon).

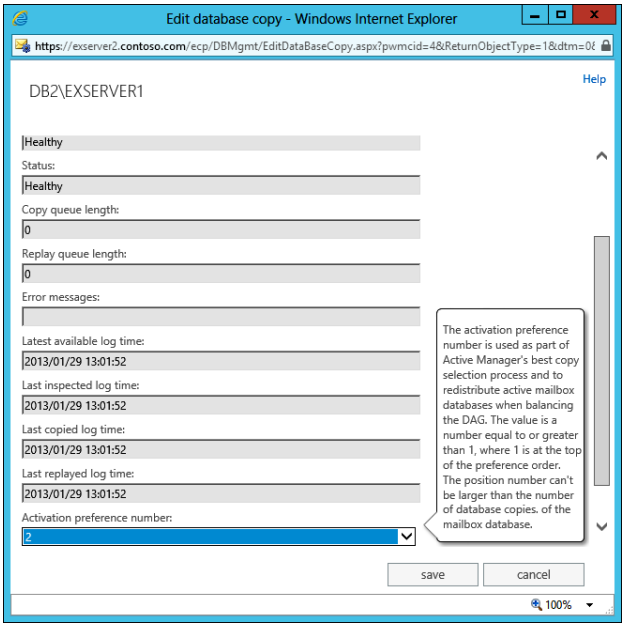


Figure 9-14 Viewing the replication status of a database copy

Running a simple EMS command is the fastest and most efficient method of retrieving replication status for larger DAGs, albeit with the understanding that once again you are fetching a static snapshot of replication activity that is accurate at only that point in time. For example, here's the output returned from scanning for replication status for all Mailbox servers in the DAG named DAG1:

```
Get-MailboxServer | Where {$_.DatabaseAvailabilityGroup -eq "DAG1"} |
Get-MailboxDatabaseCopyStatus
```

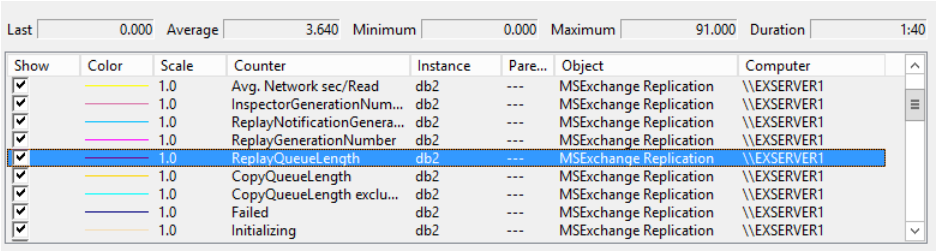
```
Name                    Status    CopyQueue  ReplayQueue    LastInspectedLogTime   ContentIndex
                                               Length              Length              State
---------               --------  ---------  -------------  ---------------------  ------------
DAG1-DB1\ExServer1       Mounted  0          0                                      Healthy
VIP Data\ExServer1       Mounted  0          0                                      Healthy
IT Department\ExServer1  Mounted  0          0                                      Healthy
DAG1-DB3\ExServer1       Healthy  0          0              2/23/2013 5:11:10 AM    Healthy
PR\ExServer1             Healthy  0          0              2/23/2013 5:09:52 AM    Healthy
ITOPS_DB1\ExServer1      Mounted  0          0                                      Healthy
```

An edited set of the database copies in DAG1 is listed here. The copies shown as Mounted in the Status column are those that are currently active. The passive copies are all shown as

Healthy, indicating that replication is progressing normally. The State column indicates that there are no errors with any of the active or passive copies; they all have a status of Healthy.

It's important to emphasize once again that the data Get-MailboxDatabaseCopyStatus returns begins to age immediately and is stale as soon as you look at it. However, it is sufficient to provide a quick and dirty overview of what's happening inside a DAG. If you want information about replication that is always up to date, you have to turn to Performance Monitor. Exchange publishes many counters that help identify and understand performance characteristics for servers, among which is a set that measures replication activity for every database copy. You can review these counters through the Performance Monitor console or provided as a feed to another application such as Microsoft System Center Operations Manager (SCOM).

Figure 9-15 shows some of the Performance Monitor counters that are available, including the replay queue length and copy queue length. You can see that the recent peak for the replay queue was 91 logs, which might prompt a question about why this might have happened. Note that the set of performance counters Exchange 2013 published differs from previous versions, so if you have procedures that depend on performance data for Exchange, they probably need to be reviewed and updated.



Last 0.000 Average 3.640 Minimum 0.000 Maximum 91.000 Duration 1:40

Show	Color	Scale	Counter	Instance	Pare...	Object	Computer
✓		1.0	Avg. Network sec/Read	db2	---	MSExchange Replication	\\EXSERVER1
✓		1.0	InspectorGenerationNum...	db2	---	MSExchange Replication	\\EXSERVER1
✓		1.0	ReplayNotificationGenera...	db2	---	MSExchange Replication	\\EXSERVER1
✓		1.0	ReplayGenerationNumber	db2	---	MSExchange Replication	\\EXSERVER1
✓		1.0	ReplayQueueLength	db2	---	MSExchange Replication	\\EXSERVER1
✓		1.0	CopyQueueLength	db2	---	MSExchange Replication	\\EXSERVER1
✓		1.0	CopyQueueLength exclu...	db2	---	MSExchange Replication	\\EXSERVER1
✓		1.0	Failed	db2	---	MSExchange Replication	\\EXSERVER1
✓		1.0	Initializing	db2	---	MSExchange Replication	\\EXSERVER1

Figure 9-15 Performance Monitor counters for replication activity

Get-MailboxDatabaseCopyStatus returns a lot of interesting information about a database copy. If you look at the output of the command, you discover many details about a copy, most of which EAC never reveals. For example:

- **Status** Hopefully Healthy. A status of ServiceDown indicates a problem with the server or that the Microsoft Exchange Replication service on the server is unable to respond to the request for information. Failed indicates that the copy is unable to replay or copy log files, so it can't be kept updated. Seeding means that the copy is currently being seeded from another copy; Suspended means that an administrator has suspended replication to this copy for some reason. A full list of the valid database copy status codes is at *http://technet.microsoft.com/en-us/library/dd351258.aspx*.

- **LastStatusTransitionTime** The date and time when the database copy last went through a transition, such as the last time it was mounted.
- **ActiveCopy** True if the copy is servicing client connections. False if it is a passive copy.
- **ActivationPreference** The number dictating the preference order for this copy when Active Manager considers how best to failover automatically to a healthy copy.
- **StatusRetrievedTime** The date and time when the command was run.
- **WorkerProcessId** The Windows process identifier for the worker process connected to this database copy.
- **Single Page Restore** The number of single page restore requests that have been made by this database copy. Ideally, this should be zero, but a small number is tolerable as long as the number does not steadily increase because this would indicate that a storage problem is causing pages to be corrupted in the database.
- **CopyQueueLength and ReplayQueueLength** The number of transaction logs available and waiting to be copied and those waiting to be replayed into the database.
- **LastAvailableLogTime** The date and time when a new transaction log became available for the active copy. Other dates include LastCopyNotificationLogTime (the last log notified to the database), LastCopiedLogTime (the last log copied), LastInspectedLogTime (last log inspected), and LastReplayedLogTime (last log replayed into the database). These values will be the same for a database copy that is completely up to date.
- **LastLogGenerated** The generation number of the last transaction log generated by the active copy. Similar to the dates previously discussed, generation numbers are also available for the last log notified (LastLogCopyNotified), copied (LastLogCopied), inspected (LastLogInspected), and replayed (LastLogReplayed). Again, these values will be the same for a database that is completely up to date. The copy queue length and replay queue length reported by a database copy are calculated by comparing the generation numbers of the last log files that have been copied and replayed to the information known about the generation number of the current log file for the active copy.
- **LastLogInfoIsStale** This is normally False, meaning that this database copy is aware of the current situation within the DAG and no danger exists that it could be activated without being aware of transaction logs that exist and need to be copied from the active server. Exchange tracks information about the last log generated for

the active copy (generation and date and time) and stores it in the cluster registry, which is then replicated to DAG members. This information can be compared to the information known to the passive copy to determine whether the information known to the passive copy is stale (a gap of more than 12 minutes exists in the time stamp retrieved from the cluster registry held in LastLastInfoFromClusterTime) and the system time. Active Manager won't attempt to activate a copy that is stale. The last known transaction log generation on the active server is also reported from the cluster registry and is stored in LastLogInfoFromClusterGen.

- **ReplayLagStatus** If the passive copy is lagged, information about the lag time will be recorded here.
- **IncomingLogCopyingNetwork** The DAG network that is used to replicate transaction logs. If a single NIC exists on the server, this will be a MapiDagNetwork; otherwise, it will be a dedicated replication network.
- **Free disk space** The percentage of free disk space is recorded in DiskFreeSpacePercent, whereas the total amount (in GB) is in DiskFreeSpace, and the total size of the disk is in DiskTotalSpace. These properties are interesting if you want to write a report to fetch information about disks holding database copies and highlight those that might need some attention because they are running out of space.

INSIDE OUT Redundancy warning events

Behind the scenes, Managed Availability checks that the redundancy status of databases mounted on DAG member servers is sufficiently protected with copies. If a database is found to have just one copy for more than 15 minutes in an hour, event 4113 is written into the application event log and continues to flag this issue with a new event every 15 minutes until a second copy is available for the database. Event 4114 is written if the database has sufficient copies. You'll see this event logged hourly for databases that Managed Availability deems well protected. Unlike Exchange 2010, these events are not generated by running the CheckDatabaseRedundancy script, which is now obsolete because Managed Availability does the checking. Managed Availability also generates events 4138 and 4139 to report the overall redundancy situation for the databases on a server.

Reseeding a database copy

Incremental reseeds are performed whenever a database copy comes online within a DAG. This procedure determines the set of transaction logs that are necessary to bring the database copy completely up to date. The logs are either already on the server or need to be copied from the server hosting the active copy. Usually, the incremental seed process is successful, and the database copy can begin processing new transaction logs as they arrive.

However, if incremental reseeding is unsuccessful, a more fundamental reseed is necessary. It might be that too many transaction logs are missing to allow Exchange to update and mend the database copy, or it might be that the initial attempt to seed a new database copy encountered problems. I find it best always to create a new database copy on the server that will host the copy; I've experienced some problems creating copies from remote servers. Perhaps it is just luck, or perhaps it is because when you are logged on to a server, you know it is functioning correctly and have the opportunity to take immediate action to rectify any apparent problems before starting the copy.

When you perform a reseed, you select a good database copy on a running server and then force Exchange to copy the database and its transaction logs to the target server to instantiate the database copy on that server. You can reseed from the active copy or from a passive copy. Obviously, reseeding can take some time to accomplish if the database is larger than a few gigabytes, so it's best to select the failed database copy (shown with a Failed status) and first use Resume Database Copy to restart synchronization by copying and replaying transaction logs from the active database to bring the failed copy up to date. If this doesn't work, you will have to reseed, so you should select the Update option. Use the following procedure:

1. Identify any potential reasons the database copy failed and make sure that any underlying causes have been resolved.
2. In EAC, select the database whose passive copy you need to reseed.
3. Find the failed database copy in the details pane and click Suspend to instruct Exchange to stop attempting to replicate transaction logs to this copy. Enter some information about why the suspension is necessary and click Save.
4. Click Update to begin the reseed process. By default, Exchange uses the active database copy as the source. If this isn't what you want to do, click Browse to see the list of servers that host copies and select the database copy to use. Ideally, it is best to select a server that is not under heavy load at the time and is in close network proximity to the target server. In other words, it's probably not a good idea to select a server in New York as the source to reseed a server in San Francisco when a good database copy is available on a server in San Jose.

5. Click Save to invoke the reseed operation. Exchange then copies the database and transaction logs to the target server.

The underlying EMS command to reseed the Sales database on server ExServer4 using ExServer1 as the source server is as follows. The DeleteExistingFiles switch instructs Exchange to remove any database and transaction log files that exist on the target server before it starts the reseed, and the ManualResume switch tells Exchange that it should not restart replication after the reseed is complete. (An administrator will have to resume replication when ready.)

```
Update-MailboxDatabaseCopy -Identity 'Sales\ExServer4' -SourceServer 'ExServer1'
-DeleteExistingFiles  -ManualResume
```

If you attempt to reseed a database within 15 minutes of a failed seed operation, Exchange asks you to confirm that the reseed should proceed. Exchange tracks seeding operations (update requests) to ensure that they don't happen too close together because this might create a scenario in which one reseed operation is not complete when another starts. If you see this prompt, you should validate that everything is okay before proceeding. Exchange clears update requests 15 minutes after they have been made.

Of course, you can avoid some of the work involved in reseeding databases by taking advantage of the new database autoreseed feature in Exchange 2013. This feature uses a designated spare disk to create a new database copy if a database fails. When set up, the process should be automatic, and its intention is to ensure that sufficient copies of a database are maintained to ensure the desired level of redundancy. See *http://technet.microsoft.com/en-us/library/jj619303(v=exchg.150).aspx* for more information on configuring database autoreseed.

INSIDE OUT Reseed failures

Before you reseed a database, ask yourself, "Why did this happen?" Make sure that any underlying issue is resolved before you proceed. The failure that affected the database might mask a problem that could, in time, compromise your ability to maintain a resilient service. There's no point in reseeding a database if that copy will fail soon afterward.

INSIDE OUT **Identifying specific database copies**

Many of the EMS cmdlets you use to work with database copies require you to identify the specific database copy by passing the database name and the server name in the form of database\server. For example, DB1\ExServer2 means the copy of database DB1 that is mounted on server ExServer2.

Adding database copies with EMS

The EMS command to create a new copy of the DB1 database on the ExServer2 server with an activation preference of 2 is shown here. You don't have to select an activation number; Exchange determines it for you by adding one to the current highest value.

```
Add-MailboxDatabaseCopy –Identity 'DB1' –MailboxServer 'ExServer2'
–ActivationPreference '2'
```

If you make a mistake with the activation value, you can adjust it afterward with the Set-MailboxDatabaseCopy cmdlet and check that the correct activation order is in place with the Get-MailboxDatabase cmdlet.

```
Set-MailboxDatabaseCopy –Identity 'DB1\ExServer2' –ActivationPreference '1'
Get-MailboxDatabase -Identity 'DB1' | Select ActivationPreference
```

All the copies of a database share the same path for database files and transaction logs, so you do not have to determine these values on the target server. However, you should make sure that these settings do not create a conflict with existing files that might already be in place on the target server. The "Activation blocks" section later in this chapter offers more details about how to manipulate database locations when databases operate within a DAG.

Using a lagged database copy

DAGs support lagged database copies. A lagged database copy is one that is maintained at a certain point in time behind the active database copy by not replaying transactions immediately when they become available. Instead, the transaction logs are kept until the lagged period elapses and are then replayed. The lagged database copy is therefore kept at a certain remove from the active database and the other nonlagged database copies.

The primary reason to use lagged database copies (7 and 14 days are common intervals) is to provide you with the ability to go back to a point in time. Some organizations use lagged database copies to replace traditional backup cycles. By delaying the replay of replicated transaction logs in a database copy, you always know that that copy represents

a point in the past when the database was in a certain condition. You can also recover information from the database that existed at that point in time, which is often the use for traditional backups.

In an Exchange context, logical corruption can mean two things:

- **Database logical corruption** A database page has a valid checksum, but the data in the page are incorrect. Typically, this occurs when ESE attempts to write a page and receives a Success status from Windows, but the data are never actually written to disk or are written to the wrong place in the database. Microsoft refers to this as a lost flush. Exchange 2013 includes a lost-flush detection mechanism to avoid these problems.

- **Store logical corruption** It is possible for an application to write data in a database in a way that generates items deemed corrupt. Messages migrated from other messaging systems through third-party applications or gateways can sometimes generate corrupt items. If this happens, sufficient corrupt items might be introduced into a database to make a recovery from a lagged copy the only way to restore a database to good health.

Lagged database copies certainly provide a different kind of protection that can be invaluable in the right circumstances. However, I did not consider the implementation in Exchange 2010 to include sufficient automation. Too much potential existed for an administrator to make an error when he attempted to restore a database to a point within the lagged period. Exchange 2013 includes some elements of self-care to make sure that transaction logs for a lagged copy don't run into problems such as disk space exhaustion; if the Replication service detects that the disk is running low on space, it replays some of the transaction logs to free space. This is a good idea when multiple databases exist on the same volume because you do not want to compromise the smooth operation of an active database by running out of space. Making space available to the active database by playing down transaction logs reduces the lagged period for the lagged copy, but it keeps everything going. Replay is also automatic when the Replication service detects that redundancy is compromised by failure in other database copies.

When you recover using a lagged database copy, you essentially roll back the database to a point in time before it experienced a logical corruption problem that has since been replicated to the nonlagged database copies, or you decide to bring the lagged database copy completely up to date because it is now the only viable database copy. This work can take many hours because you have to isolate the database copy, identify the transaction logs that are required to bring it to the point in time before corruption occurred, replay those logs to update the database and bring it to a clean state, and then use it to restart service. For more information on the steps necessary to activate a lagged database copy, see *http://technet.microsoft.com/en-us/library/dd979786.aspx*.

Another point to consider is that a large amount of storage might be required to hold all the transaction logs for the lag period. For example, if the lag period is seven days, the server that hosts the lagged database copy must be able to store all the transaction logs for more than seven days, preferably at least 10. This could be many gigabytes of logs for even a moderately busy database.

> **Note**
>
> **Because of the manual nature of the work and the time required to perform these activities, lagged copies can never be considered a high-availability option for databases. They certainly help and provide a valuable feature, but not high availability.**

If you use a comprehensive backup solution that supports DAGs such as Microsoft System Center Data Protection Manager, that software probably has the capability to restore a database to a particular point in time. This is the purpose of a lagged database copy, so it follows that the deployment of a comprehensive backup solution reduces the need to keep a lagged database copy. You can certainly keep the lagged copy as an extra protection against data loss, but this could be considered overkill because the correct operation of a backup regime should provide you with sufficient protection against operator failure, storage corruption, or any of the other issues that might cause you to resort to using a lagged database copy. Lagged databases are also less interesting if you use storage technology, which supports the ability to take snapshots of databases and transaction logs that you can use for restores. Some of the sophisticated storage technology available today includes features that make a database restore to a particular point in time very easy and fast. If you can restore a database at different points, such as four hours ago, eight hours ago, and so on, you probably don't need to incur the expense and complexity required to maintain a lagged database copy.

Two advanced properties of mailbox databases govern lagged log replay and truncation times. You can use the Add-MailboxDatabaseCopy or Set-MailboxDatabaseCopy cmdlets to set these properties; ReplayLagTime can be configured through EAC.

- **ReplayLagTime** The time (in days) governing the delay Exchange applies to log replays for database copies (replay lag time). Setting this value to zero means that Exchange should replay new transaction logs as soon as they are copied to servers that host database copies. The intention is to give you the chance to keep a server running in a state slightly behind the active copy so that if a problem occurs on the active server that results in database corruption, you can stop replication and prevent the corruption from occurring in database copies. Typically, DAGs that use a lagged copy are configured so that two or three database copies are kept up to date and

one (often in a disaster recovery site) is configured with a time lag. The maximum lag time is 14 days.

- **TruncationLagTime** The time (in days) governing log truncation delay. Again, you can set this value to zero to instruct Exchange to remove transaction logs immediately after their content has been replayed into a database copy, but most sites keep transaction logs around for at least 15 minutes to ensure that they are available if required to bring a database copy up to date if an outage occurs. The maximum truncation lag time is 14 days.

Lagged settings are present for every database copy but used for lagged copies only. This command shows how to view these settings for a database.

```
Get-MailboxDatabase -Identity 'DB1' | Format-List DatabaseCopies, ReplayLagTimes,
TruncationLagTimes
```

```
DatabaseCopies              : {DB1\EXSERVER1, DB1\EXSERVER2}
ReplayLagTimes              : {[EXSERVER1, 00:00:00], [EXSERVER2, 00:00:00]}
TruncationLagTimes          : {[EXSERVER1, 00:00:00], [EXSERVER2, 00:00:00]}
```

Both the replay lag times and truncation lag times on the two servers that maintain copies of the DB1 database are set to 00:00:00 (which is what you expect because these are the default values). These values mean that transaction logs are replayed immediately, and no delay occurs for log truncation. You can change the situation by updating the property of one of the copies. For instance, this command sets a replay lag time of seven days on the mailbox database DB1 on server ExServer2:

```
Set-MailboxDatabaseCopy -Identity 'DB1\ExServer2' -ReplayLagTime 7.0:0:0
```

The output of the Get-MailboxDatabase command now shows:

```
ReplayLagTimes              : {[EXSERVER1, 00:00:00], [EXSERVER2, 7.00:00:00]}
TruncationLagTimes          : {[EXSERVER1, 00:00:00], [EXSERVER2, 00:00:00]}
```

Immediately after you set a lagged time on a database copy, you should notice that the replay queue length for the lagged copy begins to grow. This is quite normal and is due to the accumulation of transaction logs after the imposition of the replay lag time. The replay queue length grows until logs begin to age out past the replay lag time and are replayed into the database. Over time, the replay queue length stabilizes into a range that reflects the average transaction load during the lag period. Depending on the size of the database and the workload it handles, this range could extend well into the tens of thousands of logs. (Be sure that you allocate sufficient storage to accommodate the lagged transaction

logs.) You can monitor the replay queue length with the Get-MailboxDatabaseCopyStatus cmdlet. For example:

```
Get-MailboxDatabaseCopyStatus –Identity 'DB1\ExServer2'
```

If you adjust the lag period for a database copy, the replay queue length moves in line with the adjustment. The queue grows if you increase the lag period and reduces if you decrease the lag period because Exchange begins to replay queued transaction logs that are now inside the lag period.

In most instances, you probably should block the activation of a lagged database copy so that Active Manager doesn't consider the copy to be a viable candidate for activation in case the other copies fail. You can do this with the Suspend-MailboxDatabaseCopy cmdlet. Normally, this cmdlet suspends replication for a copy. In this situation, use the ActivationOnly parameter to tell Exchange that replication can proceed, but you never want to activate the lagged copy automatically. This command applies the block to the copy of the DB1 database on server ExServer2:

```
Suspend-MailboxDatabaseCopy –Identity 'DB1\ExServer2' –ActivationOnly
```

If an event occurs that forces you to make the copy a viable target for activation, you can do this with the Resume-MailboxDatabaseCopy cmdlet:

```
Resume-MailboxDatabaseCopy –Identity 'DB1\ExServer2'
```


INSIDE OUT Lagged copies as a disaster recovery mechanism

Lagged database copies are a disaster recovery mechanism you should implement only as a last resort to recover a good copy of a database when all other routes have failed. Most system administrators don't have sufficient experience in running lagged copies over an extended period and, therefore, probably haven't had the chance to develop the skills to execute data recovery by using lagged copies. In addition, lagged copies cannot provide the truly granular restore capabilities required to put a database back to a specific point in time. You cannot tell Exchange to roll back to "4:23 P.M. on August 27, 2013."

It is a bad idea to discard the warm-blanket feeling that a well-designed and oft-practiced backup regime gives to administrators because they know their data are safe. Knowing that they can recover their data requires a leap of faith that will likely take several more versions of Exchange to achieve. In the meantime, those who want to live on the jagged edge of new techniques to protect data can depend on lagged copies, and the rest of us will use traditional backups for a little while yet.

INSIDE OUT **What's a good lag period?**

You should set the lagged period for a database copy with two criteria in mind. First, the period should be sufficient to provide the desired level of protection against corruption. Second, the period should provide administrators a reasonable amount of time to detect that corruption has occurred. For instance, a two-day lagged period is usually insufficient because it means that if corruption occurs at the start of a three-day holiday weekend, it might not be detected in time to use the lagged copy. Seven days seems like a reasonable period, but you can go all the way to 14 days if you like.

Self-maintained lagging

Two issues in Exchange 2010 made lagged copies less attractive than they would otherwise be. First, because lagged copies accumulate transaction logs for the lag replay period, it was easy to exhaust available space on the disk that held the transaction logs, especially when the mailbox database was heavily used. Second, lagged database copies do not support single page patching, a feature introduced in Exchange 2010 SP1 to give databases running within a DAG the ability to request information from other copies if the Store detected a bad page within a database. Although lagged database copies still do not support single page patching, some of the issues have been addressed in Exchange 2013 in a set of changes that are intended to make lagged database copies more self-maintaining than before. The changes do not remove administrator responsibility; they merely make looking after lagged database copies a little less onerous than before.

The solution in Exchange 2013 is to incorporate intelligence to enable lagged copies to invoke automatic log replay to play down transaction logs in specific scenarios:

- When a low disk space threshold is encountered on the disk holding the transaction logs. Playing down transaction logs relieves pressure on the disk at the expense of removing some element of the protection afforded by the lagged copy.
- When the Store detects that some pages in the lagged database copy are corrupt and need to be patched using the single page patching mechanism. Playing down the transaction logs brings the lagged database copy up to date and enables the copy to request other copies to provide the data necessary to patch the problem pages.
- When there are fewer than three available healthy copies (active or passive) of the database for more than 24 hours.

To preserve backward compatibility with the behavior of Exchange 2010, these features are not enabled as default. If you want to use them, you have to enable lag copy self-maintenance for the DAG by running the Set-DatabaseAvailabilityGroup command. For example:

```
Set-DatabaseAvailabilityGroup -Identity DAG1 -ReplayLagManagerEnabled $True
```

Regretfully, there is no user interface in EAC or property that can be tweaked through EMS to control when a lagged copy begins self-maintenance. You have to update the system registry on every Mailbox server in the DAG if you want to change the default settings. Two values can be changed:

- The number of healthy database copies that must exist with no outage of more than 24 hours before the Store will begin to replay transaction logs into a lagged copy. The default is three. To change the value, update the following registry value:

  HKLM\Software\Microsoft\ExchangeServer\v15\Replay\Parameters\ReplayLagManagerNumAvailableCopies

- The threshold for the percentage of available disk space on the transaction log disk that must be available before the Store begins to replay transaction logs. The default is 10 and is controlled by the following registry value:

  HKLM\Software\Microsoft\ExchangeServer\v15\Replay\Parameters\ReplayLagPlayDownPercentDiskFreeSpace

Lagged database copies and Safety Net

When the Store attempts to recover a lagged database, it consults the queue for that database held in Safety Net to determine whether any messages need to be redelivered to mailboxes. For this reason, it is critical to configure Safety Net so that it holds messages for at least as long as the lagged period. For instance, if you configure a mailbox database to have a lagged copy for five days, it would not make sense to configure Safety Net to hold messages for three days because the risk then exists that two days' worth of messages will be unavailable for replay if the need arises. Instead, the configuration settings for the mailbox database and Safety Net should be configured as follows:

```
Set-MailboxDatabaseCopy -Identity DB2\ExServer3 -ReplayLagTime 5.00:00:00
Set-TransportConfig -SafetyNetHoldTime 5.00:00:00
```

Now assume that a problem occurs and all the other copies of a database have been affected. The only recovery option is to use the lagged copy and update it to a point before the problem happened, for instance 36 hours ago. In this case, the steps to recovery are:

1. Suspend replication to the lagged database copy with the Suspend-MailboxDatabaseCopy cmdlet:

   ```
   Suspend-MailboxDatabaseCopy -Identity DB2\ExServer3 -Comment "Recovery due to
   virus outbreak"
   ```

2. Dismount the lagged database copy.

3. Make a copy of the lagged database copy and its transaction logs so that you can always go back to the current state. It is recommended to make two copies so that you have an additional copy to work with, just in case. There is no harm in making the second copy if you have the available storage.

4. Select a copy to work with.

5. Discard all the transaction logs for the selected copy except for those necessary to bring the database up to date to the desired time. For example, if you want to restore the database as at 1 P.M. on 5 May 2013, keep all the transaction logs up to this point and delete all the logs thereafter. Failure to remove logs creates the risk that they will be replayed back into the database and so reintroduce the corruption or other problem that caused you to start the recovery process.

6. Delete all the files in the database's location and copy over the .edb and transaction logs you want to use for recovery.

7. Mount the copy. At this point, this copy becomes the active copy, and Active Manager begins its normal process to determine whether any recovery steps are required.

   Because the database copy is 36 hours behind, Active Manager requests the transport system to redeliver any message that has arrived in the past 36 hours (the lag period). These messages will come from the Safety Net queue for the mailbox database. Depending on the topology within your organization, the messages might have originated from multiple Mailbox servers, each of which is then involved in the recovery process. In this scenario, fetching the messages and the subsequent replay into the database might take some time.

8. Log on to a mailbox in the database and check that the database contains all the expected information.

9. Begin replication as normal to bring the other database copies up to date.

It all sounds easy! However, like any other operation that involves database copies, you should attempt this in a test environment first. Carefully document all the necessary steps, including details specific to your systems, and make sure that the procedure to recover from a lagged copy is as clear as possible for those who might have to undertake the work in what is inevitably a pressure situation.

Activating a mailbox database copy

An administrative move of an active mailbox database so that a passive copy becomes active is called a *database switchover* or activation. It is usually performed when an administrator wants to redirect clients to a copy located on a different server to allow the original server to be taken offline for maintenance. To activate a database copy on another server, open Databases in the Servers section of EAC, select the mailbox database with which you want to work, and then select Activate for the preferred copy in the details pane. EAC then requests confirmation (Figure 9-16) before activating the selected copy. EAC does not flag an issue if you select the copy with the lowest activation number because an assumption is made that you know what you're doing when you make an explicit selection of the copy to become active.

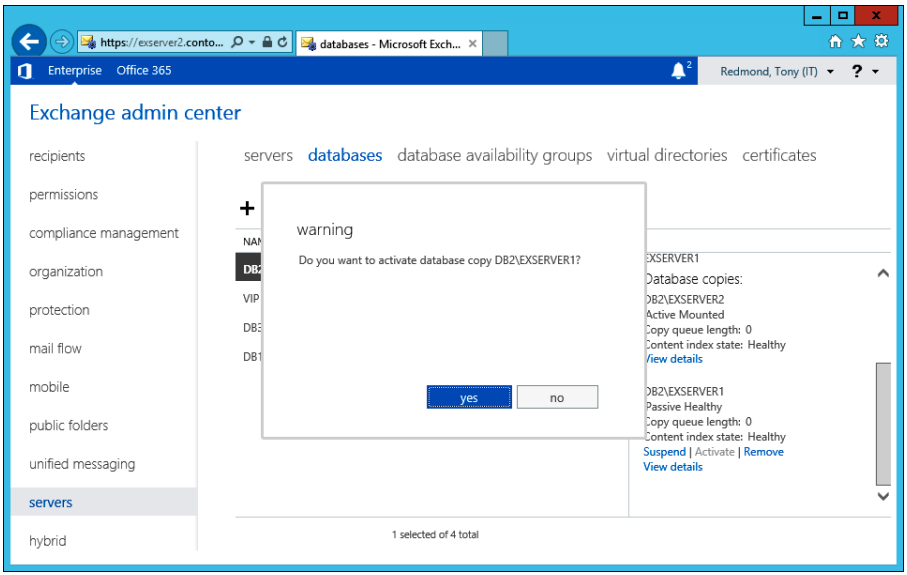


Figure 9-16 Activating a database copy with EAC

Assuming that the database copy is healthy and that any transaction logs required to bring the copy up to date are available, the Store on the selected server brings the database online and begins to accept client connections. This process happens very quickly because all Exchange has to do is switch internal pointers to indicate that a different database copy is now active. No files have to be moved, copied, or otherwise manipulated in a way that would slow things down. After the selected copy is activated, the other copies briefly resynchronize to take account of the new situation, and normal operation resumes.

Exchange uses cache warming to make sure that the newly activated database copy is not in a cold state because that would slow down responsiveness to clients. The Store on the server that hosts the newly activated database has been replaying transactions as logs arrive and maintains a cache of recently used pages. Cache warming means that the Store retains the cache created through transaction replay instead of discarding it and rebuilding the cache with pages requested by clients. The logic here is that clients are likely to want to use pages that exist in the cache because those pages belong to recent transactions that clients might want to access after they reconnect to the database.

Following a successful database mount, the Store requests Safety Net to redeliver any messages that were in transit and exist in the Recoverable Items folder. When the CAS servers handling incoming client connections consult Active Manager to find out which database is active, they learn of the transition and so route connections to the right place.

Immediately after a switchover, you might see Exchange report an increase in the copy queue length for the database copy that has just become active, followed by a rapid decrease to zero. Part of the switchover process is a pass through the transaction log set to validate the logs that are present. The pass completes quickly, and the high copy queue length is not a problem as long as it does not persist past a minute or so.

When the fault is repaired on the original server and the original server comes back online, its copy of the database is passive and is obviously behind the other copies. The Store on that server runs through a divergence detection process and then performs an incremental reseed to bring its database up to date. The first step is to determine the divergence point, which is done by comparing the transaction logs on the server with the logs on a server that hosts a copy that is up to date. The Store can work out which database pages have changed after the divergence point and then request copies of the changed pages from a copy on another server. These pages are replayed until the failed copy is synchronized with the other copies. The goal is for this work to complete and restore service to users within 30 seconds.

In some cases, it might not be possible to perform an incremental reseed because the database is so different from the other copies, perhaps because it was offline for an extended period. In these situations, Exchange informs you that a full reseed is necessary to create an up-to-date copy of the database.

Rebalancing database copies

Of course, administrators are not required to perform failovers; Exchange does this automatically if a server or disk goes offline and affects an active database. As explained earlier, in the "Best copy and server selection" section, Active Manager considers the activation preference order for a database when it decides on which server it should activate a copy, specifically when lossless transitions are required. It's important to realize that a higher

activation preference does not guarantee activation of one copy over another. It is simply a human-expressed preference that Active Manager takes into account as part of the process it goes through to select the best possible passive copy to activate.

Activation preference numbers are assigned when a database copy is created and can be modified afterward with the Set-MailboxDatabase cmdlet. To see the full list of databases and their activation preferences, use a command like this:

```
Get-MailboxDatabase | Select Name, ActivationPreference, Server | Sort Name |
Format-Table -AutoSize
```

```
Name              ActivationPreference                                Server
---------         --------------------                                ------
Dublin Users      {[ExServer3, 1], [ExServer4, 2]}                    ExServer3
DB1               {[ExServer1, 1], [ExServer2, 2]}                    ExServer1
DB2               {[ExServer2, 1], [ExServer3, 2]}                    ExServer2
DB3               {[ExServer3, 1], [ExServer1, 2], [ExServer2, 3]}    ExServer1
IT Department     {[ExServer1, 1], [ExServer2, 2], [ExServer3, 3]}    ExServer1
Operations        {[ExServer4, 1], [ExServer3, 2]}                    ExServer4
PR                {[ExServer1, 1], [ExServer4, 2]}                    ExServer1
Sales             {[ExServer4, 1], [ExServer3, 2]}                    ExServer3
VIP Data          {[ExServer1, 1], [ExServer2, 2]}                    ExServer1
```

This output only tells you what activation numbers have been assigned and the server that currently hosts the database. You can see that two databases are not running on their preferred server: DB3 is active on ExServer1, and Sales is active on ExServer3. There doesn't seem to be a reason for them not to be on their preferred server; both servers are hosting other databases, so this might be a pointer to check out why the situation exists and either adjust the activation preference to reflect the current situation or move the databases around so that they are on their preferred server. You can also see that the ExServer1 server is hosting five of the nine databases in a four-server DAG. This is an abnormal disruption of load that deserves further investigation and rebalancing.

An unbalanced DAG is not especially serious because the situation could occur after a period of perfectly normal transitions due to planned server maintenance or other events. It is also perfectly feasible for an unbalanced DAG to deliver the necessary mix of high availability and performance that you'd anticipate from a perfectly distributed DAG. However, to help manage the distribution of databases within the DAG, Microsoft provides a RedistributeActiveDatabases.ps1 script (in the default scripts directory). The script operates in two modes:

- **BalanceDbsByActivationPreference** This mode attempts to move databases to the most preferred copy without attempting to balance databases across Active Directory sites. This mode is likely to be the most commonly used.

- **BalanceDbsBySiteAndActivationPreference** This mode also attempts to move databases to the most preferred copy while taking account of Active Directory sites. For example, if databases in a DAG are spread across three sites, the script attempts to activate a third of the total databases on servers in each site. This mode is obviously more complex because of the multisite aspect and should be carefully tested before you use it in production.

For example, to rebalance databases by activation preference for the DAG called DAG1, you would type this command. The ShowFinalDatabaseDistribution parameter instructs the script to report on the final disposition of the databases:

```
RedistributeActiveDatabases.ps1 –DAGName DAG1 –BalanceDbsByActivationPreference
-ShowFinalDatabaseDistribution
```


INSIDE OUT **The case of the moving databases**

Managed Availability works closely with Active Manager to ensure that active database copies are positioned on the healthiest available server. This means that if Managed Availability detects a problem with the server that currently hosts a database, it will ask Active Manager whether a healthier server exists. If such a server is available, Active Manager will perform a switchover and activate the database copy on that server. Exchange 2013 is working as designed, but the automatic nature of the transition might come as a surprise to some administrators. Over time, it's also possible that active database copies might be distributed unevenly across DAG members. For this reason, it's wise to keep close tabs on the current state of database copies. You can do this by running the Get-MailboxDatabaseCopyStatus cmdlet on a regular basis. Alternatively, consider using a script such as Get-DAGHealth.ps1, available from *http://exchangeserverpro.com/get-daghealth-ps1-database-availability-group-health-check-script/*. Remember that you can use the RedistributeActiveDatabases.ps1 script to redistribute databases across the DAG if necessary.

Performing a server switchover

It would be nice if you never had to deal with a server whose health is failing and therefore needs some work, but life, software, and hardware don't quite cooperate to make this possible. The problem could be transient and due to something like a badly seated hardware component, or a part could be showing signs of failure, and intervention is necessary before the server comes to a crashing halt. The combination of Managed Availability and Active Manager can detect problems and then move all the databases from a server that doesn't exhibit the usual signs of system health without requiring administrator

intervention to keep Exchange running. Of course, this assumes that the databases can be activated elsewhere. Alternatively, an administrator can put a Mailbox server into maintenance mode (see the "Server maintenance" section later in this chapter) to prepare for the installation of software updates or other upgrades. Putting a DAG member into maintenance mode includes steps to transition active databases off the server before maintenance begins.

If you notice that something isn't quite right about how a server is performing, you can switch its workload to another DAG member. It's clearly best if a server doesn't fail abruptly. It is much better to handle the situation by gracefully transitioning the workload elsewhere, which is the intention behind server switchover. The option to switch over the workload from a complete server is found under the Servers section of EAC (Figure 9-17) rather than under Database Availability Groups. The option appears in the details pane, but only for DAG members. Obviously, you cannot switch the workload from a standalone server.

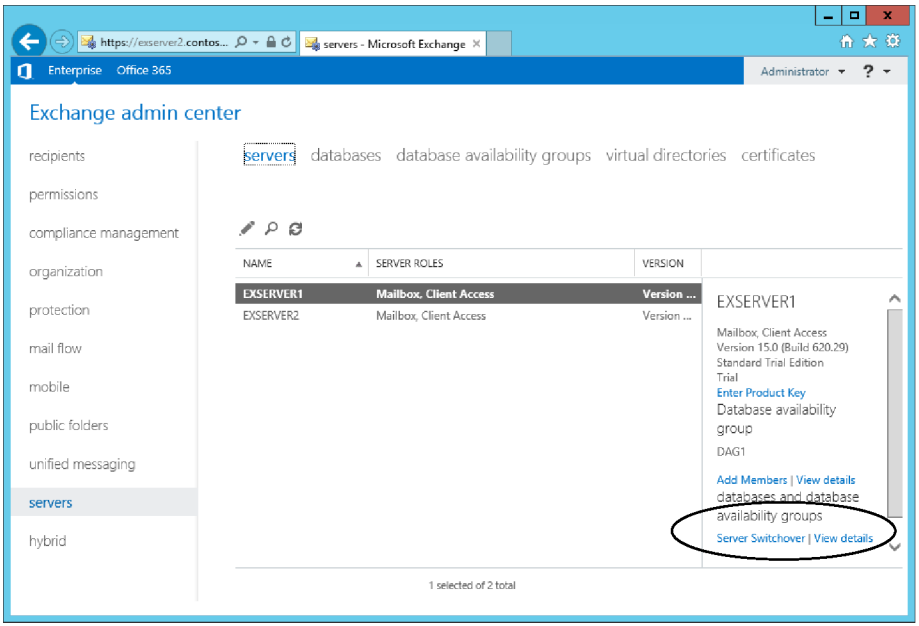


Figure 9-17 The option to switch over workload from a complete server

The options available for server switchover are to enable Exchange to select a target server from the other members of the DAG or for the administrator to browse and select from a list of available servers, which must be online, be healthy, and host copies of all the databases that are currently active on the source server. For example, Exchange cannot activate a copy whose content index is in a Failed state.

After the selection is saved, Exchange runs the Move-ActiveMailboxDatabase command to execute the option (Figure 9-18). If the command is successful, all the databases with copies that are currently active on the source server will be activated on the target. Databases that do not have copies cannot be transitioned.

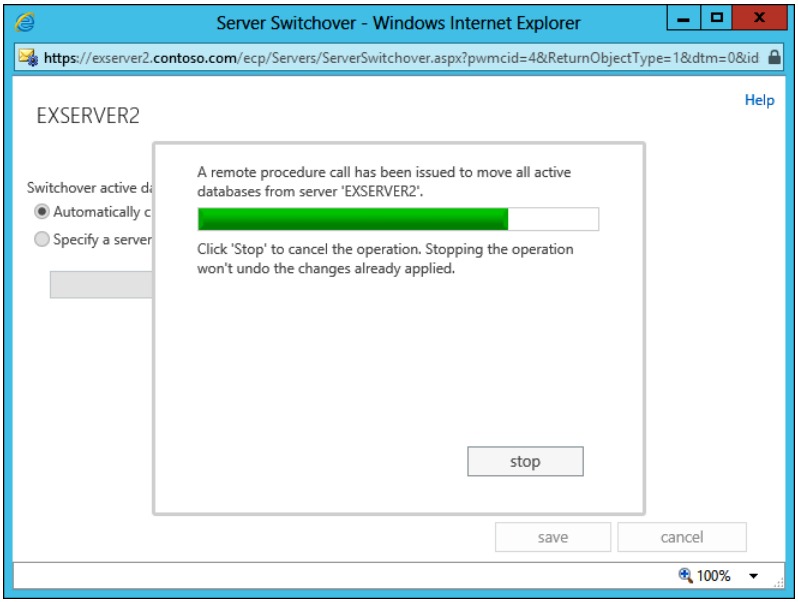


Figure 9-18 Selecting an option for server switchover

You cannot choose databases for activation on different servers. If you want to redistribute the load across multiple servers, you have to do so by selecting each database and determining which copy to activate.

If you don't want to use EAC, you can run the Move-ActiveMailboxDatabase cmdlet to move all the databases on a server. For example, this command tells Exchange to switch over all databases from server ExServer2. In this scenario, called a *targetless switchover*, Active Manager selects the most appropriate database copies to activate based on the copies available in the DAG:

```
Move-ActiveMailboxDatabase -Server 'ExServer2'
```

After the databases are activated elsewhere, you should block Active Manager from attempting to activate any database on the server. In this case, you block activation for server ExServer1:

```
Set-MailboxServer -Identity ExServer1 -DatabaseCopyAutoActivationPolicy Blocked
```

More control over the switchover can be exerted by including the ActivateOnServer parameter to specify the server on which the databases should be activated and the MountDialOverride parameter to determine what level of transaction log loss will be tolerated during the move. For instance, this command moves all active databases from ExServer1 to ExServer2 and specifies that no data loss is acceptable (something that you'd expect if both servers are up and healthy):

```
Move-ActiveMailboxDatabase –Server ExServer1 -ActivateOnServer ExServer2
-MountDialOverride:LossLess
```

Be aware that moving to activate multiple databases on a server might affect the performance characteristics of that server and result in some degradation of responsiveness to clients. In other words, make sure that the target server has sufficient capacity to accept the load before it is transferred. If you have blocked Active Manager from activating databases on a server, remember to remove the block when it is ready to resume full activity.

AutoDatabaseMountDial and potential issues moving databases

You encountered the AutoDatabaseMountDial setting for a mailbox database earlier, in the discussion about how the ACLL process works when Active Manager performs an automatic database failover. The AutoDatabaseMountDial setting tells Exchange how much, if any, data loss is acceptable following a database transition. The setting is based on the copy queue length or the number of transaction logs that have not yet been copied from the server that hosted the previously active copy.

It is hoped that all the database copies are fully synchronized with the active database and that the copy you want to bring online is fully up to date, so the default override setting for automatic database mount on the target server is None, meaning that the Replication service will apply whatever the AutoDatabaseMountDial setting is on that server. However, these are ideal circumstances; some condition might have occurred to prevent full synchronization of the database copy on the target server. For example, a network outage might have prevented the most recent set of transaction logs from being copied to the target server. In situations like this, you can instruct the Replication service to override the AutoDatabaseMountDial setting for the database to decide whether it can mount the database. AutoDatabaseMountDial is set with the Set-MailboxServer cmdlet, and the same value applies to all databases hosted by the server. For example, to set the value so that Exchange will not automatically activate databases on a server unless all transaction logs are present, run the following command:

```
Set-MailboxServer –Identity 'ExServer2' –AutoDatabaseMountDial 'LossLess'
```

The default value for the AutoDatabaseMountDial setting on a server is GoodAvailability, meaning that a database can be automatically mounted if the copy queue length is less

than or equal to six transaction logs. The other values are BestAvailability, meaning that the database will mount if the copy queue length is less than or equal to 12, and Lossless, meaning that the database will only mount if all logs generated on the active server have been replicated to the server hosting the passive copy. Lossless clearly ensures that no data are lost following a failover because all logs are available. Varying degrees of data loss could be encountered with the other settings because up to 12 MB of transaction log data might be missing. It's difficult to quantify what this loss means in practical terms because it depends on the transactions that exist in the missing logs.

Be careful about seeking the elusive goal of lossless failovers. Even in the best possible circumstances, it's likely that one or two transaction logs will be lost during a transition. For example, Figure 9-19 shows the crimson channel report to let you know that ACLL determined that a database lost one log during a transition. In most cases, the loss of 1 MB of transaction log data is inconsequential because the loss will be recovered when Exchange resubmits messages from Safety Net.

An administrator can mount a database that Active Manager cannot mount automatically because its AutoDatabaseMountDial requirement cannot be met or for another reason. For example, the content index of a database copy might be incomplete so that Active Manager ignores it. An unqualified attempt to activate this copy will fail because the catalog is incomplete for some reason. This is by design to ensure that clients can continue search operations on the newly activated database.

You can force Exchange to ignore the search catalog by including the SkipClientExperienceChecks parameter with Move-ActiveMailboxDatabase. This seems like an odd name for a parameter. It's all to do with the Microsoft view that the lack of a content index degrades the client experience. Ignoring the search catalog is quite safe and will not affect users unduly until they attempt online searches and find that the catalog is unavailable or partially complete. The effect of an incomplete catalog tends to be felt by Outlook Web App, Outlook for Mac, and mobile clients, rather than by Outlook because when you configure Outlook in cached Exchange mode, it performs searches of the primary mailbox on the client rather than the server. Therefore, the impact is only felt for searches of archive mailboxes. For example, here's how you could move database DB1 to server ExServer2 even if the catalog is unhealthy:

```
Move-ActiveMailboxDatabase -Identity 'DB1' -ActivateOnServer 'ExServer2'
-MountDialOverride:BestEffort -SkipClientExperienceChecks  -MoveComment 'Moved to
ExServer2 because of problems with disk controller'
```

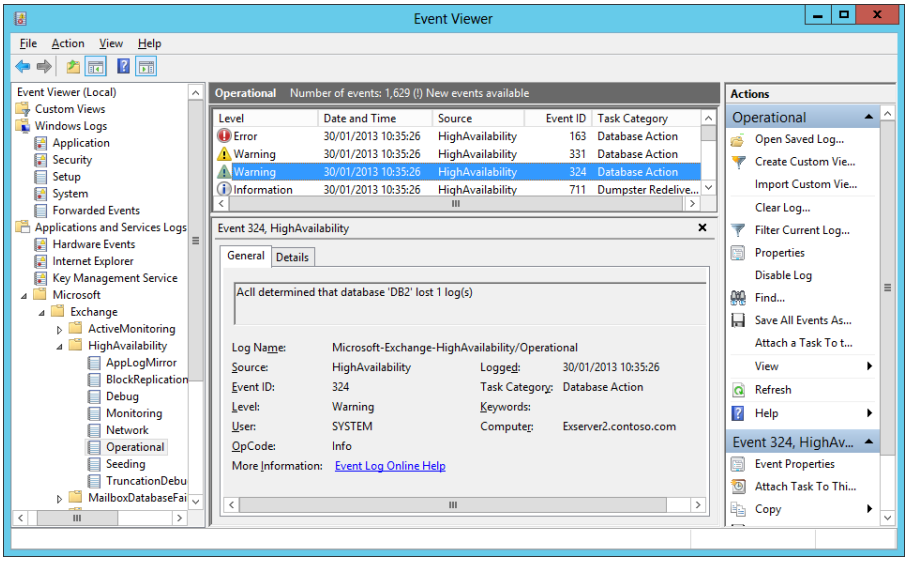


Figure 9-19 ACLL reporting that a single transaction log is lost during transition

Other switches for the Move-ActiveMailboxDatabase command are as follows:

- **SkipActiveCopyChecks** This forces Exchange to activate a database copy even if the currently active copy is being used as the source to seed a new copy. In this case, the seed will halt and have to be restarted.
- **SkipHealthChecks** Normally, Exchange won't allow you to activate a database copy that is in the Failed state. If you want to ignore the possibly poor condition of a database copy, you can activate it by specifying this switch.
- **SkipLagChecks** This enables a database copy to be activated even if its current state is outside the lagged conditions specified for the copy. In other words, it might not be quite up to date.

Another thing to observe in the preceding command is the use of BestEffort as the value to override the AutoDatabaseMountDial settings. This value means that Exchange can mount the database with any value for the copy queue length. In other words, you don't care how many transaction logs might not have been copied because all you want is to get the database online. You might use this override setting when a catastrophic outage has occurred in one data center that renders that data center inaccessible, and you want to activate databases in a secondary data center.

When you are forced to activate a database copy with a failed catalog, you can fix the problem by updating the catalog by reseeding it from a healthy copy. (The ContentIndexState property for the database copy is reported as FailedAndSuspended by Get-MailboxDatabaseCopyStatus—a more serious condition than a simple Failed because Exchange can often bring a failed content index back to a healthy state over time.) In this example, you reseed the catalog for the copy of database DB2 on server ExServer1 by using the passive copy on server ExServer3:

```
Update-MailboxDatabaseCopy -Identity 'DB2\ExServer1' -CatalogOnly -SourceServer
ExServer3
```

If no pressing need exists to bring a database copy with a failed catalog online, you can wait to see whether Exchange can restore the catalog index to good health and then activate the database. The problem here is that activations are often forced by operational necessity, meaning that it's more important to have mailboxes online and available to users than to wait until everything is perfect, so you might not have the time to reseed the catalog before you are forced to bring the database online.

CAUTION!

Be careful when using the BestEffort override; you are almost certain to lose data. Because of the potential for data loss and to acknowledge its intended use as a one-time attempt to bring a database copy online, Microsoft doesn't allow you to set AutoDatabaseMountDial to BestEffort with the Set-MailboxServer cmdlet.

Activation blocks

Sometimes you might want to block a database copy from being activated as the result of an outage. For example, you probably don't want Active Manager to activate a copy on a server when you are upgrading or applying an update to Exchange. To stop automatic activations on a server, use the Set-MailboxServer cmdlet to set the DatabaseCopyAutoActivationPolicy property to Blocked. For example:

```
Set-MailboxServer -Identity 'ExServer1' -DatabaseCopyAutoActivationPolicy 'Blocked'
```

To restore the automatic activation setting, use something like this:

```
Set-MailboxServer -Identity 'ExServer1' -DatabaseCopyAutoActivationPolicy
'Unrestricted'
```

Some companies like to deploy a special server that receives copies of transaction logs to replay into a database copy but are never actually used in production. The server might be under-configured in terms of hardware but can support tasks such as offloading of backup

operations from active servers, recovery of data from backups, legal discovery searches, and so on. A similar technique was used with Exchange Server 2003 and Exchange Server 2007 in multinode clusters in which one node was dedicated to backups and other maintenance operations.

If you deploy a server for this purpose, you should use a server activation policy to block activation because the server will be unable to handle the load imposed by a database transition. If you need to make a database copy active on the server, you can amend the server activation policy or activate the database copy explicitly with the Move-ActiveMailboxDatabase cmdlet.

You can also set a maximum value for the number of databases Exchange is allowed to activate on a server. This is important when you want to be sure that a server will not be swamped with demand for CPU, storage, or memory resources that it cannot handle. For example, you might have a relatively small server within a DAG that hosts the databases for VIP users. Assume that the server hosts five active databases and five passive databases and has the capacity to handle the load generated by eight active databases. You can use the Set-MailboxServer cmdlet to declare this limit to Active Manager as follows:

```
Set-MailboxServer –Identity 'ExServer4'  –MaximumActiveDatabases 8
```

With this value in place, Active Manager can activate three of the five passive databases following outages on other servers. For more information about how the Information Store assigns memory based on the number of active databases on a server, see the "Workers, controller, and memory" section in Chapter 8, "The Exchange 2013 Store."

The same approach can be useful to ensure that activation is balanced across multiple servers in a DAG. It's usual practice to balance the number of active (and passive) databases across all available servers in the DAG, and you can determine the value for maximum databases so that load should be evenly distributed after an outage and no server is swamped by additional load.

Finally, it's also important to understand that limiting the number of databases a server can support might cause Active Manager to ignore a server when considering where to activate a database. Suppose a database has three copies. The copy on the active server fails because of a disk outage and causes Active Manager to look for the best choice passive database copy and then mount this copy to restore service. However, if the server that holds that copy is constrained in terms of the number of active databases, it will force Active Manager to go to the next healthy copy. Assuming that no limitation exists, the copy will be mounted. All this just goes to show that you need to think through activation scenarios before you apply database limitations to servers.

INSIDE OUT **Preventing performance problems**

Every mounted database requires some system resources, and this fact has to be taken into account when you consider the scenarios that occur during the lifetime of a DAG. You might not want Active Manager to activate databases in a way that overloads a particular server simply because it doesn't have sufficient memory to support so many active databases in a way that doesn't impinge on responsiveness to clients. It therefore makes sense to consider what workload a server can take on while still performing well and then use that information to set a limit for the number of active databases a server will support. For example, set a limit of 10 active databases. Passive database copies don't count toward this limit.

```
Set-MailboxServer –Identity 'ExServer1' –MaximumActiveDatabases 10
```

If an administrator attempts to mount a database that exceeds the limit set on a server, the Store refuses to mount the database and issues an ecTooManyMountedDatabases error in the Application Event Log.

Moving database locations within a DAG

It is possible that you won't pick the optimal path for database files and transaction logs when you first create a database, especially for the default database that is created when you install Exchange on a new Mailbox server. Because all copies of a given database share the same path for database files and transaction logs, it's important to relocate the database to the best location before you begin to create additional copies in the DAG. This means that the paths must be valid on any DAG member server that will host a database copy.

EAC provides no method to relocate a database. Moving a database to another location is an operation that requires a little planning because it requires a multistep approach executed through EMS. The following are the basic steps in the operation:

1. Record the full set of properties for the database you want to move with the Get-MailboxDatabase cmdlet.
2. Turn off circular logging. (This step assumes that you follow the Microsoft advice to use circular logging for databases protected by multiple DAG copies.) This step ensures that all the transaction logs are maintained for the active database during the time when its copies are not available within the DAG.
3. Remove the passive database copies from the other servers in the DAG with the Remove-MailboxDatabaseCopy cmdlet.

4. Use the Move-DatabasePath cmdlet to change the database's path and copy the database files to the new location. The database will be dismounted and then remounted as part of this operation.

5. Create the equivalent location on the servers that will host the database copies.

6. Move the database files on the other servers to the new location and remount the copies. Perform this operation on the server where the database is currently mounted.

7. Add the database copies back to the DAG with the Add-MailboxDatabaseCopy cmdlet. Remember to set the correct activation order for the copies as you add them.

8. Validate database properties and make adjustments as required. For example, you might need to change the lag replay time or truncation lag time on a database copy.

9. Turn circular logging back on if required.

10. Use the Get-MailboxDatabaseCopyStatus and Test-ReplicationHealth cmdlets to validate that replication is proceeding normally within the DAG. Unless the database copies have been offline for an extended period during which the active database was very busy, you should not need to reseed the database copies because normal replication processing will perform an incremental reseed to bring the passive databases up to date.

An alternative and simpler approach that has been used effectively and involves a lot less effort is as follows:

1. Suspend all the database copies on the DAG nodes that host a copy by running the Suspend-MailboxDatabaseCopy cmdlet. For example:

```
Suspend-MailboxDatabaseCopy -Identity 'DB2\ExServer1' -SuspendComment 'Move
database file locations'
```

2. Dismount the active database.

```
Dismount-Database –Identity DB2
```

3. Create the new location (database and/or transaction logs) on the server that hosts the database and the servers that host the copies.

4. Copy the files to the new locations on all servers.

5. Move the database files by using the Move-DatabasePath cmdlet, specifying the ConfigurationOnly switch so that Exchange updates Active Directory with only the new location data. You will be informed that this operation "skips the safety check" and prompted to confirm that it should proceed. For example:

```
Move-DatabasePath -Identity 'DB2' -EDBFilePath 'G:\Databases\DB2\DB2.EDB'
-LogFolderPath 'G:\Databases\DB2' -ConfigurationOnly
```

6. Verify that the file locations have been updated in Active Directory.

```
Get-MailboxDatabase -Identity 'DB2' | Format-List Name, *Path*
```

7. Mount the active copy of the database. When this is done, you can resume replication to allow the database copies to resynchronize. Use the Resume-MailboxDatabaseCopy command to restart replication:

```
Mount-Database -Identity DB2
Resume-MailboxDatabaseCopy -Identity 'DB2\ExServer1'
```

8. Check that email is being delivered to mailboxes in the database.

9. Check that logs are being created and copied in the new location and that replication queues are not building to ensure that replication is proceeding normally and that the state of all the database copies is Healthy. View the properties of the database copy through EAC or run the Get-MailboxDatabaseCopyStatus command.

```
Get-MailboxDatabaseCopyStatus -Identity 'DB2\ExServer1'
```

10. Remove the database and transaction logs from the old location on each server.

Because every organization will be slightly different in its implementation of a DAG, it is strongly suggested that you test these procedures thoroughly before you attempt to use one of them in your production environment.

Removing database copies

Adding new database copies is all very well, but sometimes you need to reverse course and remove a copy. You cannot remove the active copy of the database until all passive copies are removed, and you cannot remove the last copy of a database until all the mailboxes it contains have been either moved or deleted.

If you need to remove a database copy that is currently active, you have to force a transition to another copy, wait for the switchover to be successful, and then remove the now-passive database copy. To remove a complete set of database copies, you must delete each passive copy until only the active copy remains; then you can use the normal Remove Database option to delete it.

To remove a database copy, first check that replication is healthy and no network issues exist within the DAG. This is to ensure that all the servers that host copies are aware of the update you are about to make. Second, because it needs to be sure that no potential exists for data loss, Exchange cannot remove the database until circular logging is disabled if the removal will force a transition from multiple copies to a single copy. As explained earlier,

this is because Exchange uses a different form of circular logging when multiple copies exist for a database. When you move a database from two copies to a single copy, Exchange transitions between the two forms of circular logging. It cannot do this until circular logging is disabled and will signal an error if this condition is met. To solve the problem, you have to disable circular logging, dismount and mount the database, and let replication between the copies settle down before attempting the operation again. When everything is ready, select the passive database you want to remove from the list of copies, right-click, and select Remove from the menu in the details pane.

To confirm that the database copy has been removed from the Exchange configuration in Active Directory, you can run the Get-MailboxDatabase cmdlet to return the list of servers and database copies that Exchange knows about as follows:

```
Get-MailboxDatabase –Identity DB2 | Select Servers, DatabaseCopies
```

To complete the job, you need to delete the database files from the server that supported the now-removed copy. Make sure that you select the correct set of files on the right server!

If you want to remove a database completely, you have to remove all the database copies. Naturally, before you remove the last copy of a database, you need to transfer all the mailboxes it hosts to another database. This is extremely important for mailboxes that are used for Exchange features such as arbitration, search, and monitoring mailboxes.

Before removing the database, you can use the Get-MailboxStatistics or Get-Mailbox cmdlets to check that no mailboxes are present. For example:

```
Get-MailboxStatistics –Database 'DB4'
```

Or, to find whether any arbitration and health mailboxes exist in a database:

```
Get-Mailbox –Database 'DB4' –Arbitration
Get-Mailbox –Database 'DB4' –Monitoring
```

All the mailboxes in the database are listed, and you can see that the only mailbox that is present is the default system mailbox, so you know that it is quite safe to remove the database. You can do this with EAC by selecting the database and then clicking the Remove option in the details pane. Click OK to confirm that the removal should proceed, and Exchange will remove the database object from Active Directory.

The EMS commands to remove a database copy are as follows. First, perform the same task as you did with EAC and remove the database copy for DB2 from server ExServer1. The command is:

```
Remove-MailboxDatabaseCopy –Identity 'DB2\ExServer1'
```

After all but the last database copy are removed, you can run the Remove-MailboxDatabase command to remove the database from the organization:

```
Remove-MailboxDatabase -Identity 'DB2'
```

Removing servers from a DAG

Before you can remove a server from a DAG, you must remove any database copies from the server by following the process discussed in the preceding section. You can check for the presence of database copies with the Get-MailboxDatabase cmdlet. In this case, check the mailbox databases that exist on the ExServer1 server:

```
Get-MailboxDatabase -Server ExServer1
```

```
Name                    Server          Recovery        ReplicationType
--------                ---------       ----------      ---------------
Sales                   ExServer1       False           None
VIP Data                ExServer1       False           Remote
```

The output reveals that you have one copy of a replicated database (VIP Data) that is currently mounted on server ExServer1. The first step is to remove the copy of this database from ExServer1. You can leave the nonreplicated database (Sales) intact because it does not affect the server's membership in the DAG.

When you are sure that all replicated databases are removed, select Manage DAG Membership in EAC or use the Remove-DatabaseAvailabilityGroupServer command to evict the server from the DAG. For example, to remove the server called ExServer1 from the DAG named DAG1:

```
Remove-DatabaseAvailabilityGroupServer -Identity 'DAG1' -MailboxServer 'ExServer1'
```

INSIDE OUT **To remove a server, reverse the process of adding a server**

The process of removing a server from a DAG essentially reverses the order of the steps performed to add a server to the DAG. The server is evicted from the cluster, the quorum model is adjusted to take account of the reduced number of cluster members, the server is removed from the DAG object in Active Directory, and the information about the databases hosted by the server is removed from the quorum database. Fortunately, Exchange manages these steps automatically so you don't have to do them yourself.

Sometimes you need to remove a server when it is physically unavailable or cannot be brought online for some reason, perhaps following a hardware outage. You can leave the server in the DAG, but this might affect the ability of the cluster to maintain a quorum. Because the server cannot be brought online, you cannot run Remove-DatabaseAvailabilityGroupServer in the normal manner. Instead, you can use the cmdlet just to remove all traces of the server from the DAG configuration data held in Active Directory. The ConfigurationOnly parameter does the trick. For example:

```
Remove-DatabaseAvailabilityGroupServer -Identity 'DAG1' -MailboxServer 'Exserver2'
-ConfigurationOnly
```

This command removes the ExServer2 server from DAG1. Later, when the server is restored to full health, you can reintroduce it into the DAG.

Handling storage failures

Without a properly functioning storage subsystem, Exchange cannot function. Except in the case of an obvious disk failure, it is easy for administrators to fail to notice that storage is having problems—until users start to complain.

Some automation to identify and take action to address lurking hardware problems would obviously help. Exchange 2010 SP1 began the journey to self-maintenance by incorporating code to handle these conditions more effectively to improve its high-availability capabilities. The first problem the developers attempted to address was storage hangs, which occur when an application such as Exchange issues an I/O request that cannot be satisfied by the I/O subsystem for some reason, usually when a bug is encountered in either the storage software or hardware.

Exchange 2010 SP1 periodically checks for storage hangs that occur in two ways:

- ESE or the Information Store detects a data drive hang. The process attempts to read or write to a data drive and no response is received. When a sufficient period has elapsed (a couple of minutes) to ensure that the drive should have been able to respond and the same problem is observed on multiple data drives, ESE concludes that the server is suffering from a problem from which it is unlikely to recover; to restore service, ESE therefore raises a failure item. In turn, Active Manager notices that a failure item has been raised and initiates recovery through a bug check (a forcible reboot).

- The Replication service (running in a DAG) detects that the operating system is not responding to its requests. This quickly becomes an issue that can compromise high availability, so the solution is blunt and rapid: Exchange forces the server to stop with a bug check and reboot. This step is important because the cluster heartbeat service might still function. Administrator action is usually required to discover why

the operating system has failed and to address any lurking problems that might exist. When the failed server is brought back online, the incremental reseed process restores its database copies to a healthy state.

In both cases, the logic is that it is better to reboot a failing server because this restores service as quickly as possible to clients. If the server is standalone, the hope is that the reboot will bring a fully functional server back online, and the databases will be remounted and become available for connections. If not, the administrator needs to investigate why the issue occurred. Usually, the cause lies in some hardware failure.

Exchange 2013 adds a number of other monitored conditions that have been identified as issues that affect the stability of Mailbox servers. These include:

- **Failure to schedule threads** Exchange is a multithreaded application and depends on its ability to create new threads to perform processing on behalf of mailbox owners. If Exchange is unable to create new threads, it's a sign that the server is experiencing deep problems. Moreover, if the condition lasts for longer than five minutes (302 seconds to be precise), Exchange forces Windows to restart.

- **Excessive I/O latency** Exchange measures the responsiveness of the server to its I/O requests because getting data from the database is fundamental to its operations. Given the speed of current hardware, the usual response is almost instantaneous. Therefore, if the latency is measured in tens of seconds, it's a fair indication that something is seriously wrong. Exchange waits until it observes that 41 seconds has elapsed since a response to an I/O request and then forces a server restart.

- **Excessive growth in Replication service working set** On DAG members that host active and passive databases, MSExchangeRepl.exe is a busy process because it takes care of all the activities necessary to replicate and replay transaction logs. Even though it is a critical process that works with a lot of data, it should not pass a 4 GB working set size. If this happens, Exchange concludes that some problem has caused an excessive use of memory and terminates the Replication service, logging event 4395 (in the crimson channel) to indicate that this has happened. The service should restart after it is terminated. If Exchange cannot terminate the service, it forces a server restart.

Other conditions that are problematic within a DAG are added to the set that Exchange monitors in the cumulative updates released by Microsoft. For example, Exchange 2013 CU1 adds a check to ensure that the Replication service endpoints are responding.

Forcing server restarts seems to be a crude method to fix problems. However, it's simply a reflection of the difficulty of dealing with some of these hard to understand and measure conditions that occur with very complex software. Although you can anticipate that Microsoft will introduce more sophisticated error handling over time, it's still a fact of

computer life that some conditions can only be solved with a reboot, if only to get the server back to a known status. It saves time and restores service more effectively when Exchange takes action automatically without requiring a human administrator first to realize that a problem is happening, think about what to do, and finally conclude that a reboot is necessary to clear the problem.

The fact that the software is capable of handling server outages helps mitigate the effect of needing to reboot a server. If the server is inside a DAG, forcing it to reboot causes Exchange to transfer the active databases from the now-failed server to other copies within the DAG and resume service to users as quickly as possible. The normal server failover mechanism then determines that the databases hosted on the failed server are offline and brings the best database copy to restore service to users.

INSIDE OUT **No Watson here**

You might expect that Exchange generates a Watson dump when it forces a bug check. This doesn't happen for the very good reason that the bug check is forced to bring a server back online and restore service, something that you typically want to happen as quickly as possible. Creating a Watson dump would simply slow down this process and might not provide any useful data, so the Exchange developers chose not to create Watson dumps.

Datacenter Activation Coordination

Some DAGs are deployed across data centers in multiple sites. These environments create a specific problem when catastrophic hardware failures occur that knock out a complete data center. Normally, when multiple servers in a DAG are affected, the DAG remains online if quorum can be maintained or stays offline until the majority of the servers are restored to good health and come online again to establish quorum. At this point, Active Manager attempts to restart DAG operations and activate databases to recover from the failure.

Things are a little more complex when a DAG stretches across multiple data centers, especially if the network connections between the two data centers are lost. As the DAG members gradually come online and the DAG restarts operations, it is possible to provoke a condition known as *split brain syndrome* because the DAG members cannot receive heartbeat signals from the other members.

Consider when a six-member DAG extends across two data centers with the witness server in data center A and an alternate witness server in data center B. Database copies are

distributed across both data centers so that either data center can provide a full service. If a network outage occurs so that data center A cannot be contacted, it is possible to perform a switchover and bring the three surviving members in data center B online by configuring the DAG to use the alternate witness server to achieve quorum. If data center A were now to come online in a manner that still prevented connectivity between the two data centers, its servers could form quorum using the original witness server. Now you have a problem because the DAG is split into two parts, each of which seems functional with active databases.

To prevent this problem, a DAG can be configured to use Datacenter Activation Coordination (DAC) mode so that a majority of DAG members (including the witness server if required) must always be online and communicating with each other before the DAG is deemed operational and thus able to mount databases. DAGs configured to use DAC use the Datacenter Activation Coordination Protocol (DACP) to enforce DAC mode.

When a DAG is configured to use DAC, Active Manager maintains an in-memory flag (called the *DACP bit*) to inform the DAG whether it can mount local databases on the server. When Active Manager starts on a server, the bit is set to zero, meaning that Exchange cannot mount local databases. As it starts, Active Manager communicates with the Active Manager role running on the other members of the DAG to determine the value of their DACP bit. If any server responds with a DACP value of 1, local mounts are possible, and the server where Active Manager is starting changes its DACP bit to 1 and mounts its active copies. When a DAG has only two members, it uses the boot time of the witness server instead of the DACP bit. Active Manager periodically reads and saves this value to be able to use it for arbitration purposes. Because the DACP bit from the witness server is 1, the first server starting in the DAG can mount its databases if the witness server is available. For two-node DAGs, it is therefore critical for the witness server always to be operational before the first DAG member to ensure that the DAG can come online properly.

The default setting for a DAG is that DAC mode is turned off because DAGs support third-party replication technology that might not support DACP. All DAGs that use Exchange-based replication should be configured to use DAC mode by using the Set-DatabaseAvailabilityGroup cmdlet to set the DatacenterActivationMode property.

```
Set-DatabaseAvailabilityGroup -Identity 'DAG1' -DatacenterActivationMode 'DagOnly'
```

Approaching DAG designs

Any technology only moves out of the interesting category to become valuable when it can be applied to solve real-life operational problems in production environments. You've already looked at how you might calculate the raw storage requirement for a deployment. In this section, take a different view and see how to use DAG technology to address the high-availability requirements for a fictional company. The options discussed here are not

exhaustive and are simply intended to get the conversation going. Planning for a production DAG environment needs to be done on the basis of solid information, well-specified requirements, and probably buy-in from different parts of the company. All these points must be covered before you come to discuss the finer points of Exchange.

Like all technologies, the DAG includes various constraints and limitations that system designers have to take into account before they can deploy a DAG. To begin, the following is a list of the most important constraints you have to take into account when you consider the kind of DAG you can create:

- No more than 16 servers can exist in a DAG. The general rule is that a DAG becomes more resilient as the number of servers in the DAG increases. This is because the DAG can host more database copies spread across more servers, the increased number of servers provides additional robustness against losing a server, and the increased number of members in the underlying cluster reduces the likelihood that a quorum failure will occur and render the cluster useless. These are important points that should be taken into account as you consider whether to build a single DAG or split servers across multiple DAGs. No practical limitation governs the number of DAGs that can exist within a single Exchange organization.

- The normal situation is that all the servers are in the same Active Directory site. However, if necessary, a DAG can be stretched across multiple data centers and multiple Active Directory sites as long as the necessary network connectivity (round-trip latency of less than 500 milliseconds) exists between the two data centers. Five hundred milliseconds should be considered the highest acceptable amount of latency. A more capable network that delivers faster throughput and lower latency is preferred, if not always possible.

- Sufficient network capacity must exist to support the log replication with latency of no more than 500 milliseconds.

- The dependency on Windows Failover Clustering means that all servers within the DAG must run the appropriate edition of the same version of the operating system (for example, all servers run the Standard edition of Windows Server 2012). Any hotfixes or other updates that Microsoft releases for failover clusters should be installed. The same software version for Exchange 2013 should be used, including the current cumulative update.

- You can deploy a DAG with the Standard or Enterprise version of Exchange 2013. The Standard edition is limited to five databases, whereas the Enterprise edition allows a single server to support up to a maximum of 100 databases. A server that supports a large number of active databases will need sufficient resources to support the I/O and memory demands required to mount and service its workload. You can use the

Exchange 2013 server role requirements calculator (available for download from *http://gallery.technet.microsoft.com/Exchange-2013-Server-Role-f8a61780*) to validate server configurations against expected workload.

- Two copies of the same database cannot be on the same server.
- Microsoft recommends the maximum size of a database to be 2 TB, assuming that the database is protected by at least two copies. Many system designers are uncomfortable with the idea of running such large databases, even when protected by replication, so 1 TB might be a better design point until the Exchange community gains more operational experience with very large Exchange databases running within a DAG.
- Circular logging is supported for databases that are protected by at least two copies. In this situation, the transaction logs and the Content Indexing (CI) catalog can be placed on the same drive as the database.
- Microsoft performance tests indicate that active users generate approximately 0.2 I/O operations per second. The storage infrastructure must be able to support the I/O demand generated by database operations.
- Any business requirements such as the maximum tolerance for an email outage should be taken into account.

With a full understanding of DAG design principles and limitations in mind, you can explore how to apply the technology to meet common requirements for high availability and disaster recovery that you find in enterprise deployments. Assume that your fictional company wishes to meet these requirements:

1. Host 25,000 mailboxes with 5 GB quota. This is a good figure for the purpose of the design exercise because the needs of most enterprises can be satisfied with 25,000 mailboxes. Larger enterprises can scale up the numbers outlined in this design until they hit the limits of 16 servers in a DAG or 100 active databases mounted on a single server, in which case, they can deploy multiple DAGs to meet their needs. Smaller enterprises can scale down the numbers of servers and databases.
2. Deploy across a primary and a secondary data center to ensure the ability to resist a complete data center outage. If the primary data center is inoperative due to an outage, service must be transferred seamlessly to the secondary data center.
3. Provide resilience against a storage failure on a single server within the primary data center. Provide the same resilience within the secondary data center if service is transferred there.

4. Handle multiple storage failures on different servers in each data center without losing all online copies of a database. VSS backups to disk followed by copies to tape provide archival copies and satisfy audit requirements.

5. Have a maximum of 5,000 mailboxes on a server during normal operations and maintain sufficient capacity for servers to handle the redistributed mailbox load following the failure of a complete server or during times when a server is taken offline for scheduled maintenance.

The first implication that arises from these requirements is that you have to create a design that employs two data centers to ensure continued service following an outage that removes the primary data center from service. To satisfy the requirement, the DAG will be stretched across both data centers within a single Active Directory site, and sufficient servers will be placed in both data centers to handle the complete load that 25,000 mailboxes generate. If multiple Active Directory sites are already in place, the DAG can use them. Selecting one option or the other is simply a matter of personal choice.

You will also create sufficient database copies in both data centers to ensure that you can handle the various storage outage scenarios envisaged in the requirements. The need to accommodate the database copies drives the need for sufficient high-quality bandwidth to connect the two data centers to handle the log replication traffic and information other applications use (client, cluster, Active Directory, monitoring, and so on). The database copies on the servers in the secondary data center will have higher activation preference so that the databases in the primary data center are always used first. Sufficient CAS and hub transport servers will be deployed in both data centers to enable the servers in one data center to handle the complete connectivity and transport load.

The second implication is that you must have at least two database copies in the primary data center. One of these copies is the active database, and the other will provide the onsite resilience against a storage failure on a drive that hosts a database. Two copies are also required in the secondary data center to provide the same resilience if a storage failure affects a database following the transfer of service to the secondary data center. Your design therefore features four copies of every database. Every transaction log generated on the active database has to be replicated and replayed on three other servers, so all the servers will be reasonably busy as they update their database copies.

The third implication is that you require at least five Mailbox servers in both sites to satisfy the requirement to have no more than 5,000 mailboxes on a server. If one server fails or needs to be taken offline for maintenance, the remaining servers must be able to handle the load generated by 6,250 mailboxes (5,000 original plus 1,250 from the failed server). Ten Mailbox servers fit quite comfortably in a single DAG and allow for six additional servers to be added if load increases through additional mailboxes or an increase in mailbox quota.

All servers will run Windows 2012 Standard edition. You could deploy Windows 2008 R2 SP1, but you will not be able to perform in-place upgrades to the operating system, so there is no point in deploying an operating system version that might be outdated soon. All servers will run Exchange 2013 complete with the latest cumulative update and whatever other operating system and security updates are available.

25,000 × 5 GB mailboxes results in roughly 125 TB of mailbox database. Microsoft recommends that a mailbox database can be a maximum of 2 TB if it is protected by multiple copies within a DAG. You could distribute the 125 TB of mailbox quota into fifty 3 TB databases and place five active databases on each server. However, this would mean that each database would hold 1,000 mailboxes. Two issues could result. First, the I/O load generated by such a database under full load is likely to be around 200 I/O operations per second, which is quite a heavy load for a single disk. Second, if a server outage happened, the workload represented by the five databases from the failed server would be distributed across the other four servers in the data center. One of the servers would have to take two additional databases, or the load of 2,000 mailboxes, which exceeds the requirement to restrict the maximum load to 6,250 mailboxes following a server outage.

In any case, a 3 TB database is too large. Today, not many people have much operational experience with running even 2 TB mailbox databases, so perhaps you don't want to use databases that are quite so large. A server running Exchange 2013 Enterprise edition can accommodate many more than five active databases, and a larger number of smaller databases allows more flexibility in distributing I/O load across available disks and transferring load following a server outage. In addition, smaller databases enable you to grow if necessary to accommodate larger mailbox quotas or archive mailboxes without creating the requirement to move mailboxes around. Therefore, set your maximum database size to 1 TB (200 mailboxes) and decide that each server will host 25 active databases. Each server will also host 25 copy databases from other servers. In the event of a server failure, the work for six or seven databases (1,200 to 1,400 mailboxes) will be transferred to each of the four remaining servers. You can constrain each server so that it can mount no more than 32 active databases, a workload of 6,400 mailboxes.

If the primary data center suffers a complete failure, the workload will be transferred to the secondary data center. Exchange 2013 makes this transition easier because reconfiguration of DNS is not required to redirect client traffic. Two database copies are still available in the secondary data center, so a further storage failure can still be tolerated, including a failure that takes out a complete server. If more than one storage failure is experienced, it could remove service to one or more databases, depending on the distribution of the database copies. However, such a series of failures following a complete data center outage is highly unlikely and could only be considered the height of bad luck.

In addition to the 125 TB of space needed for mailbox databases, you should add 40 percent to ensure that no disk is ever completely full and to account for the content index

(CI) catalogs, transaction logs for the databases, and associated headroom for growth and maintenance. The design goal is therefore 175 TB × 4 copies, or 700 TB. Note that this figure accounts for mailboxes that are completely full. It is unlikely that a production environment will operate in a state in which all mailboxes are full, especially with a generous 5 GB quota. An average mailbox might be half full, so you could plan to provide 350 TB of storage initially and then grow over time. Archive mailboxes are a complicating factor because they can be held in the same database as the primary mailbox and could therefore double the storage requirement. Of course, you can split primary and archive mailboxes across different databases, so your design might use dedicated databases for archive mailboxes that are managed in a different manner than the databases that hold primary mailboxes. Each of the mailbox databases is a maximum of 1 TB. Adding 40 percent for the CI catalog and transaction logs gives 1.4 TB, so a single database and its associated files will fit comfortably on a single 2 TB disk and still allow space for growth. However, this is a simplistic calculation; the storage has to be able to satisfy the I/O demand as well. Microsoft performance guidelines use 0.2 I/O operations per second per mailbox. This figure depends highly on user work habits and is influenced by the number of items and folders in user mailboxes. To be safe, you can use a figure of 0.25 I/O operations per second, so a mailbox database with 200 mailboxes generates a demand of 50 I/O operations per second at peak load, assuming that all the users are concurrently active.

The calculations do not take into account archive mailboxes or features such as an extended deleted items retention period or in-place holds. The exact storage design that is implemented will have to consider these factors. On a pragmatic level, the storage design will be highly influenced by the existing storage architecture, current available storage products, and other requirements such as the backup regimen. For example, an organization that uses storage area network (SAN) storage for Exchange 2007 might want to extend its use for Exchange 2013 and will therefore make the necessary adjustments to accommodate the storage and I/O requirements of DAG member servers plus the demands of other applications that are run on the other servers in each data center. However, an organization might want to take advantage of the lower I/O demand of Exchange 2013 to move to a lower-cost just a bunch of disks (JBOD)–based storage design, especially if it has an eye on expanding mailbox quota from 5 GB to a higher value in the future.

The storage calculators and other sizing tools available from hardware vendors and Microsoft provide invaluable assistance to system designers because they calculate the foundation for a storage design that has to deliver a certain amount of raw storage and I/O capabilities. In any but the simplest deployment, you shouldn't expect these calculators to provide the most effective and cost-efficient storage design. Instead, treat the output from the sizing tools as a good starting point and then give the data to a storage expert to have her fine-tune the design to meet your exact requirements. One idea is to provide the output from the sizing tools to storage vendors and invite them to state how their products will deliver the necessary performance, capacity, security, and resilience. Apart from the

potential of getting some free advice from vendor storage specialists in the responses you receive, you might benefit from competitively priced storage packages specially designed for Exchange that might be an off-the-shelf solution for many designs. In addition, the vendors can brief you on new storage and other technology with capabilities that might not be captured in the current generation of sizing tools.

How Exchange 2013 changes the resilience equation

Designs for an Exchange 2010 DAG have to take into account the characteristics of the application and its clients. The same is true for Exchange 2013 DAGs. The question therefore is, "What's changed, and how does this influence DAG design principles?"

The most fundamental change is the use of HTTP-based connections across the vast majority of client protocols. Leaving IMAP4 and POP3 to one side, Outlook, Microsoft ActiveSync, Outlook Web App, and Exchange Web Services are all HTTP-based in an Exchange 2013 environment, as are the EAC and remote Windows PowerShell protocols used for administration. This change removes much of the complexity for site switchovers seen in Exchange 2013. Whereas Exchange 2010 requires DNS reconfiguration to redirect clients to a different site, Exchange 2013 merely requires multiple IP addresses to be configured in DNS for the namespace that clients use to access mailboxes. For example, if a DAG stretches across data centers in London and Dublin, each of which has one or more virtual IPs (VIPs) serviced by a CAS array, it is sufficient to publish the IP addresses for the VIPs for the FQDN of a common namespace, say mail.contoso.com. If a client cannot access the first IP address, it will automatically use the second (and third, fourth, and so on if other addresses are available). Navigating through a list of IP addresses to find a connection to a site is a technique inherent to all modern HTTP clients, so there should be no issue using it with Exchange 2013 clients.

A failure of the VIP or the underlying load balancer or CAS servers in a data center is no longer the problem that it was in Exchange 2010. Instead of worrying about DNS reconfiguration and TTL timeouts for clients, you rely on the HTTP stack running on clients to detect the failure automatically and then to move on to the next address for the FQDN, a transition that usually takes about 20 seconds. So, if the load balancer in London fails, clients attempt to connect to the VIP, fail, and move on to the IP address for the VIP in Dublin. The load balancer in Dublin has to be able to handle the load of the clients switched from London. Assuming that this is the case, the connections will be transferred to CAS servers in Dublin and proxied back to the Mailbox servers running in London. Outlook Web App users might have to refresh their browsers, but Outlook users configured in cached Exchange mode should be unaware of the transition. Defining multiple IP addresses for a FQDN also helps when a component such as a load balancer needs to be taken offline temporarily to apply an update or even needs to be rebooted to clear a transient problem. Depending on the period for which the component might be offline, you can either depend on client failover to the other site or remove the affected VIP from DNS to prevent clients from

attempting to use it. When everything is fixed, you can then add the VIP back into DNS, and clients will begin to connect and use it again.

Of course, you assume that network connectivity exists between Dublin and London to allow connections to be proxied to the Mailbox servers. Defining multiple IP addresses for a FQDN does nothing to provide additional resilience for databases, which continue to require database copies mounted on multiple servers spread across multiple data centers to provide maximum protection against failure. Similar to Exchange 2010, the trick is to make sure that sufficient database copies exist to handle different failure conditions, such as the loss of a complete data center, and to make sure that the Mailbox servers have sufficient capacity to activate more databases than the norm. Given the heavier memory requirements to cache database contents on Exchange 2013 servers, you might need to revise server configurations to ensure that the Mailbox servers can activate a heavier database load following a complete data center outage.

With the proper hardware in place, the usual approach to creating a DAG that can survive a complete data center failure is to:

- Divide the Mailbox servers across the two data centers.
- Spread database copies across the two data centers so that copies for all the databases in each data center exist in the other data center.
- Ensure that the witness server is always available so that quorum can be maintained.

For instance, you could place the witness server in a third data center, which has independent connectivity so that it cannot be affected by a failure of either of the two data centers where the Mailbox servers are located. As long as all the Mailbox servers in one data center are online and can communicate with and lock the file share resource on the witness server, quorum can be maintained, and the surviving DAG members remain online.

Like any other guidelines given about DAG design, these are principles to take into account in your design rather than hard and fast directives. The fact that Exchange 2013 DAG design principles differ from Exchange 2010 because of an architectural change in client connectivity points to the wisdom of not assuming that previous DAG designs will work as well in Exchange 2013. You are forced to move mailboxes to form new DAGs as part of the transition to Exchange 2013. Before you move anything, make sure that the new DAGs are based on Exchange 2013 principles rather than on outdated knowledge. Take the time to read up on new experience with operational DAGs, the problems people have encountered, and the current state of best practice and perhaps listen to some of the sessions from TechNet and other conferences that are available online before you settle on a final DAG design.

Building a DAG with Exchange Standard edition

Some ask whether it is a good idea to build a DAG using the Standard edition of Exchange. Although this is obviously possible because neither Exchange 2010 nor Exchange 2013 requires the Enterprise edition to form a DAG, I'm not sure that this is a good route to take.

All companies need to keep a close eye on IT expenditures and take all reasonable steps to restrict cost wherever possible. Thus, a natural tendency exists to investigate lower-cost options. The Standard edition of Exchange is less expensive than the Enterprise edition, so it makes sense to consider its use wherever the Enterprise edition is not required. For example, using the Enterprise edition on dedicated CAS servers wastes money because the Enterprise edition provides absolutely no advantage over its lower-priced counterpart when it's not necessary to support mailbox databases. The same is true for Exchange 2010 edge servers.

A DAG can accommodate servers running either the Standard or Enterprise edition or a mixture of both. However, to ensure consistency across the DAG, best practice is to run the same server (operating system and application) software on all DAG members unless you have a good reason to vary it.

A DAG built on the Standard edition of Exchange 2013 can support five mounted databases on each server (mailbox and older-style public folder databases count against the limit), whereas using the Enterprise edition allows each server to mount up to 100 databases. Assuming that the DAG design provides sufficient servers to deliver adequate data redundancy by allowing for three copies of each database, it's therefore reasonable to argue that you can avoid the higher software license fees required for Exchange 2010 Enterprise edition by using the Standard edition.

A small DAG (say, three Mailbox servers) can certainly deliver good redundancy at an excellent price point, especially if you deploy JBOD-style storage. But you deploy a DAG to deliver highly available mailboxes, pure and simple. If you accept that this statement is true, then it follows that you should support the software with a high-quality environment. In other words, you pay attention to points such as intelligent storage controllers, disks with an extended mean time between failure (MTBF) capability, and proven server hardware. In short, you invest in the total solution to reduce the potential of hardware failure.

To take this approach further, shouldn't you then opt to deploy the Enterprise edition of Exchange to be able to support more than five mounted databases on a server? The rule of thumb is that the more flexibility a DAG possesses, the more resilient it will be to the failure of any single component (software, disk, controller, or server). That's why, in general, it is preferable to operate a large DAG (up to the 16 members that Exchange 2010 or Exchange 2013 supports) rather than several small DAGs because the large DAG can tolerate more failures than a smaller DAG can. For example, if a small, three-server DAG experiences the

failure of two of its servers, it immediately loses cluster quorum and the ability to maintain more than one copy of any database. Although you can use the dynamic quorum feature to force a single member server to remain online and so make some or all the databases available, this requires administrator intervention. A larger, six-member DAG can tolerate the loss of three member servers and continue to operate without a problem with all its databases still having at least two available copies.

In addition, having the ability to mount more databases on a server means that the databases can be smaller. Many good things flow from not having to cope with very large databases. Transitions following failures and administrator-initiated switchovers are faster, as are reseeding operations (if the need occurs). Backups also take less time, and if a database experiences a catastrophic failure in a way that prevents it from being recovered from a copy, a restore from disk is quicker, too. DAG expansion is also easier because you don't have to worry about hitting the five-database limit. All in all, a number of operational issues are made easier when more databases can be mounted. Apart from the higher license costs, I'm not sure I can think of any downside in saying that the Enterprise edition of Exchange is usually the best choice to provide the underpinning of a DAG.

This isn't to say that there aren't some interesting edge cases to be explored. For example, you might want to have a single server in a DAG that runs the Standard edition of Exchange but is expressly intended to host only passive database copies, usually copies of databases holding the mailboxes of very important people, often executives or others who are deemed critical to the company. In such a scenario, other copies of these databases exist within the DAG and can be activated if required, the copies on the standard server are assigned higher activation preferences, and the entire Mailbox server could have its activation policy set to Blocked with the Set-MailboxServer cmdlet so that Active Manager won't even consider attempting to bring the copies online. For instance:

```
Set-MailboxServer –Identity ExServer1 –DatabaseCopyAutoActivationPolicy Blocked
```

In essence, the idea is that the server will provide an extra copy of up to five databases containing critical mailboxes that could be brought online through administrator intervention if something really horrible happened to all the other database copies. It's possible to argue that some value exists in the idea, but the circumstances in which value can be gained are unlikely. As such, it is still better to run the Enterprise edition on all servers so that they can be fully functional within the DAG.

A possibly more interesting scenario is when you want to have a server that only hosts archive mailboxes. Exchange allows primary mailboxes to be separated from their associated archives. In most instances, administrators accept the default and keep primary and archive mailboxes in the same database. There is value here because it's a simple setup, and simplicity is always better. However, even if only on the "don't put all your eggs in the same basket" principle, value can also be gained by separating archive mailboxes from their

primary mailboxes. Some system designers contemplate creating an archive server running Exchange Standard edition when the five available databases store only archive mailboxes. The idea here is that the server hardware can be purpose-designed to accommodate the lower resource demands of archives while taking advantage of the lower cost of Exchange Standard edition. In addition, some see an advantage in clearly separating archive data from live data. Once again, the notion is interesting, but overall, it is still better to have all DAG members run the same hardware and software so that the load can be distributed equally across all servers.

Every company is different, and the arguments advanced here will be more or less attractive, depending on the circumstances in which a company finds itself. It's an interesting debate to have, perhaps after discussing whether Mailbox servers should be virtual or physical.

Server maintenance

It's a fact of administrator life that software updates will become available on a regular basis for Exchange, Windows, and other products. Although you don't have to apply the regular updates and service packs that Microsoft releases for Exchange, it's a good idea to do so to ensure that you run the latest software that contains all the fixes to known problems, including security fixes for vulnerabilities that are discovered and fixed on an ongoing basis.

The Exchange code base is massive, and it's inevitable that some bugs will be encountered, so it makes sense to keep up to date with software fixes as Microsoft releases them. This is not a call to install updates immediately as they become available. You should take the time to test any new software before installing it in a production environment. Software testing is a well-developed art these days, and Microsoft and other vendors put enormous effort into testing updates against many configurations, including regression testing to ensure that a fix doesn't reveal another problem, but it is possible that something is unique in your deployment that could cause a problem for a cumulative upgrade or service pack.

All the usual caveats about software upgrades are valid for DAG servers. Make sure that you've previously run the procedure on a test server that accurately represents the production environment, that you have all the software that might be required, that you've allowed enough time to perform the work, and that you have scheduled it to be done at a time when any outage will not inconvenience users severely. The presence of Managed Availability makes DAG management slightly more complex when the time comes to upgrade a member server. After testing that an update is suitable for deployment, you'll want to install it on your production servers, including those that are DAG members.

As you'll recall from the "Managed Availability and high availability" section earlier in this chapter, the new Managed Availability system is designed to monitor the health of Exchange components running on servers. If Managed Availability detects that a problem

exists, it takes action in an attempt to fix the problem right up to when it might bugcheck a server to force a server failover. Of course, this action is in line with the age-old administrator adage that a reboot can't hurt when a server is unwell. Bug checks are the last resort of Managed Availability and happen when other actions such as restarting an application pool or service doesn't fix a problem.

When software is being upgraded on a server, it's likely that the server will appear unhealthy in the eyes of Managed Availability. For instance, an update might require a service to be stopped while software modules are replaced on disk. During this time, the Managed Availability probes will return failures. Even though the human administrator expects the failures, Managed Availability does not know that a software update is in progress and so might respond to the data returned by its probes by concluding that action is necessary to restore the server to good health. The last thing you want is for Managed Availability to interfere with the ability of the software update to proceed. A forced bug check in the middle of an upgrade is not a good thing, so some method is required to inform Managed Availability that it shouldn't worry about a particular server for a period. This is done by placing a server in maintenance mode before running any procedure that might cause Managed Availability to worry. Maintenance mode is intended to signal that a server is currently available or unavailable to service clients, with the transition from one mode to the other being performed by running the Set-ServerComponentState command.

Follow this procedure to place a mailbox or multirole server in maintenance mode. This procedure replaces the scripts supplied to assist with DAG maintenance in Exchange 2010 (StartDagServerMaintenance.ps1 and StopDagServerMaintenance.ps1). These scripts work, but they don't tell Managed Availability that a server is going into maintenance mode, so they should not be used with Exchange 2013.

The following commands operate against a server named ExServer1. You can replace the server name with $env:ComputerName to force EMS to use the name of the server to which you are connected. Although convenient, this is not a safe practice because the nature of remote Windows PowerShell connections means that you can never be certain to which server you are connected. It is therefore better practice to be specific and state the name of the server you want to manage.

1. Put the transport system into a state in which messages that are currently queued will be processed, but new messages will not be accepted. You can monitor the transport queues with the Get-Queue command to make sure that outgoing messages are dispatched and the message count for the queues goes down to zero. You can ignore the Poison or Shadow queues because these are internal queues. You just want to make sure that any outbound message to a real person has been dispatched before you put the server into maintenance mode. Depending on the volume of messages the server handles, it might take a couple of minutes before you see the queues drain fully.

```
Set-ServerComponentState –Identity ExServer1 –Component HubTransport
–State Draining –Requester Maintenance
Get-Queue –Server ExServer1 | Format-Table Identity, MessageCount
```

2. If the server is a DAG member node, suspend it from the cluster, move its active databases to other nodes in the cluster, block activation, and then check that all the database copies are Inactive but Healthy. Replication still continues at this point because although Active Manager cannot activate any database copies on the server, it is still a functioning member of the DAG.

```
Suspend-ClusterNode –Name ExServer1
Set-MailboxServer –Identity ExServer1 –DatabaseCopyActivationDisabledAndMoveNow
$True
Set-MailboxServer –Identity ExServer1 –DatabaseCopyAutoActivationPolicy Blocked
Get-MailboxServer –Identity ExServer1 | Format-Table DatabaseCopy* -AutoSize
Get-MailboxServer –Identity ExServer1 | Get-MailboxDatabaseCopyStatus
```

3. Redirect messages to another mailbox or multirole server. Any messages on active queues will be redirected to the target server, and the server against which you run the command will stop accepting new messages.

```
Redirect-Message -Server ExServer1 -Target ExServer2.contoso.com
```

4. Set the server to be marked as inactive due to maintenance.

```
Set-ServerComponentState –Identity ExServer1 –Component ServerWideOffline
–State InActive –Requester Maintenance
```

5. Check that the server is now in the appropriate state for maintenance and that all the mailbox databases have been activated on other DAG members. You should see Inactive reported as the state for each component in the list, and the Get-ClusterNode command should report the status as Paused.

```
Get-ServerComponentState –Identity ExServer1 | Format-Table Component,
State –AutoSize
Get-ClusterNode –Name ExServer1 | Format-List
```

6. Exchange 2013 CAS servers are essentially stateless, so all you need to do to prepare these servers for maintenance is to tell Managed Availability that the server should be regarded as offline.

```
Set-ServerComponentState –Identity ExCASServer1 –Component ServerWideOffline
–State InActive –Requester Maintenance
```

7. Issuing the commands to put a server into maintenance mode might not have immediate effect because Managed Availability has to acknowledge the command and then adjust its probe and measurement activity. It is best to wait 5 to 10 minutes after running the commands to put the server into maintenance mode before proceeding with whatever software updates are required. When these are complete and you have verified that they completed successfully, you can restore the servers to

normal operation. For CAS servers, this just requires you to reverse the switch and set the server back to an active state:

```
Set-ServerComponentState –Identity ExCASServer1 –Component ServerWideOffline
–State Active –Requester Maintenance
```

After the command completes, the server should begin to process incoming client connections. The process is a little more complicated for Mailbox servers, but again, it's a matter of reversing the process used to put the server into maintenance mode.

8. Make the server active. Note that the requester specified here is Maintenance, the same requester used when the server was placed in an inactive state. It is important to match the requesters; otherwise, the server will remain in maintenance mode.

```
Set-ServerComponentState –Identity ExServer1 –Component ServerWideOffline
–State Active –Requester Maintenance
```

9. Resume the server as a cluster node.

```
Resume-ClusterNode –Name ExServer1
```

10. Remove the restriction on database activation for the server.

```
Set-MailboxServer –Identity ExServer1 –DatabaseCopyAutoActivationPolicy
Unrestricted
Set-MailboxServer –Identity ExServer1 –DatabaseCopyActivationDisabledAndMoveNow
$False
```

11. Allow transport to begin accepting messages for processing.

```
Set-ServerComponentState –Identity ExServer1 –Component HubTransport
–State Active –Requester Maintenance
```

These steps make the server fully operational again. However, the databases that were previously active on a DAG member are not automatically switched back to the now-upgraded server. You have to select and activate databases individually, perhaps as part of the preparation to apply a software update to the next DAG member.

Stressed servers

Test servers tend to have fewer capable hardware and performance characteristics than production machines. It's easy to stress these servers in a way you cannot stress production-quality systems. For this reason, it's sometimes necessary to inform Managed Availability that a server is about to be stressed so that it doesn't respond to the symptoms produced by the workload and attempt to rectify matters by restarting services or even restarting the server. This command informs Managed Availability that it should not take any actions if it detects a problem. In effect, you are flying under manual control:

```
Set-ServerComponentState -Identity ExServer1 -Component RecoveryActionsEnabled
-State Inactive -Requester Sidelined
```

To revert to automatic operation, run the command again and set the RecoveryActionsEnabled component to Active.

On to protecting data

You've read a lot about how to store information in mailbox databases. Now it's time to introduce some data. People will create some as they use their mailboxes, but much of the data that arrives in Exchange 2013 will come from Exchange 2010 and Exchange 2007 servers. Your next step is therefore to explore how to move mailboxes and the wonders of the Mailbox Replication service.

CHAPTER 10

Moving mailboxes

The Mailbox Replication service 567
Moving mailboxes 568
The Migration service 572
Mailbox Replication service processing 575
Creating and managing migration batches with EAC . . 581
Using EMS with migration batches 592
Reporting mailbox moves 595
Managing individual mailbox moves 605
Migration endpoints 611
Enabling MRSProxy 612
Planning mailbox moves 614
Ensuring high availability 619
MRS configuration file 621
Mailbox import and export 623
Alerts 638
Time to comply 640

Mailboxes do not typically stay in one place for their entire lifetime. Indeed, as we have learned from the last few releases of Exchange Server, moving mailboxes from servers running old software to those running the new software is the only way to transfer information. Mailbox moves happen for lots of other reasons, too, including to populate new databases or servers or to balance workload across available disks or computers. Other operations can affect mailbox contents—importing information from PSTs to take advantage of larger mailbox quotas and make items discoverable is an obvious example. However, you might also have to export data to PSTs, like when providing information retrieved in response to a legal discovery request. The Mailbox Replication service (MRS) made its debut in Exchange 2010 and has taken on increased responsibility for different mailbox operations ever since, now including the movement of modern public folders. Stripped back to its basics, the Exchange 2013 version of MRS retains many of the characteristics that exist in Exchange 2010. A new factor in mailbox moves appears in the form of the Migration service. There's lots to talk about where moving mailbox data is concerned, so let's start at the beginning.

The Mailbox Replication service

The Exchange Mailbox Replication service (MRS) is the service that is responsible for processing move, import, export, and restore requests. In Exchange 2013, the MRS runs on every Mailbox server rather than on the Client Access Server (CAS), as is the case in Exchange 2010. The name of the MRS process is MSExchangeMailboxReplication.exe, sometimes confused with the database replication used with Database Availability Groups (DAGs; MSExchangeReplication.exe), but they are separate and very different processes.

Along with the relocation of MRS to the Mailbox server, the other important points of difference between Exchange 2013 and Exchange 2010 are that:

- MRS now only scans for move requests registered in the system mailbox in the active database copies on the same Mailbox server. In Exchange 2010, MRS scans for move requests in every mounted database within its Active Directory site. This change is made to keep processing as local as possible because Exchange 2013 originally did not require a CAS server to be in every Active Directory site that hosts Mailbox servers. The requirement to have a CAS server in an Active Directory site was reintroduced after Exchange 2013 was first shipped.
- MRS communicates with the Information Store running on the local server using XSO, an internal Exchange application programming interface (API). XSO reads and writes information from mailboxes as MRS moves, imports, exports, or restores them. One benefit gained by using XSO is that data is validated before it is provided to MRS.
- MRS no longer uses MAPI.Net to make connections to mailbox databases mounted on other servers. When a connection is needed to a remote mailbox on a remote server, MRS connects to the MRS running on that server through the MRS Proxy, a Windows Communications Foundation (WCF) interface.
- MRS now handles the processing of public folder content moves for the migration of content from older-style public folders to new public folder mailboxes and whenever the need exists to move content between public folder mailboxes. Only a single public folder move request can be processed at one time. See Chapter 12, "Public folders and site mailboxes," for more information about modern public folders.

The Exchange 2013 CAS still supports an MRS proxy. However, it exists purely to route calls to the appropriate MRS running on a Mailbox server to access the Store. In Exchange 2010, the MRS Proxy running on the CAS could access the Store directly.

These changes reflect the evolution of the Mailbox server role and the way the CAS has evolved to become more stateless than in Exchange 2010.

Moving mailboxes

The ability to move mailboxes between databases is a fundamental operational procedure for Exchange. Because you have to install new servers to deploy Exchange 2013, moving mailboxes is the basic method to migrate user data to Exchange 2013 from earlier servers. Migration also includes hybrid environments, where MRS is capable of moving mailboxes to and from Office 365.

Apart from migration, the other reasons mailboxes are moved include the following:

- **Balancing user load across available mailboxes** As users leave and join the company, the number of mailboxes assigned to databases might vary. Some mailboxes might be larger than others. Administrators often seek to rebalance workload across databases by transferring mailboxes so that each database hosts approximately the same number of mailboxes of roughly the same size. With the introduction of personal archives, administrators now have to take into account the load and storage requirements generated by these mailboxes when they rebalance user load. A variation on the theme occurs when administrators decide that a database is too large for operational reasons. (Perhaps the time required to make a backup exceeds the desired period.) In this case, the administrators can move mailboxes to other databases. Moving the mailboxes will not reduce the database's size immediately because the moved mailboxes are retained until the deleted mailbox retention period expires, but it will stop the database from swelling even larger.

- **Outsourcing to hosted environments** An Exchange organization can span a mixture of on-premises and hosted servers. Mailbox moves are used to transfer mailboxes from on-premises servers to hosted servers and vice versa.

- **Organizational change** Users might be transferred around the company and change their physical location in a way that makes their current mailbox database inappropriate (for instance, the database is on a server in a remote data center). In this scenario, it's common for an administrator to move the user's mailbox to a server that is closer to the user.

- **Organizational hierarchy** Some companies like to place all the mailboxes belonging to selected user groups in specific databases so that special policies are applied to these mailboxes (high availability, storage quotas, deleted item retention periods, and so on). For example, the company executives might all be placed on one database that has four copies to ensure the highest availability, whereas the mailboxes belonging to transient or part-time employees might be placed on another database that has only two copies.

- **Server transition** Before a Mailbox server can be decommissioned from the Exchange organization, all the databases or database copies it hosts must first be removed. Active database copies can be transitioned to other DAG members, and then the database copy can be removed from a server, but before the last copy of a database is deleted, all its mailboxes must be moved to other databases.

- **Mailbox or database corruption** Corrupt items can occur within a mailbox that causes problems for the database. (See the discussion about mailbox quarantine in Chapter 8, "The Exchange 2013 Store.") Moving a mailbox can remove corruption if

you set the bad-item limit to allow Exchange to discard any suspect items it encounters as it moves mailbox content.

- **Mailbox grouping** Some administrators like to collect all the mailboxes of a particular type in a certain database. For example, collect all room mailboxes in one database, place linked mailboxes in another, and so on. The same approach is often taken to group mailboxes on a department basis so that all the mailboxes from the Sales Department are in the Sales database.

INSIDE OUT **Introducing the Migration service**

Microsoft made a major change in the way mailboxes are moved in Exchange 2010. In previous versions, mailboxes are moved synchronously. In other words, you would start a mailbox move and then wait until it finishes before the user could access the mailbox again. Exchange 2010 was the first version to support asynchronous moves that proceed behind the scenes to copy mailboxes in a way that allows the user to continue working until his newly moved mailbox is ready to be accessed. Exchange 2010 uses a set of move-request cmdlets to manage the process of moving mailbox content from one database to another. Microsoft refers to these operations as online mailbox moves. Exchange 2013 improves matters further by introducing the Migration service to provide administrators with a more automated framework to manage mailbox moves through batched migrations, a technique that works well to move mailboxes between databases within the same organization, between different organizations in different forests, and between on-premises and cloud servers. Move requests haven't gone away. They are simply cloaked in migration batches.

Microsoft doesn't change the way Exchange works without good reason. Mailboxes have been swelling in size over the past few years. Whereas a 1 GB mailbox was once uncommon, 10 GB mailboxes have become the norm, and 25 GB mailboxes are offered as a major feature of all modern email systems. Synchronous moves have many problems when dealing with large mailboxes; they take too long, the user is blocked from working with her mailbox (sometimes for many hours), and they are prone to failure—sometimes in the last stage of the move if a bad item is encountered. These limitations are acceptable when mailboxes are small but quickly become a big operational problem as mailboxes grow.

The influence of cloud systems is also apparent. Moving large amounts of mailbox data between servers by using an internal network demands a certain number of resources; moving the same data across the Internet to servers operating in a cloud platform creates quite a different challenge. Most companies have far better internal network bandwidth

and capabilities than they do to the Internet, where excessive latency and packet loss can make moving large mailboxes a challenge. Transferring a 10 GB mailbox to an internal server might take a few minutes, compared to a few hours when a cloud server is involved. Synchronous moves simply cannot cope with these issues.

Asynchronous moving

Exchange mailbox moves happen asynchronously, so you can start many move requests and then do something else. The basic flow of processing is as follows.

A move request is created in the system mailbox of the active database copy that identifies the source mailbox and the target database. Move requests are manipulated with the *-MoveRequest set of cmdlets. To automate the process further, Exchange 2013 creates migration batches, each of which is represented by a set of move requests that can be managed as a single unit. However, the underlying unit for mailbox moves remains a move request.

- MRS copies the contents of the selected mailbox to the destination database.
- MRS performs a check to determine whether any items have been updated or created in the source mailbox since the move started; if any items are discovered, they are copied across, using an incremental synchronization process.
- If a move is interrupted by some error such as a bad item threshold being exceeded, MRS can restart the move from the point at which it halted.
- After all content has been moved to the new mailbox, MRS completes the move by updating the user's Active Directory account to point to the new mailbox location.

The following are the advantages of this approach:

- Users can continue to work as MRS moves their mailboxes to the target server, a factor that becomes more important as mailbox sizes grow.
- Exchange 2013 throttles the workload generated by mailbox moves according to the health of the servers (as measured by Exchange Workload Management), using the limits set in the MRS configuration file (described in the "MRS configuration file" section later in this chapter) as a backstop.
- Archive mailboxes can be moved along with their primary mailboxes or left in the source database. You can also elect to move just an archive to a different database, which is a useful option when you want to relocate archive mailboxes to a dedicated database.

- Deleted items stored in the Recoverable Items folder structure that still have not exceeded their retention period and any items that are subject to an in-place hold are moved along with mailboxes. This statement is true for moves from Exchange 2010 and Exchange 2013 servers, but due to the change in the Recoverable Items structure implemented in Exchange 2010, these items are not preserved for moves from Exchange 2007 servers.

- Items are incorporated into the content index of the target database as they are moved to ensure that users retain fast search capability. This factor isn't as important if you use Microsoft Outlook configured in cached Exchange mode because searches are then performed on the client, using Windows Desktop Search. However, it does mean that eDiscovery searches performed by administrators will continue to locate items after the mailbox is switched from source to target database.

- Online moves executed on Exchange 2007 don't generate much information to help an administrator understand what has occurred during the move. To enable troubleshooting and debugging, MRS captures comprehensive details about every mailbox move and stores these reports in the mailbox. By default, Exchange 2013 retains details of the last five moves for a mailbox. This number can be increased if required.

The Migration service

MRS proved to be one of the most effective and useful new features introduced in Exchange 2010. However, on-premises administrators found that the creation and management of mailbox move requests required a lot of hands-on intervention. For example, if a move failed, an administrator would have to determine what the problem was, fix any issue that was found, and then re-queue the move request.

Microsoft encountered many of the same difficulties as it began moving large numbers of customer mailboxes to Office 365. The solution was to build a framework to allow customers to set up batches of mailboxes for migration from on-premises (IMAP or Exchange) servers to the Exchange Online cloud service. Moves occurred in the background, and administrators had the opportunity to review each move before allowing the mailbox to be switched over. All of this was accomplished by providing a wrapper around MRS that disguised some of the complexities of creating and managing mailbox move requests. The technique proved to be very successful and has enabled Microsoft to move millions of mailboxes to Office 365 over the past few years, proving once again that automation and simplification are extremely useful attributes for IT processes to have.

Exchange 2013 incorporates a similar approach to mailbox moves. The new Migration service is separate but intertwined with MRS and is based on the migration utility for Exchange Online. Its purpose is to manage the processing of migration batches, sets of mailboxes that have to be moved to a destination. The Migration service runs within the

Microsoft Exchange Service Host process (*Microsoft.Exchange.ServiceHost.exe*) on Exchange 2013 Mailbox servers and communicates with MRS by using the same set of cmdlets you can use to create and manipulate move requests, the idea being that it is more efficient to create batches of mailbox moves that the Migration service processes as a unit. The Migration service uses the cmdlets to create (or inject) the new mailbox moves that are part of a batch, manage the moves during processing, inspect their status, and take actions such as suspending and resuming moves.

Whereas MRS stores information about individual move requests in the system mailbox of a mailbox's source database, the Migration service uses the migration arbitration mailbox (named Migration.8f3e7716-2011-43e4-96b1-aba62d229136). This mailbox stores the information the Migration service uses to track the processing of migration batches, including an entry for every mailbox to be migrated, referred to as a migration user.

Because only a single migration arbitration mailbox is present in an organization, it is important to position this mailbox in an appropriate database. If the database is not mounted or is otherwise unavailable, the Migration service cannot run, and you'll see errors when you attempt to work with migration batches through Exchange Administration Center (EAC) or Exchange Management Shell (EMS). The migration arbitration mailbox is created when the first Exchange 2013 Mailbox server joins an organization, so unless it is moved, the mailbox remains in the default database created on that server. Often, that database is not the best choice to be the fulcrum for migration batch processing, and you will end up moving the mailbox to a more appropriate (and better protected) database. For example:

```
New-MoveRequest -Identity 'Migration.8f3e7716-2011-43e4-96b1-aba62d229136'
-TargetDatabase DB1
```


INSIDE OUT The traffic through the migration arbitration mailbox

The Migration service uses the migration arbitration mailbox to store information it uses to process migration batches, including the batches, migration users (objects representing the mailboxes to be moved), and endpoints. Each of these objects is represented by an item in the mailbox, and many updates can occur for each object as a batch is processed. Consequently, during large migration projects in which a lot of mailboxes are being moved to Exchange 2013, the database that hosts the migration arbitration mailbox is likely to generate a lot of transaction logs due to the updates to items in the migration arbitration mailbox. For this reason, it is best to position the migration arbitration mailbox in a database where circular logging is enabled.

EAC uses migration batches as its exclusive method to move mailboxes—moving an individual mailbox is dealt with as a one-mailbox batch. All the necessary user interface is in place to create and manage migration batches while the Migration service provides the necessary automation behind the scenes. You can still create individual mailbox moves or emulate the same kind of processing with EMS if you want to integrate mailbox moves into the process your company uses to deal with user accounts. Although it might seem very different to think of mailbox moves as batches rather than individual move requests, the fundamentals of the operation remain the same. After the Migration service sets up everything in terms of the mailbox moves, MRS performs the processing and provides the link between the servers involved in transferring data. Content comes from a source mailbox and is copied to a new target mailbox. You can pause moves before mailboxes are finally switched over, and a mailbox move is finalized only when its Active Directory pointers are updated.

INSIDE OUT **Look but don't touch move requests the Migration service manages**

The Migration service creates mailbox move requests to prompt MRS to move the mailboxes. These move requests are registered with the Migration service, which takes the responsibility for their end-to-end processing during a life cycle, covering batch creation to completion. Because the Migration service manages the move requests in its batches, it's important for you not to interfere with these move requests by running the *-MoveRequest commands because you run the risk of doing something that will cause a migration batch to fail. For example, don't suspend a move request that is part of a migration batch by running Suspend-MoveRequest; use the Stop-MigrationBatch command instead to allow the Migration service to track what's happening with the move requests under its control.

Another issue created because only a single migration arbitration mailbox exists in an organization is that the traffic created when the Migration service updates the user entries within the migration batch goes to the server that currently hosts the active copy of the database holding the mailbox from servers throughout the organization. This can generate unwanted network overhead and cause problems if that database becomes unavailable while the Migration service is processing a batch. The underlying mailbox moves MRS performs will probably complete, but the Migration service won't be able to update the status for individual migration users or its batches. For this reason, the migration arbitration mailbox should be moved to a database that is protected with copies and is mounted on a server that can process the inbound processing load generated by migration batches.

The new approach delivers the following advantages:

- A far more automated framework for mailboxes to be moved in large batches. The same degree of automation applies for small batches, but the advantage is obviously greater when you have to process more than a few mailboxes.
- Email notification for administrators when processing is complete, including when mailboxes have been autosuspended when they are ready for the final processing step to complete moves. At this point, administrators can review mailbox moves before allowing them to go through to completion—or cancel the entire process.
- Automatic retry and prioritization of moves.
- Periodic incremental synchronizations to keep moved but not finalized mailboxes up to date. An incremental synchronization is performed every 24 hours for each mailbox to avoid the problem that can occur with Exchange 2010 when a mailbox is left in an AutoSuspended state pending completion for several days, resulting in a very large incremental set of updates that have to be made before the mailbox can be finally moved.
- Ability for primary and archive mailboxes to be moved together or separately.

Batching mailbox moves is not new. Indeed, Exchange 2010 supports the BatchName parameter for the New-MoveRequest cmdlet to identify an individual mailbox move as belonging to a particular set. For example, you might identify the moves for all the mailboxes in a particular department by identifying them as a batch and then use the name assigned to the batch to monitor processing of the moves by reference to that name. Although they don't use the Migration service, third-party migration products such as E2EComplete from Binary Tree take much the same batch-driven approach when they move mailboxes to and from Exchange.

Mailbox Replication service processing

Before you return to the topic of how to create new move requests through EAC and EMS, you need to understand the basics of how MRS moves mailboxes after requests are created individually or as part of a migration batch.

MRS periodically scans all the active databases mounted on a server to discover new move requests. The default scan interval is 15 minutes as defined in the FullScanMoveJobsPollingPeriod setting in the MRS configuration file. It is possible to adjust this setting, but great care should be taken because setting the value lower (for instance, five minutes) will consume additional CPU; setting the value higher (for instance, 60 minutes) will mean that move requests are not processed as promptly as you might wish.

When a new move request is created or when an existing job is modified, the cmdlet signals MRS that new work is available. This signal is called a tickle, and it prompts MRS to check what work is now available. Tickles happen whether cmdlets such as New-MoveRequest are run through EMS or as part of the processing of the Migration service to set up or manage migration batches. Tickles are also signaled by cmdlets such as New-MailboxImportRequest or Restore-MailboxExportRequest. When prompted, MRS scans the mailbox database associated with the request immediately without waiting for the next periodic scan, and if Workload Management reports that sufficient server resources are available to process the job, MRS picks it up and begins working. However, if the server is currently under strain, MRS creates an entry in an internal queue for the job and waits. The internal queue is checked every minute, and the jobs are processed as soon as system resources allow.

As it reviews queued jobs, MRS divides them into heavy and light workloads. Heavy jobs are those that require data to be moved, such as when mailboxes are moved from one database to another or data is imported from a PST. Light jobs don't require this kind of intensive processing because they are things like resuming a queued move request or clearing out expired move requests. To clear queues quickly, MRS processes light jobs before heavy jobs.

Details of a new move request such as the source and target databases, bad-item limit, and so on are stored in the system mailbox as elements of an XML-coded item. Apart from the move request queues, the system mailbox also stores details of current move requests that are updated as moves proceed. The data include the current stage of each move, percentage complete, bytes transferred, items transferred, timestamps, any warning information, and so on. As you see later in this section, these data are reported by the Get-MoveRequestStatistics cmdlet.

INSIDE OUT Checking MRS queues

EAC exposes no information about MRS queues so you cannot see what time MRS last checked a queue. However, you can gain some insight into what's happening with MRS queues with the Get-ExchangeDiagnosticInfo cmdlet. The example shown below queries Exchange to retrieve diagnostics data for MRS. The argument specifies that you are interested in queue information for all databases whose name matches the *DB* string mounted on server ExServer1. If you examine the output, you'll see the times for the last scan and the next scan, and you'll see the last job processed for each database.

```
Get-ExchangeDiagnosticInfo -Process MsexchangeMailboxReplication -Component
MailboxReplicationService -Argument "resources='*DB*', healthstats, queues"
-server ExServer1
```

Figure 10-1 shows the different steps that occur during the process of moving a mailbox with MRS. Move requests are usually processed in first-in, first-out (FIFO) order. However, MRS might hold a request to move a mailbox to a specific database for a short interval to allow a transient condition to pass, such as a long replication queue for the target database. When a new move request is detected, the MRS running on the Mailbox server takes responsibility for processing the move request. A number of Active Directory properties are also updated for the account that owns the mailbox to capture information about the move request, including its current status, the source and target databases, the batch name, and some flags. The properties are as follows:

- **msExchMailboxMoveBatchName (optional)** A name that identifies a group of move requests
- **msExchMailboxMoveFlags** Flags that indicate options selected for the move
- **msExchMailboxMoveRemoteHostName** The name of any remote server involved in the move
- **msExchMailboxMoveSourceMDBLink** The source database
- **msExchMailboxMoveStatus** The current status for the move
- **msExchMailboxMoveTargetMDBLink** The target database

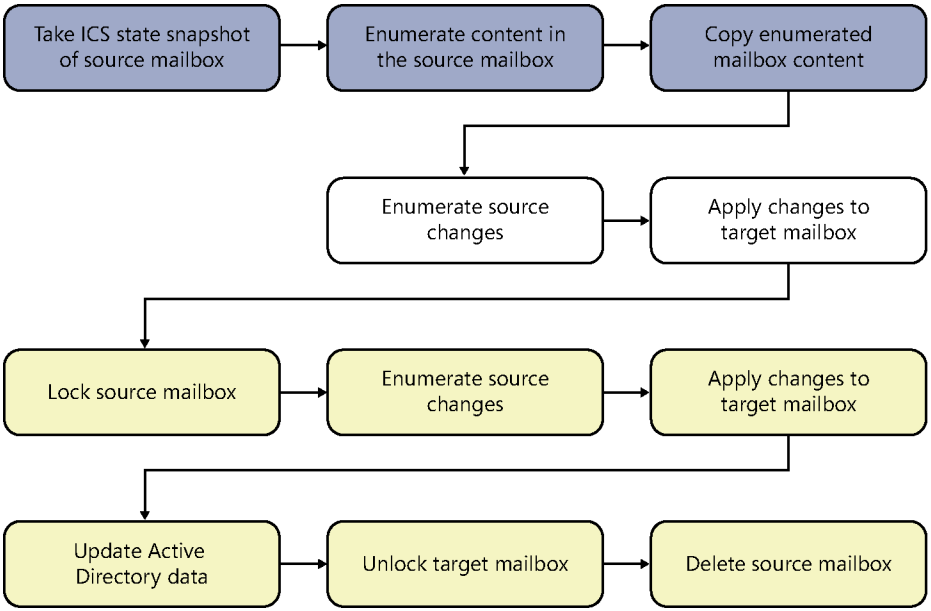


Figure 10-1 Steps in the process of moving an Exchange mailbox

MRS then begins to pull data from the source mailbox to populate the new mailbox. Data are moved in a multiphase process. MRS first does an initial seeding pass during which it enumerates all the content in the source mailbox and copies it to the target mailbox. While the initial seeding pass is progressing, MRS tracks changes that occur in the source mailbox and then performs one or more incremental updates to move any content that has been changed or added into the target mailbox to bring it completely up to date.

Items move from source database to target database continuously unless a high-availability stall condition occurs. This can happen only on servers that are part of a DAG. Exchange 2010 measured a stall in terms of replication queues. Exchange 2013 uses a more holistic view and considers factors such as the content index health, replication queues, server CPU usage, and so on. If the condition is transient, MRS can stall the move to let conditions improve. Otherwise, it might drop the move completely and put it back in the queue to be reprocessed.

CAUTION!

Remember, copying a mailbox from one database to another creates transaction logs that have to be replicated and replayed on servers that host copies of the databases. If the database remains in an unhealthy state for 30 minutes, MRS will stall the move. If conditions do not improve and the replication queues do not go down, the move might be put back in the queue to be redone when conditions are better.

MRS uses Incremental Change Synchronization (ICS) to perform incremental updates. This is the same mechanism Outlook uses to perform background drizzle synchronization when it operates in cached Exchange mode. To complete the move process, MRS locks the source mailbox when it performs the last round of incremental synchronization to make sure that no further change can occur. This step is called the finalization phase, and it can last from 10 seconds to five or six minutes, depending on the number of folders in the mailbox. The more folders, the more work MRS has to perform to check that no items arrived or were updated in each folder while the initial seeding occurred.

Mailbox content is indexed by the Search Foundation processes running on the target server as items are transferred to ensure that the new mailbox is functionally complete when MRS finishes moving everything. MRS also moves mailbox metadata such as views to provide users with the same experience with the new mailbox as they had with the original. The move request will not complete if the content indexing service is stalled, cannot process items as quickly as they arrive, or is otherwise unavailable.

MRS and system resources

An MRS instance runs on each Mailbox server and tracks the amount of processing it does. Exchange 2010 uses relatively low limits set in the MRS configuration file to control how much processing MRS can do. For example, only five concurrent moves can be processed to a target server. This approach worked to ensure that MRS did not swamp a server by attempting to process too many mailbox moves concurrently. However, the static nature of fixed limits means that MRS could not take advantage of available system resources during periods of low user demand. Conversely, it was also possible for a poorly configured MRS to impose too high a load on a server due to increased limits set in the configuration file.

The advent of Workload Management in Exchange 2013 and the increase in the intelligence available to Exchange components about the overall state of system health and available resources through Managed Availability permit MRS to use a different approach. The MRS configuration file still exists, but the limits it specifies represent a large increase over Exchange 2010 and serve as a backstop for intelligent use of resources that Workload Management monitors and controls.

Two Workload Management policies are defined for MRS. The policy used for normal work is called MailboxReplicationService, and its workload classification is discretionary, meaning that Exchange can process this workload as system resources allow. The separate MailboxReplicationServiceHighPriority policy defines its workload classification as urgent. This policy is applied when MRS has to process high-priority work such as critical mailbox moves and allows MRS to use system resources even when they are scarce.

The net effect of Workload Management is that when MRS starts processing, it does not attempt to accelerate and use all available resources. Instead, it starts by processing a single mailbox move, measures the system resources it used and the overall system health, and then increases the rate it processes, measuring all the time until the workload generated by MRS might affect other server workloads. MRS then backs off and stalls move processing until system load reduces. You know this is happening when mailbox moves report values such as StalledDueToWriteThrottle or StalledDueToHA in the StatusDetail property returned by the Get-MoveRequestStatistics command. More information is revealed in other properties, where you can find out how long a mailbox move has been held by MRS due to different conditions. The following extract is extreme because it comes when moving mailboxes on a stressed and underpowered virtual server, but it shows what MRS can do. The data shows that the biggest problem on the server was writing data to the target database because that's the cause of the largest stall condition (16 minutes, 17 seconds), with a high-availability issue causing more delay, probably due to a replication queue building for a database copy within the DAG:

```
TotalQueuedDuration                              : 00:00:02
TotalInProgressDuration                          : 00:28:58
TotalStalledDueToCIDuration                      :
TotalStalledDueToHADuration                      : 00:12:00
TotalStalledDueToMailboxLockedDuration           :
TotalStalledDueToReadThrottle                    :
TotalStalledDueToWriteThrottle                   : 00:16:17
TotalStalledDueToReadCpu                         :
TotalStalledDueToWriteCpu                        : 00:00:32
```

The process of measuring system resources and then increasing or decreasing workload is called dynamic throttling; it allows MRS to make much better use of overall system resources than it can in Exchange 2010.

By default, the limits for processing a single MRS instance perform as follows:

- 20 concurrent moves to a single target database (two in Exchange 2010)
- 100 concurrent moves to a single target server (five in Exchange 2010)
- 20 concurrent moves from a single source database (five in Exchange 2010)
- 100 concurrent moves from a single source server (50 in Exchange 2010)
- 100 moves across all databases and servers per MRS instance (same in Exchange 2010)

The difference in the limit of concurrent moves MRS will process for a single source server and a single target server gives a good indication of the relative processing load generated on these servers. You can control how MRS operates to increase or decrease the amount of concurrent activity by editing the MRS configuration file. (See the "MRS configuration file" section later in this chapter.) It is also possible to change the workload priority for the MailboxReplicationService component by running the Set-WorkloadPolicy command. However, like any tuning, it is best not to do this on a production server without conducting some detailed testing first.

Preventing loss of data

The last action MRS performs to finalize a mailbox move is to switch the Active Directory pointer to the active mailbox so that the user connects to that mailbox and begins using it. In its original implementation, MRS then cleaned up by deleting the source mailbox. This mechanism worked well when the source and target servers were healthy and no server outage happened immediately after the move completed. However, experience demonstrated that it was possible for data loss to occur if the target server hosting the newly moved mailbox suffered a lossy outage caused by a storage failure or other problem.

> **Note**
>
> **In this context, a lossy failure means that the database is recovered with some loss of data due to missing transaction logs. Often, the loss was minor, meaning that just one or two logs containing inconsequential data could not be recovered, but there were instances when mailboxes lost major chunks of information.**

The solution introduced is to take a more cautious approach. Instead of removing the source mailbox immediately following a successfully completed move, Exchange now puts the source mailbox into a SoftDeleted state. This means that the source mailbox—containing all the mailbox content—is retained until the deleted mailbox retention period expires for the database, which is typically anytime from seven days to a month and is stored in the MailboxRetention property for a database. During this time, data can be recovered from the old source mailbox by using a request generated for the New-MailboxRestoreRequest cmdlet.

Creating and managing migration batches with EAC

EAC doesn't deal with individual mailbox move requests unless you consider a migration batch that contains a single mailbox to be an individual move. Instead, the focus is on providing the framework for mailboxes to be moved in a relatively painless and automatic manner. The basic sequence is to decide which mailboxes should be moved and to where they should be moved, gather the mailboxes into a batch, and then submit the migration items (mailboxes) in the batch to MRS to be processed. More sophisticated moves take into account issues such as scheduling but only in a rudimentary fashion that requires hands-on supervision, as you'll see.

To create a new migration batch, open the Migration options in the Recipients section in EAC, click New, and select Move To A Different Database (within the organization), which causes EAC to display the screen shown in Figure 10-2. The default is to move mailboxes to databases in the same organization, but if you want to create a migration batch to move mailboxes to another forest (using a migration endpoint), you can select Move To This Forest from the drop-down list beside New.

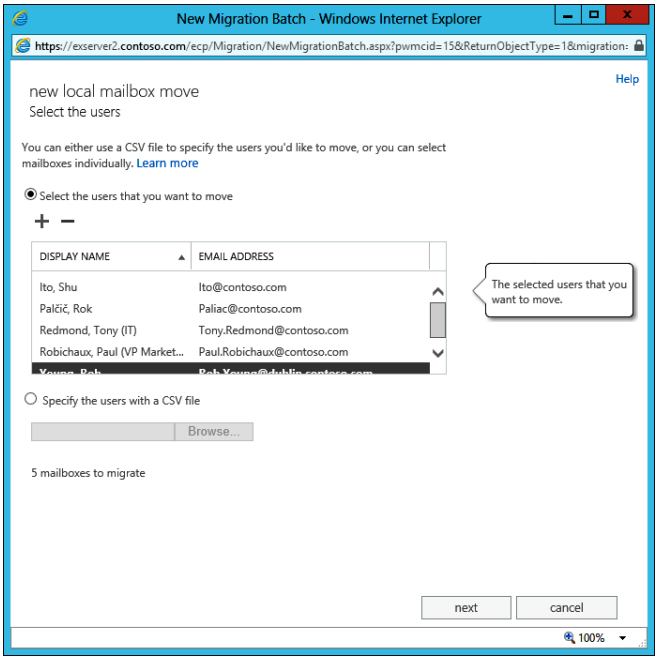


Figure 10-2 Selecting mailboxes to include in a new migration batch

The next thing to do is specify the names of the mailboxes to be moved. This is done by browsing through the address list and selecting individual mailboxes to add to the migration list or by providing the name of a comma-separated value (CSV) file containing the email addresses of the mailboxes to be moved. The first approach is perfectly acceptable for on-demand processing when only a few mailboxes have to be moved—selecting more than 20 or so items from a large list of mailboxes can become tiresome. The second is more appropriate when you have to deal with a large number of mailbox moves over a period. For example, if you wanted to move several hundred mailboxes from databases on an earlier Exchange server to a new server (or from on-premises to Exchange Online), you could create migration batches based on mailboxes belonging to certain departments, locations, or based on other criteria. The sorting capabilities of a tool such as Microsoft Excel are useful for this purpose because you can extract a CSV file from EAC that contains the complete set of mailboxes and then manipulate that data in Excel to build suitable migration batches, storing each batch in a separate CSV file that is then used as input to EAC as the basis for a new migration batch.

Figure 10-3 illustrates the next step in the process when EAC gathers information that controls the data to be moved, the target, and how to deal with specific data. This information will be fed as parameters to the move requests that are eventually created. In this instance,

you've elected to move any archive mailboxes that might exist for the primary mailboxes included in the batch.

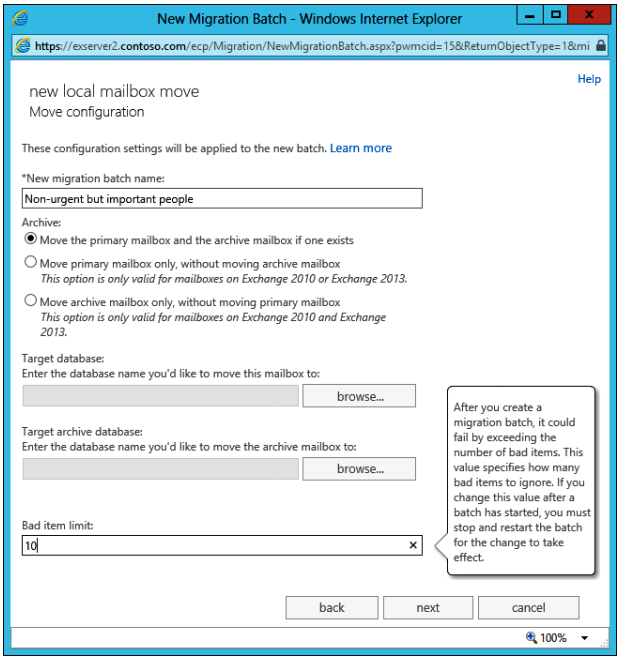


Figure 10-3 Specifying parameters for the new migration batch

The following options exist to deal with archive mailboxes:

- Move both primary mailbox and archive mailbox to the same target database.
- Move the primary mailbox to a different database, leaving the archive mailbox in the original database.
- Move the archive mailbox to a different database, leaving the primary mailbox in the original database.

The target database fields are left blank, so the Migration service can select from any active database in the same Active Directory site that is currently available for provisioning, apart of course from the database from which the mailboxes are moving. You can browse the set of available databases to create a list of target databases for the primary and archive mailboxes, in which case the Migration service selects target databases for individual moves from the lists on a round-robin basis.

The bad-item limit field is normally hidden and is exposed when you choose the More Options link. This is a common parameter for MRS operations because it is also used for operations such as mailbox import and export. In this case, the bad-item limit is set to 10; MRS can drop up to 10 corrupt (obviously bad) items per mailbox as it processes the moves. You can set the bad-item limit to zero, and MRS will stop processing a mailbox immediately after it encounters a corrupt item. Because it's obviously better to move as much data as possible to a new mailbox, one tactic often used is to set a low bad-item limit and then increase it if problems are detected during the move. You can do this as many times as necessary until the move succeeds, with the understanding that if a mailbox stores many bad items, some data loss can be expected before it can eventually be moved to a different database.

INSIDE OUT Clearing migration batches

Exchange 2010 administrators are accustomed to the fact that you cannot create a new mailbox move request if a move request (even one that is completed) exists for the same mailbox. Move requests now expire after 30 days unless they are explicitly removed by running the Remove-MoveRequest command. The same is true for migration batches; you cannot include a mailbox in a migration batch if a move request exists for the same mailbox. Therefore, it is best to remove all migration batches soon after they complete, after you have had a chance to validate that all is well with the mailboxes that have been processed. Removing a migration batch removes all the underlying move requests. Attempting to include a mailbox in a migration batch when a move request already exists is an act of folly because the Migration service simply removes the mailbox from its list before it begins to process the batch.

If you're creating a migration batch to move mailboxes into or out of the organization (remote moves), you'll also see a field to specify a large item limit. This is a new parameter in Exchange 2013. By default, all large items are transferred when a mailbox moves to a local database. This is logical because it is much easier to move large items across local networks than it is when extended connections are involved. A large item is defined as an item larger than the size defined in the transport configuration MaxSendSize value. By default, this is 10 MB, which means that MRS will not move an item larger than this unless permitted to by the administrator. Users are capable of importing much larger items into their mailbox (think of video files; the most outrageous anecdote is of a user who attempted to drag and drop a DVD into a folder), and the idea is that you should be able to control the movement of such items.

Exchange 2010 does not attempt to move large items. If MRS encountered an item larger than the MaxSendSize limit when processing a mailbox, it would halt the move. The only way you could permit large items to be moved was to increase the value of MaxSendSize, which is obviously undesirable because it affects the way the transport system works. In this case, EAC allows you to set a value for the number of large items MRS will skip when it moves a mailbox. The default is zero, meaning that MRS behaves like it did in Exchange 2010, but if you were to set a value such as 5, MRS could continue to move the mailbox because it would skip (ignore) large items up to the set limit. In other words, MRS could complete a mailbox move unless it met more than five large items, in which case the move would halt. The problem here is that although the mailbox can be moved, MRS skips items that contain a lot of data. Five 11-MB items mean that 55 MB of data might be dropped, and a move would still be deemed successful. For this reason, when moving general-purpose mailboxes, it's best to leave the large-item limit blank or zero because users want all their information to be available in the new mailbox.

If you do set a nonzero value for the large-item limit, you can discover whether any large items were dropped during the move with EAC. Select the batch, click the View Details link in the details pane, select a mailbox, and then choose Skipped Item Details. You can then see the list of any skipped items, the date, and the subject of the item. This information should be sufficient to go back to the user and help him move the large items to a PST. He can then either continue to access the large items in the PST or drag and drop the items into his newly moved mailbox. If a bad item is skipped during processing, the user can start Outlook without connecting to Exchange and examine the copy of the skipped items in his local copy of the mailbox (the OST). An item skipped as bad might not be corrupt in the OST and therefore be recoverable. Recovered bad items can be exported to a PST and then dragged and dropped into the new mailbox to make them available online.

You can also discover information about large items by running an EMS command to interrogate the LargeItems element of a move-mailbox report:

```
$(Get-MoveRequestStatistics -IncludeReport -Identity "JSmith").Report.LargeItems
```

Or even:

```
$(Get-MigrationUserStatistics -Identity JSmith@contoso.com).SkippedItems
```

The next step is to finish creating the migration batch by deciding the sequence of processing and how the Migration service notifies you of progress. The first part of the screen, shown in Figure 10-4, specifies whether the Migration service sends email to anyone when it completes processing the batch. By default, the user who creates the batch is entered here, but you can browse to add recipients to or remove the original recipient from the list.

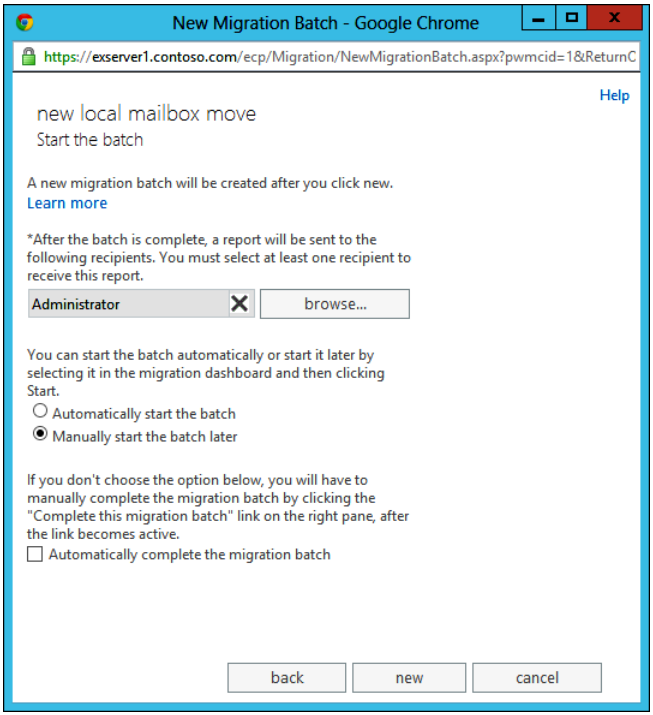


Figure 10-4 Completing details for a new migration batch

The next thing to decide is whether the Migration service should begin processing the batch immediately or wait for the administrator to release the batch at a more appropriate time. It is common practice to stage batches of mailbox moves to occur at times of low server demand such as the weekend. To do this, you just create whatever batches are needed to contain the mailboxes you want to move and choose Manually Start The Batch Later. Splitting a large number of mailboxes into a number of smaller batches creates more manageable chunks of work. For example, if you want to move 1,000 mailboxes over a weekend, it is better to create 10 batches, each containing 100 mailboxes, than one massive batch, if only because you can then launch the batches at phased intervals so MRS can process the mailboxes at a measured pace.

INSIDE OUT Use a meaningful bad-item limit

You can enter a value range from 0 (zero) to 2,147,483,647 for the bad-item limit. Exchange 2010 uses a value of –1 to indicate that you don't care how many corrupt items MRS encounters. Exchange 2013 replaces –1 with the Unlimited string. It is a

really bad idea to use an unlimited bad-item limit because then a move can succeed no matter how many apparently bad items are encountered. It's good to know this information, and it's even better to be able to investigate and deduce a likely reason if you find that a mailbox includes many bad items, so a more measured approach is to set the bad-item limit to something reasonable, such as 10, and review the move report afterward to see whether any bad items were in fact encountered. Remember that if a move fails because it exceeds its bad-item threshold, you can always investigate the problem and then increase the bad-item limit to allow MRS to resume the move.

EAC doesn't include a scheduling engine, so you have to intervene manually to release a paused batch and allow it to start processing. To do this, select the batch from the set listed and click the Start icon. The Migration service will then begin to process the mailboxes contained in the batch.

The last setting controls whether the Migration service will instruct MRS to autocomplete the mailbox moves in the batch. Autocomplete means that MRS performs all the necessary processing to move the mailbox to its new database and then immediately switches the Active Directory pointers for the mailbox so that the user can begin to access the new mailbox. Sometimes, administrators prefer to have the chance to check whether the initial synchronization of mailbox contents has proceeded satisfactorily (for example, that an excessive number of bad items was not encountered) before they finalize moves. Others like mailbox moves to proceed over the week and then switch the mailboxes at the weekend when people are not working. In either instance, MRS performs a final synchronization to copy any changes that have occurred in the source mailbox since the completion of its initial synchronization to bring the new mailbox completely up to date before it finalizes the move. Remember that the Migration service performs an incremental synchronization every 24 hours to make sure that it tracks changes to the source mailbox, so the final synchronization should be short and sweet.

MRS allows you to leave a mailbox move in an autosuspended state for up to 30 days, and this is what happens if you move a mailbox by running the New-MoveRequest command. The Migration service provides additional control. First, the Migration service normally performs an incremental synchronization every 24 hours to keep a mailbox move fresh. (You can stop the autosynchronization by stopping a migration batch when it is in a Synced state.) This prevents the problem that basic mailbox moves encounter if they are left suspended for a long period—in effect, so much change might have occurred in the source mailbox that a complete resynchronization might be required to complete the move. Second, although the Migration service keeps mailbox moves fresh and ready to complete, if 60 days elapse without any administrative intervention (for example, stopping, starting, or changing batch parameters), the Migration service stops the batch and ceases

performing autosynchronization. If another 120 days elapse without administration intervention (that is, 180 days total), the Migration service removes the batch and cancels all the pending mailbox moves.

After all the parameters are entered, the Migration Batch Wizard creates the new migration batch and returns to EAC (Figure 10-5). This section of EAC is sometimes referred to as the migration dashboard because it displays information known about current migration batches. Note that the status of the new batch is Created, indicating that the Migration service has not yet begun processing the mailboxes and will not start until an administrator releases the batch for processing by clicking the Start button (highlighted).

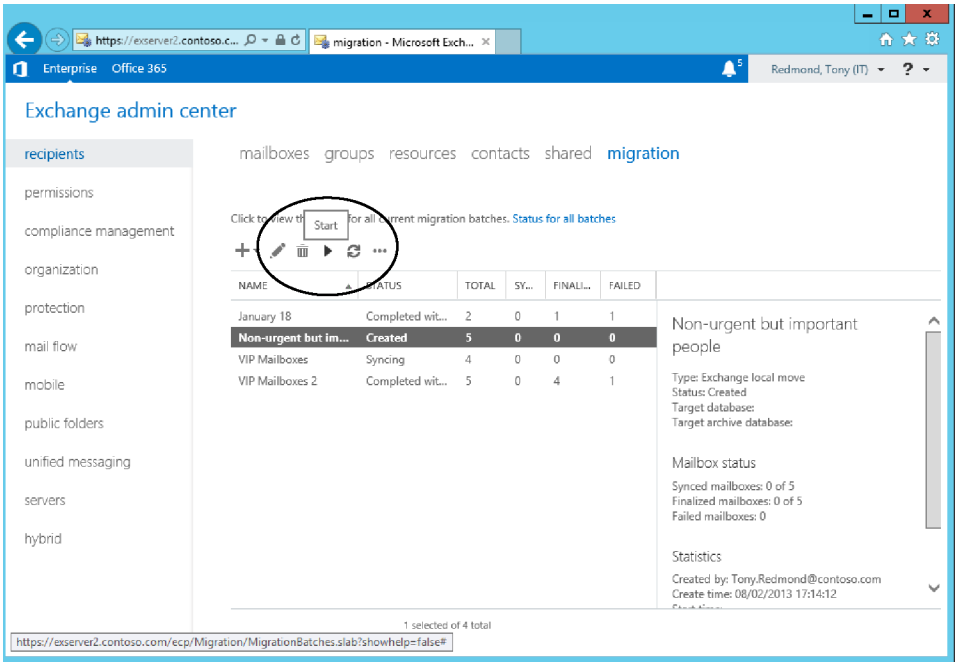


Figure 10-5 The new migration batch appears in EAC

EAC asks whether you really want to start the batch. Upon confirmation, the Migration service starts to direct new mailbox move requests to MRS, which then processes the mailbox moves as system resources allow. The status of the batch changes to Syncing. Processing continues in the background until all the move requests in the batch have reached the point at which MRS is allowed to proceed (completion or autosuspend).

Behind the scenes, the Migration service first breaks up the migration batch into a series of migration users, each of which is created as a separate item in the migration arbitration mailbox. You can think of a migration user as a work item the Migration service translates

into a mailbox move request that it then manages. The difference between a migration user (manipulated with the *-MigrationUser commands) and the underlying move request (manipulated with the *-MoveRequest commands) is that the migration user object holds a set of properties the Migration service uses to control the flow of the migration, including steps such as the daily autosynchronization. You know when a move request has been created for a migration user because its status changes from Queued to Syncing. The primary identifier for a migration user is his email address, a reasonable choice given that you don't tend to migrate users multiple times when you move mailboxes from on-premises servers to the cloud. However, a mailbox might be moved multiple times within an on-premises deployment, and it's therefore important to clean up migration batches and remove the migration user objects. If you include a user in a migration batch when she is present in another batch, the Migration service detects the presence of an existing migration user and drops her from the batch, flagging the problem as a Validation Warning for the batch.

You can view the progress of the individual mailbox moves in a batch by choosing View Details under Mailbox Status in the right-side details pane. This reveals a screen similar to that shown in Figure 10-6. Seven migration users are visible in this batch. The selected user has been successfully migrated, and EAC reports that the mailbox size was 97.56 MB. Clicking the Download The Report For This User link requests EAC to fetch the full migration move report for the mailbox. The content of this report is discussed in the "Accessing move report histories" section later in this chapter. Two migration users are in a Failed state. Sometimes a Failed state is due to some catastrophic event such as a component failure, but it's also possible to see a failure reported for the very good reason that an administrator attempted to move a mailbox to the database in which it is currently located. I know because I have made this mistake.

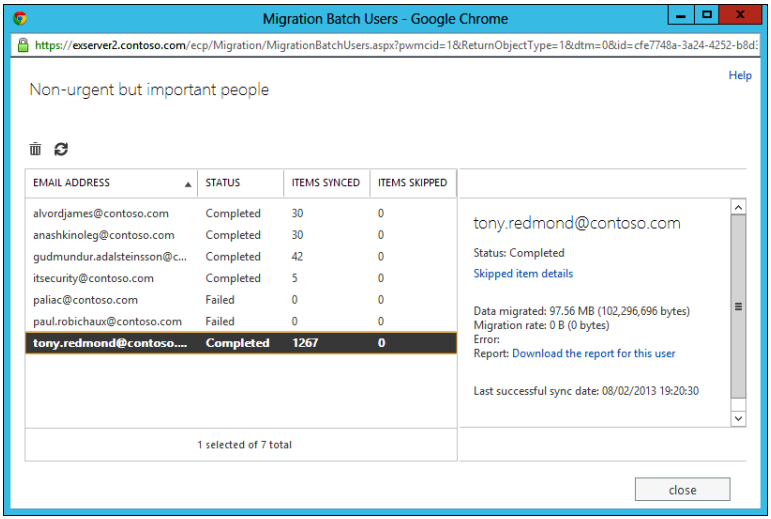


Figure 10-6 Viewing migration batch details

If you need to include additional mailboxes in a batch, you can do so by stopping the batch, selecting its properties, opening the migration mailboxes page, and then uploading a CSV file that contains the details of the mailboxes you want to add to the batch. The Migration service will process the information in the CSV file and add the users to the batch after the batch is restarted. Any user already in the batch is updated with details in the file. Note that new values for settings such as target database and archive database will be respected only if the Migration service was unable to direct the move request to MRS, as when you opt to move a mailbox to the database in which it is already present. They will not be used if MRS is already processing a mailbox. However, the Migration service picks up updates for settings such as bad-item limit or large-item limit and applies them when the batch restarts.

Eventually, the Migration service completes processing of the mailboxes in the batch and updates its status to Synced (Figure 10-7), and the recipient chosen to receive notification sees a new message telling her that initial synchronization is complete, which is a prompt to check EAC to see whether anything else needs to be done. The subject of the message varies if errors have occurred, and the message will contain links to two reports, one containing a list of all successful users and another containing a list of failed users. If the batch is completely successful, the link for the second report is not shown.

However, because you chose to autosuspend the moves, the new mailboxes have to be finalized by allowing MRS to perform a final synchronization and switch the Active Directory pointers. This is done by choosing Complete This Migration Batch. After you are prompted to confirm the decision, MRS updates the status for the batch to Completing and starts to finalize the mailboxes.

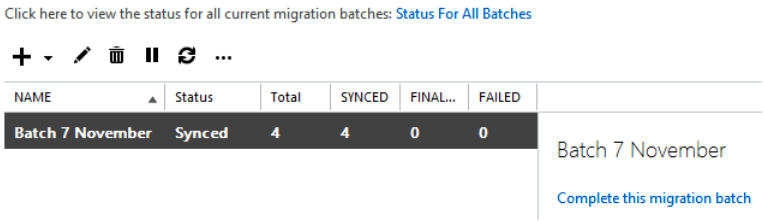


Figure 10-7 Completing a migration batch

The process outlined here is for local moves—that is, moves between databases in the same Exchange organization. Much the same process is used to move mailboxes between different Exchange organizations that share a federated trust or from Exchange on-premises to Exchange Online in a hybrid deployment with the caveats that more preparation is required to set up the migration batches, the moves will take longer due to the sometimes undependable nature of external connections, plus other factors such as throttling imposed by Exchange Online.

INSIDE OUT Oops I made an error—intervention required

Assume that you create a migration batch and release it for processing. MRS creates all the necessary move requests and begins to move mailboxes. Then you find that you've included some mailboxes you didn't intend to move. Fortunately, EAC allows you to remove a mailbox after a batch has started processing. To do this, select the batch, stop it, and then click the View Details link to expose the set of mailboxes being processed. Select the mailbox to remove and then click the wastebasket icon (shown in Figure 10-6).

Removing the problematic mailbox from the batch is the right way to solve the problem because it forces Exchange to clean up matters behind the scenes. It is a mistake to stop the batch and then run the Remove-MoveRequest command to remove the move request for the mailbox. The move request will be deleted, but it will leave a gaping hole in the migration batch that will cause EAC to flag an error (Figure 10-8) because one of the move requests in the batch no longer exists. The other mailboxes will be processed to completion and won't experience any problems. However, the migration batch itself won't be able to complete because the move request you've just removed will remain in a Syncing state—in effect looping while it searches for its missing move request—until it eventually fails. You can resolve matters by removing the migration user with the Remove-MigrationUser command, but it's much better to use the simple and right approach in the first place.

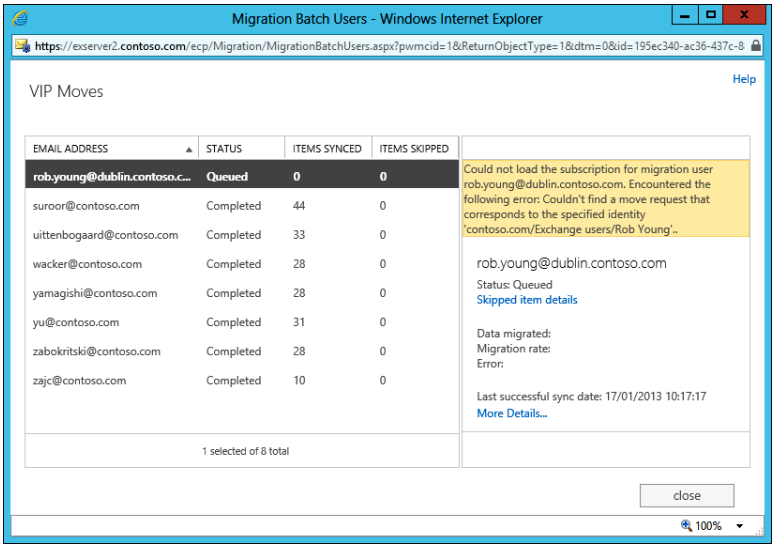


Figure 10-8 An error after removing a move request

TROUBLESHOOTING

Migration batch logs

MRS notes the history of each mailbox move and stores entries in the mailbox. Mailbox move reports can be extracted from these entries to help understand exactly what happens during a move. The Migration service also logs its activity, but this information is stored in the \Exchange Server\V15\Logging\MigrationLogs directory on the Mailbox server that hosts the migration arbitration mailbox. You won't need to look at these logs on a regular basis, but they can be useful if problems occur with migration batches, if only to provide to Microsoft Support personnel to help troubleshoot the problem.

Using EMS with migration batches

Exchange 2013 mailbox migrations are created using the New-MigrationBatch cmdlet. The input to New-MigrationBatch is a CSV file containing the names of the mailboxes to be moved. The CSV file is created by EAC or by an administrator. It can specify moves to occur between databases in an on-premises deployment, between Exchange deployments in different Active Directory forests (these moves are always initiated from the target forest), from an on-premises deployment to a cloud platform such as Office 365 (a process that Microsoft calls on-boarding), or indeed from the cloud to an on-premises deployment (logically, off-boarding). The cmdlet can also be used to create batches of mailboxes to be migrated from an IMAP server to Office 365.

The CSV file contains a single column and starts with an EmailAddress line. Each of the lines thereafter contains the SMTP email address of a mailbox to be moved. The purpose of the email address is to enable MRS to locate the mailbox. Although any valid addresses assigned to a mailbox can be specified, it's best to select one (PrimarySmtpAddress is recommended) and use that consistently everywhere. For example:

```
EmailAddress
Tony.Redmond@contoso.com
EPRedmond@contoso.com
Andy.Ruth@contoso.com
…
```

This file is the simplest kind of input that the Migration service will use. The lack of any other columns means that the Migration service should use the default values for any option it takes, such as which target database to use. If you want to exert more control, you can add these columns:

- **TargetDatabase** Specifies the target database for the primary mailbox. This database is used as a target for archives if the TargetArchiveDatabase is not defined for the batch.
- **TargetArchiveDatabase** Specifies the target database for the archive mailbox (if one exists).
- **BadItemLimit** Specifies the bad-item limit to use for the mailbox move.
- **MailboxType** Can be PrimaryOnly, meaning that only the primary mailbox is moved; ArchiveOnly, meaning that only the archive mailbox is moved; or PrimaryAndArchive, meaning that both are moved.

After the input file is created, processing is initiated by running the New-MigrationBatch cmdlet to read the contents of the CSV file and create the underlying move requests. For example:

```
New-MigrationBatch –Name 'Executive Moves' –Local –CSVData
([System.IO.File]::ReadAllBytes("C:\MailboxMoves\Executives.CSV"))
–TargetDatabases DB1, DB2
```

This command causes the Migration service to create a new batch called Executive Moves to move mailboxes between databases in the local organization (indicated by the Local switch). The name of the CSV file is on the local disk of the workstation where the command is run. Often, it will be more convenient to position these files on a Universal Naming Convention (UNC) file share. The TargetDatabases parameter tells the Migration service the names of the databases it can use as targets for the mailbox moves. The effect of this command is to initiate the migration batch (in a Created state), but the associated move requests are not created. The Migration service will not begin to process the mailbox moves until the Start-MigrationBatch cmdlet is run. For example, to instruct the Migration service to begin processing a batch of mailbox moves that have been created, use this command:

```
Start-MigrationBatch –Identity 'Executive Moves'
```

Alternatively, you can specify the AutoStart parameter to the New-MigrationBatch command to instruct the Migration service that it can begin processing the mailbox moves as soon as the move requests have been created.

Batched mailbox moves are autosuspended after the initial copy of data to the databases that host the new mailboxes. This enables administrators to review the status of the migration and decide whether it is safe to allow the Migration service to complete the process of moving the mailboxes. The work done to copy a mailbox to the point at which MRS autosuspends a move is roughly 95 percent of the process. The work that remains to complete the move is (a) to perform a final incremental synchronization to capture details of any changes that have been made to the source mailbox since the move process began; this

ensures that the target mailbox is completely up to date when the move completes, and (b) to switch the Active Directory pointers Exchange and its clients use to locate the current copy of the mailbox.

Logically, the Complete-MigrationBatch cmdlet tells the Migration service that it is okay to complete a batch of mailbox moves. For example:

```
Complete-MigrationBatch -Identity 'Executive Moves'
```

Of course, you might decide that the mailbox moves should run through to completion without any need for administrator oversight. In this case, you can add the AutoComplete parameter to the New-MigrationBatch command. Thus, a command to start a new batch of mailbox moves that will run from the time the command executes through to completion would look something like this:

```
New-MigrationBatch -Name 'Executive Moves' -Local -CSVData
([System.IO.File]::ReadAllBytes("C:\MailboxMoves\Executives.CSV"))
-TargetDatabases DB1,DB2 -AutoStart
-AutoComplete -NotificationEmails "Admin@contoso.com"
```

In this instance, you've also specified that you want to know when the Migration service completes processing the batch. The NotificationEmails parameter enables you to provide the Migration service with one or more email addresses to which it should send a status report detailing what happened during processing. This parameter is not required, and if it is omitted, the Migration service won't send a report anywhere. However, you can always check individual mailbox move reports if you need this information. The logical completion to the creation of a migration batch is to remove it when processing is done. This is accomplished with the Remove-MigrationBatch command:

```
Remove-MigrationBatch -Identity "VIP Moves"
```

It's possible to adjust settings for a migration batch with the Set-MigrationBatch command. Ideally, this should be done when the batch is in a Created, SyncedWithErrors (for instance, some mailboxes have included too many bad items), or Stopped state so that the new settings can be applied to the mailboxes that are either waiting to be processed or in the middle of being processed. For instance, this command sets new limits for bad items and large items and instructs the Migration service to notify two users when the batch completes.

```
Set-MigrationBatch -Identity "VIP Moves" -BadItemLimit 20 -LargeItemLimit 5
-NotificationEmails "Tony.Redmond@contoso.com", "Administrator@contoso.com"
```

During the migration process, you can view the current progress by issuing the following commands:

```
Get-MigrationUser | Get-MigrationUserStatistics
```

The Get-MigrationUser cmdlet returns details of a specific mailbox that is in the process of being migrated. Normally, the mailbox is identified so Exchange knows the mailbox in which you are interested. For example:

```
Get-MigrationUser –Identity 'Tony.Redmond@Contoso.com'
```

In this case, you pipe the set of mailboxes to the Get-MigrationUserStatistics cmdlet to output the desired information.

Reporting mailbox moves

It's all very well to move mailboxes around, but it's even better to know when the work has been done. Small mailboxes will likely move quite quickly unless there are a large number of mailboxes in the queue, either in migration batches or queued individually, but as mailbox sizes swell, the time required to move increases, even between databases on the same server. For this reason, migration batches allow you to nominate people to receive email notifications when a batch is ready to complete (in other words, suspended, ready for the final synchronization and Active Directory switch to happen) or after it is completely finished.

Figure 10-9 shows an example of an email notification. In this case, you can see that the migration batch has finished its initial synchronization. Notice the link at the bottom of the message. Clicking the link downloads a CSV file from the server that processed the batch containing some statistics for the mailboxes processed as part of the batch. The CSV file is shown open in Notepad and, as you can see, the information provided is terse and omits data such as the size of each mailbox. A second link, to a Failed Users Report, appears if any failed moves exist in the batch.

The reports generated when batches are ready to complete and have completed are also available by viewing the properties of a migration batch through EAC (Figure 10-10). Although only one report is shown for this migration batch, it is possible for several reports to be available if the batch has stopped and restarted a number of times; for instance, after the limit for bad items was reached and needed to be adjusted.

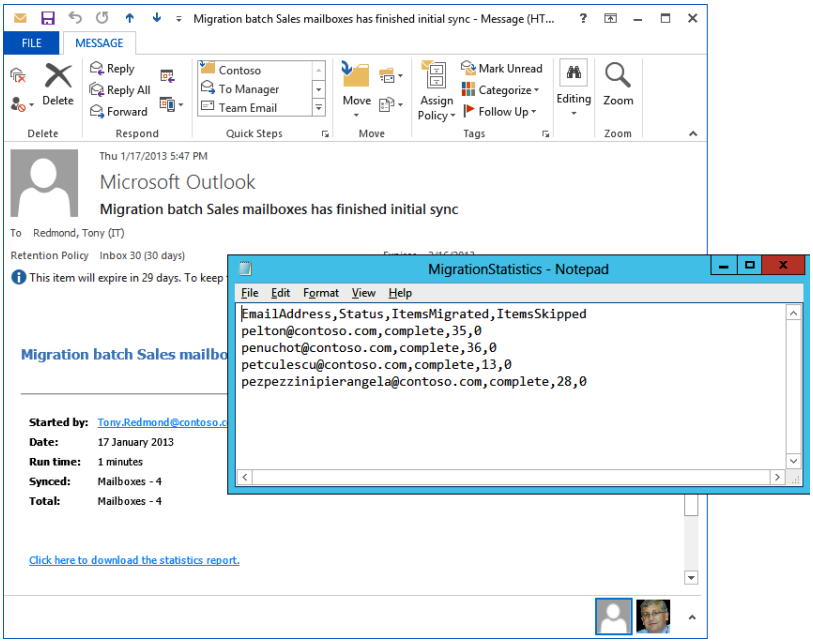


Figure 10-9 Viewing an email notification about a migration batch

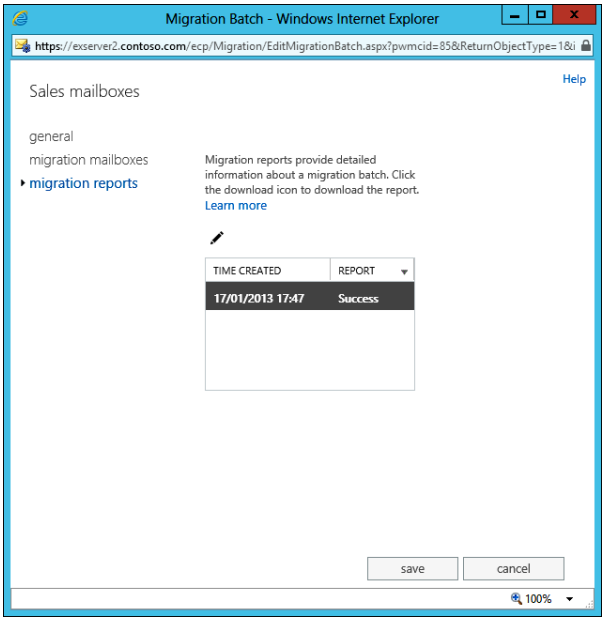


Figure 10-10 Viewing the available reports for a migration batch

Getting more information about a move

As a graphical interface, EAC is always limited in terms of the quantity of information it can display. The information EAC displays about a mailbox that moves as part of a migration batch reveals whether it completed or failed and how much mailbox data was moved (or not), but it does not really say very much about what actually happened during the move. To get that information, you have to download the migration report for the user (an option shown in Figure 10-6) or plunge into EMS and retrieve the data from the move report MRS compiles and stores in the mailbox when the move is over.

A move report is best consulted when a move is completed or when it has failed for some reason. How to get this information is discussed further in the "Accessing move report histories" section later in this chapter. First, during the move, it's often good to know how much of the mailbox has been transferred. To do this, you can use the Get-MoveRequestStatistics cmdlet. For example:

```
Get-MoveRequestStatistics –Identity 'Smith, Jeff' | Select DisplayName,
Status, TotalMailboxSize, TotalMailboxItemCount, PercentComplete, BytesTransferred,
ItemsTransferred, BytesTransferredPerMinute
```

The TotalMailboxSize property reported by Get-MoveRequestStatistics includes content in the enhanced Recoverable Items folder. You are most interested in the size because it determines the additional storage required in the target database. It also determines the number of transaction logs that will be generated on the target server during the move—you can anticipate that a 200-MB mailbox will generate at least two hundred 1-MB transaction logs. By contrast, no additional transaction logs are immediately generated on the source server because the old mailbox will be removed during online background maintenance, and the delete operations performed then will be interleaved with the other transactions caused by this processing. However, you might observe growth in a source database size as mailboxes are moved from it. This seems counterintuitive, but it is explained if the source mailboxes were not used by Outlook configured in cached Exchange mode. In this case, the source mailbox does not contain any indexes or views that are otherwise created by Outlook, and MRS will be forced to create them to perform its incremental synchronization at the last part of the mailbox move. Depending on the available white space in the source database, the creation of these views can cause the database to grow and impose some additional I/O demand on the source server.

If you're interested, the split between the storage occupied by items in folders (including items in the Recoverable Items folder) and deleted items in a mailbox can be viewed with the Get-MailboxStatistics cmdlet. For example:

```
Get-MailboxStatistics –Identity 'Smith, Jeff' | Select DisplayName, ItemCount,
TotalItemSize, DeletedItemCount, TotalDeletedItemSize
```

The percentage of move complete reported is an approximation based on the size of moved messages rather than the number of items. The reported transfer speed (bytes transferred per minute) is based on a sample taken every second and averaged over the last 20 seconds. Treat this value as a rough guide to how things are moving rather than a very accurate measurement of network speed. After a mailbox move is complete, Exchange reports the total items transferred, the size of the moved mailbox, and the total number of transferred bytes. This value is invariably larger than the actual mailbox size because of a certain amount of transmission overhead involved in the copy plus some metadata about the mailbox that is transferred along with the content.

INSIDE OUT Who drives a mailbox move?

For any mailbox move, one MRS process is responsible for moving the data from the source to the target mailbox. You can tell which MRS is responsible for a move by looking at the move request properties by running the Get-MoveRequest command and then examining the Direction property. If it is Pull, MRS on the target server is responsible for moving the data. If Push, MRS on the source server is directing operations. In this context, target server means the server on which the active copy of the target database is mounted, and source server means the server on which the active copy of the database holding the source mailbox is mounted. Generally speaking, the following happens:

- **Exchange 2013 to Exchange 2013: MRS on target server**
- **Exchange 2013 to Exchange 2010: MRS on Exchange 2013 server**
- **Exchange 2013 to Exchange 2007: MRS on Exchange 2013 server**
- **Exchange 2010 to Exchange 2013: MRS on Exchange 2013 server**
- **Exchange 2007 on Exchange 2013: MRS on Exchange 2013 server**

The exception is when an Exchange 2013 source server runs a higher version of the software than an Exchange 2013 target server. In this instance, the source server drives the transfer based on its more recent software being aware of any new processing that has to be performed.

TROUBLESHOOTING

There was a problem—what happened?

If you encounter a problem with a move, your first step is to run the Get-MoveRequestStatistics cmdlet with the IncludeReport parameter to extract some information about what happened during the move. This is the most common way to find out what kind of problem occurred during a move.

```
Get-MoveRequestStatistics -IncludeReport -Identity 'Tony.Redmond@contoso.com'
```

Accessing move report histories

Three commands can be run to retrieve mailbox move history information. As shown in Table 10-1, the differences between the three commands are the source from which the information is fetched and the content that is covered. In reality, the three commands deliver the same information for a completed mailbox move that is part of a migration batch. The Migration service generates a set of data that should correlate with the data MRS generates. At least, that's the way it should work for a given mailbox.

TABLE 10-1 Sources of mailbox move information

Command	Retrieves	Stored in
Get-MoveRequestStatistics	Information about a current mailbox move	System mailbox of the source database
Get-MigrationUserStatistics	Information about the processing of a mailbox as part of a migration batch	Migration user object in migration arbitration mailbox
Get-MailboxStatistics	Mailbox move history reports	Move report items in the mailbox root; number stored determined by the MRS configuration

The Get-MoveRequestStatistics command obtains an overview of what happens when MRS processes a mailbox move in terms of the number of items moved, the total size of the items, and how long the move took. By using the Get-MigrationUserStatistics command, you can fetch similar information as the Migration service processes users in a migration batch. These commands can only be used to fetch information about a mailbox as long as the underlying move request or migration user objects are in place. After these objects are removed, the commands can't be used to fetch move history information for the mailbox.

A more detailed report, showing the processing steps that were performed during a mailbox move, can be generated with the Get-MailboxStatistics command. This report shows how MRS goes through the steps of initializing a move and enumerating mailbox content, the initial synchronization, and, finally, completion. Collectively, this information is called

a move report, and it is stored in the non-IPM subtree of the mailbox when a move completes. By default, Exchange 2013 stores the last five move history entries in each mailbox, and the entries remain available even if all move requests for the mailbox are removed. If necessary, you can change the number of move history entries Exchange retains by updating the value specified in the MRS configuration file to instruct MRS to hold between zero and 100 reports. Remember that the MRS configuration file is unique to a Mailbox server, so if you want to have the same behavior everywhere, you have to make the same change on every Mailbox server.

You can extract two types of reports from the history kept for a mailbox move. Both reports are based on the mailbox move history entries, and the difference is purely in the amount of detail each contains.

Basic history is provided if you pass the IncludeMoveHistory parameter.

A much more detailed report is generated if you pass the *IncludeMoveReport* parameter.

To generate basic history for a mailbox move request, you can use the following code:

```
Get-MailboxStatistics -Identity 'CEO Mailbox' -IncludeMoveHistory | Format-List
MoveHistory
```

```
MoveHistory             : {(1/7/2013 22:14:45: TargetMDB=VIP, Size=1.4 GB
(1,503,471,581 bytes), Duration=02:55:24), (12/11/2012 00:47:33: TargetMDB=DB2,
Size=814.8 MB (854,397,936 bytes), Duration=00:40:50)}
```

Here you can see that the latest move (7 January 2013) was to the VIP database when the mailbox size was 1.4 GB, and the move took 2 hours and 55 minutes to complete. The second move listed was on 11 December 2012 to database DB2 and took 40 minutes, most likely because the mailbox was a good deal smaller at the time, and the servers were under less of a load.

INSIDE OUT Output the move report to a file

Even the basic history report includes a reasonable amount of information. For this reason, it's easier to review the information by writing the move report to a file that you can then open with a text editor. The following code extracts the history for the most recent move request for a mailbox and pipes the output to a text file that is then opened with the Notepad editor.

```
$BasicReport = (Get-MailboxStatistics -Identity 'CEO Mailbox'
-IncludeMoveHistory).MoveHistory
```

```
$BasicReport[0] | Out-file -FilePath 'c:\Temp\MRS-History.Log' | Notepad
'c:\Temp\MRS-History.Log'
```

In this context, [0] refers to the first (more recent) move report stored in the user's mailbox. If you want to view details of the second (older) move report, you can do so by referencing it as $BasicReport[1]. Up to five move reports are kept for a mailbox, so you can refer to them as [2], [3], [4], and so on. Now generate the full move report.

```
$FullReport = (Get-MailboxStatistics -Identity 'CEO Mailbox'
-IncludeMoveReport).MoveHistory
$FullReport[0] | Out-file -FilePath 'c:\Temp\MRS-Report.Log' | Notepad
'c:\Temp\MRS-Report.Log'
```

On the surface, Exchange seems to generate much the same file, but using the -IncludeMoveReport parameter forces Exchange to output a lot of additional information in the Report section after the statistical information, which helps you understand exactly what actions MRS performs to move a mailbox.

The following information is extracted from a move mailbox report to highlight the most important stages in a move request. Go through the information you can find in the report. The report begins with MRS acknowledging the move request:

```
18/01/2013 10:18:16 [Exserver2] 'contoso.com/Users/Administrator' created move
request.
```

If the target database is on an Exchange 2010 or Exchange 2013 server, MRS uses XSO for the connection. If the target database is on an Exchange 2007 server, MRS uses a MAPI.Net connection. The globally unique identifier (GUID) shown for the target mailbox is the ExchangeGuid, which uniquely associates the user object with the mailbox. The server name that hosts the active copy of the database is also recorded along with the version of Exchange that it runs. If the mailbox has an archive, you also see which database is selected as a target for the new archive.

```
18/01/2013 10:18:21 [Exserver2] Connected to target mailbox 'Primary
(0e875eca-e804-4008-b1ad-45e0cdb5075e)', database 'DB3', Mailbox server
'EXSERVER2.contoso.com' Version 15.0 (Build 516.0).
```

Logically, MRS must connect to the source mailbox and its archive (if present). You can now see that this move is between two databases on the same server. Note the version number, indicating that the server runs Exchange 2013 release to manufacturing (RTM):

```
18/01/2013 10:18:21 [Exserver2] Connected to source mailbox 'Primary
(0e875eca-e804-4008-b1ad-45e0cdb5075e)', database 'DB2', Mailbox server
'EXSERVER2.contoso.com' Version 15.0 (Build 516.0).
```

MRS now enumerates the content of the source mailbox. FAI means folder-associated items, hidden items used for features such as mailbox rules. If an archive is enabled, its contents are also enumerated.

```
18/01/2013 10:18:21 [Exserver2] Source mailbox information:
Regular Items: 1107, 83.05 MB (87,080,044 bytes)
Regular Deleted Items: 3, 19.46 KB (19,924 bytes)
FAI Items: 17, 40.48 KB (41,451 bytes)
FAI Deleted Items: 0, 0 B (0 bytes)
```

MRS now initializes the folder hierarchy in the target mailbox:

```
18/01/2013 10:18:23 [Exserver2] Stage: CreatingFolderHierarchy. Percent
complete: 10.
18/01/2013 10:18:23 [Exserver2] Initializing folder hierarchy in mailbox
'Primary (0e875eca-e804-4008-b1ad-45e0cdb5075e)': 41 folders total.
18/01/2013 10:18:25 [Exserver2] Folder hierarchy initialized for mailbox
'Primary (0e875eca-e804-4008-b1ad-45e0cdb5075e)': 41 folders total.
```

The first checkpoint is created:

```
18/01/2013 10:18:25 [Exserver2] Stage: CreatingInitialSyncCheckpoint. Percent
complete: 15.
```

MRS prepares to transfer items by counting the messages it has to move. The number reported here (1127) is the total of regular items (1107), items in the Recoverable Items folder (3), and FAI items (17):

```
18/01/2013 10:18:26 [Exserver2] Stage: LoadingMessages. Percent complete: 20.
18/01/2013 10:18:26 [Exserver2] Messages have been enumerated successfully.
1127 items loaded. Total size: 83.07 MB (87,109,327 bytes).
```

The MRS process running on the target server now starts to pull data from the source mailbox in a constant stream. This stage is usually the longest of any in the process, but it completes quickly because the mailbox is small. No further checkpoints are taken because the copying lasted less than a minute:

```
18/01/2013 10:18:26 [Exserver2] Stage: CopyingMessages. Percent complete: 25.
18/01/2013 10:18:26 [Exserver2] Copy progress: 0/1127 messages, 0 B (0
bytes)/83.07 MB (87,109,327 bytes).
18/01/2013 10:19:12 [Exserver2] Initial seeding completed, 1127 items copied,
total size 83.07 MB (87,103,932 bytes).
```

Now the target mailbox is populated with all the information that existed when MRS enumerated the source. However, new items might have been created in the mailbox since the move began, so MRS checks to see whether anything has changed in the source mailbox to know whether it should perform an incremental synchronization to bring the target mailbox completely up to date. If present, the same check is performed for the archive. If required, an incremental synchronization is then done.

```
18/01/2013 10:19:12 [Exserver2] Stage: IncrementalSync. Percent complete: 95.
18/01/2013 10:19:12 [Exserver2] Changes reported in source 'Primary (0e875eca-
e804-4008-b1ad-45e0cdb5075e)': 0 changed folders, 0 deleted folders, 0 changed
messages.
18/01/2013 10:19:12 [Exserver2] Incremental Sync 'Primary (0e875eca-e804-4008-
b1ad-45e0cdb5075e)' completed: 0 changed items.
```

MRS is now in the final stage of moving the mailbox. The mailbox is locked temporarily against further changes to allow a final copy of any content that has recently changed in the source mailbox. Per-user read/unread data is copied (which tells clients which items have been read), and the contents of the target mailbox are verified to ensure that everything is present.

```
18/01/2013 10:19:13 [Exserver2] Stage: FinalIncrementalSync. Percent complete:
95.
18/01/2013 10:19:13 [Exserver2] Copying per user read/unread data.
18/01/2013 10:19:13 [Exserver2] Per user read/unread data copied successfully.
18/01/2013 10:19:13 [Exserver2] Stage: FinalIncrementalSync. Percent complete:
95.
18/01/2013 10:19:13 [Exserver2] Verifying mailbox contents...
18/01/2013 10:19:14 [Exserver2] Mailbox contents verification complete: 10
folders, 1127 items, 83.07 MB (87,103,932 bytes).
```

MRS now updates the database that holds the mailbox in Active Directory to make the switch effective. When Active Directory is updated, the user will connect to the new mailbox, and the move is basically complete.

```
18/01/2013 10:19:15 [Exserver2] Mailbox 'CEO Mailbox' was loaded from domain
controller 'ADRoot.contoso.com'.
18/01/2013 10:19:18 [Exserver2] Mailbox was updated using domain controller
'ADRoot.contoso.com'.
```

The final stage in the move process is to clean up the old source mailbox. This is done by marking the source mailbox as SoftDeleted, meaning that it will be retained in its database and can be reconnected if something catastrophic happens to the mailbox in its new location.

```
18/01/2013 10:19:18 [Exserver2] Move has completed and final cleanup has
started.
18/01/2013 10:19:18 [Exserver2] Source mailbox 'Primary (0e875eca-e804-4008-
b1ad-45e0cdb5075e)' was successfully cleaned up after the move.
```

INSIDE OUT Checkpoints

As MRS moves items from the source to the target mailbox, it creates frequent checkpoints to enable it to resume the move from the last checkpoint if it is interrupted for any reason. MRS supports online moves from Exchange 2007 to Exchange 2010 or Exchange 2013, and checkpoints work for these moves; they don't when you move mailboxes to Exchange 2007 from Exchange 2010 or Exchange 2013. This is because the Exchange 2007 Information Store supports online moves only when it is the source rather than the target. Offline moves are used whenever MRS moves data back to an Exchange 2007 server, and, in this case, data is streamed from source to target until completion, and no checkpointing is done. Any interruption due to network outages, storage exhaustion, or other conditions forces MRS to restart the move from the beginning.

Managing individual mailbox moves

Behind the scenes, the Migration service translates a migration batch into the individual move requests that actually move a mailbox's content from a source database to a target database and then transfer pointers so that clients can locate the new mailbox. EMS enables you to exert more programmatic control over move requests than the migration batches that are created and managed through EAC and is a good option to use when you have to manage more complex move requests or integrate mailbox moves with other processing such as rebalancing load across available mailbox databases.

Consider the earlier discussion about setting a value for the large-item limit for moves in a migration batch. It was concluded that this should be kept at zero when moving general-purpose mailboxes because users really prefer MRS to retain as much of their mailbox data as possible. However, if you had to deal with a situation in which you had to move a mailbox to a remote forest that is intended to hold items larger than the limit set by *MaxSendSize*, you could use EMS to create a new move request for the mailbox and set the *AllowLargeItems* parameter to $True. This instructs MRS to ignore the normal maximum item size when moving this particular mailbox.

The cmdlets communicate with MRS through TCP port 808 and are as follows:

- **New-MoveRequest** This cmdlet initiates a new move request and performs the processing to ensure that a mailbox can be moved from its existing location to a different database. The request begins in a Queued To Move state and moves through In Progress, meaning that data are being replicated from the source to the target mailbox, and then AutoSuspended, meaning that you can complete the request.
- **Get-MoveRequest** This cmdlet provides information fetched from Active Directory about a move request and tells you how it is progressing.
- **Remove-MoveRequest** This cmdlet cancels a move request that is queued, in progress, suspended, or failed. Any data that have been replicated to the target mailbox are removed. Remove-MoveRequest also removes a completed move request from the list MRS maintains.
- **Suspend-MoveRequest** This cmdlet suspends further processing of a move request until the request is released by an administrator.
- **Resume-MoveRequest** This cmdlet resumes processing for suspended move requests, including move requests that are queued with the SuspendWhenReadyToComplete flag.
- **Get-MoveRequestStatistics** This cmdlet fetches data about move requests from the system mailbox.

To initiate a move request for a mailbox, you just identify the mailbox to move. You do not have to select a target database unless you want MRS to move the mailbox to a specific location. If you don't specify a target database, MRS uses the Exchange database provisioning feature to select one from the set currently mounted and healthy in the same Active Directory site. If the mailbox has an associated archive, it is also moved.

```
New-MoveRequest -Identity 'Tony.Redmond@contoso.com'
```

New-MoveRequest initiates the request and puts it into a queued status. As you know, MRS is running on every Mailbox server. The MRS processes running on the Mailbox servers hunt for new queued move requests for mailboxes held in databases mounted on the server and begin to process queued requests soon after they are created. An individual MRS handles as many requests as it can, subject to available server resources.

INSIDE OUT Ensure that the quota is sufficient for the target database

The earlier Move-Mailbox cmdlet offered one advantage over the new move request cmdlets in that you can pass the PreserveMailboxSizeLimit parameter to have Exchange respect the existing mailbox quotas on a mailbox when it is moved. In other words, if the mailbox has a quota of 1 GB in the source database, it is given a 1 GB quota in the target database even if the default quota for mailboxes is 100 MB. The New-MoveRequest cmdlet doesn't support a similar parameter, so you have to ensure that sufficient quota is available in the target database; otherwise, a move request will fail. Typically, this is done by raising the default quota for the database before starting to move mailboxes and making whatever adjustments are necessary afterward. In some instances, you might leave the default database quota as it is; in others, you might set certain elevated quotas on specific mailboxes and reduce the default database quota to control all the other mailboxes.

INSIDE OUT Move requests autocomplete

As soon as MRS has moved all the mailbox content to the target database, it updates Active Directory to reflect the new location and to tell the CAS to redirect any future client connections. The exception is when a move request is created with the SuspendWhenReadyToComplete parameter because this instructs MRS to move the mailbox contents but not to switch the mailbox pointer in Active Directory until authorized by an administrator.

Assigning move priority

Exchange 2010 assigns the same priority to every mailbox move. Exchange 2013 uses a more sophisticated form of workload management than the simple throttling in Exchange 2010, which enables MRS to exert finer control over move requests. If you don't do anything to set a priority for a move request, MRS gives it Normal priority. The full set of priorities is Emergency (highest), Highest, Higher, High, Normal, Low, Lower, and Lowest. If you cannot find a suitable priority from that set, you are a very demanding person.

All move requests are kept in the same queue. MRS uses priority along with system resource availability and the load factors for target databases and servers to assess which move request should be picked off the queue for processing. Move requests with Normal priority or below are processed according to the default MailboxReplicationService workload policy, whereas MRS can use the MailboxReplicationServiceHighPriority workload policy with higher-priority requests, the net effect of which is that MRS can continue processing high-priority moves even if the server is under strain due to demand for resources.

Knowing that you can create an emergency move request to have a mailbox put at the top of the queue for processing is a useful feature that might just come in handy someday. Then again, you might never have to use it.

Moves and mailbox provisioning

If you move mailboxes between Active Directory forests, you'll discover that the New-MoveRequest cmdlet requires a fully provisioned Active Directory object in the target forest against which to work. In the past, you could move a mailbox to a server in another forest (for example, as part of a migration project from Exchange 2003 to Exchange 2007 when you create a new forest for Exchange 2007), and Exchange would create the Active Directory account, move the mailbox, and then update the necessary attributes to establish a fully provisioned user account and mailbox. If your account provisioning procedure depends on this functionality, you have to change it to make sure to create the Active Directory account before attempting to move a mailbox. The reason for the change is that Microsoft found it too difficult to allow for all the unique scenarios that exist around account provisioning. It could have continued to write code that accounts for all these circumstances, but concluded that it would be easier and more effective to have Exchange take care of moving mailboxes and Active Directory procedures take care of everything else. It's up to you to define what those Active Directory procedures should be to meet the needs of your company.

Handling move request errors

By default, MRS tolerates up to 60 transient errors that force it to retry to copy data before it aborts an attempted mailbox move. These errors are different from the declared

threshold for bad or large items MRS encounters in the source mailbox that can be tolerated before MRS is forced to fail a move. Errors that occur during moves must be consecutive. If progress is made between errors, MRS resets the error count and proceeds. You can use the Resume-MoveRequest cmdlet to resume an aborted move, and MRS will restart where it failed. Most errors are short-lived and are due to conditions such as temporary network glitches between the source and target servers. You can check the event log for event 1101 for MSExchange Mailbox Replication to see whether errors occured. (Bad items are also noted in the move report.) The information reported isn't tremendously helpful, but it might indicate a likely cause.

Tip

You should check the event logs on both source and target server. If you do see errors in the event log, you can increase the diagnostic level for Mailbox Replication to high or maximum to increase the amount of information generated in the event log and see whether that helps pinpoint the problem.

Each move request includes a bad-item limit MRS can tolerate during the move. As explained earlier, the default bad-item limit is zero; any problem with an item will force MRS to stop the move. There is no real point in moving corrupt items from database to database, and it is good to cleanse a mailbox by removing corrupt content during the move. Any mailbox that has been in use for more than a few years, especially if it was used with Exchange 2000 or Exchange 2003 or was migrated from a different email system, contains the possibility of some lurking corrupt items. It's true that if you allow MRS to skip the corrupt items, you create some potential for data loss. However, if an item is corrupt anyway, it probably isn't very valuable, hasn't been accessed in some time (otherwise, the user would have complained about it), and could cause a mailbox to be placed into quarantine if the corrupt item causes the Store to consume excessive resources.

You can always check the move request report afterward to determine whether any corrupt items were encountered in a mailbox. If a move fails because it exceeds the bad-item limit, you can always restart the move by setting a higher bad-item limit with the Set-MoveRequest cmdlet. For example:

```
Set-MoveRequest –Identity 'JSmith' –BadItemLimit 100
```

MRS then restarts the move at the point at which it failed and, hopefully, goes on to complete it. If a large number of items are rejected by the target server, one solution is to identify the problem items from the move report, export them to a PST, and then delete them from the source mailbox. You can then move the rest of the mailbox as normal and attempt to import the problem items from the PST into the new mailbox.

Mailboxes that have high numbers (more than five or so) of bad items indicate that the source database has experienced problems in the past, and this is an issue that deserves further investigation. Of course, if you know that a database has had problems (usually due to hardware failure) and don't have a reliable backup copy to restore, you might not care how many corruptions MRS encounters as long as it manages to move the majority of a mailbox to a new database. In other words, you're willing to accept some data loss to recover a mailbox.

Moving a mailbox to a database that has a single copy is a straightforward operation. Things are slightly more complicated when you move a mailbox from a database that has multiple copies (in a DAG) because Exchange must ensure that it can recover from a failure that occurs on the primary (active) copy of the database during the move operation. To do this, MRS copies all the content from the source mailbox to the new mailbox and then uses the Data Guarantee API to check that the *DataMoveReplicationConstraint* requirement (set with the Set-MailboxDatabase command) for the database is satisfied.

Exchange reports problems with target databases as errors in the event log, where you'll see indications such as Some Database Copies Are Behind, so it is relatively easy to check for these conditions. A healthy server that is participating properly in log replication will have no difficulty meeting these conditions because it should be copying logs from active databases and replaying them into its database copies on an ongoing basis. Any buildup of transaction logs to be replayed or logs queued to be copied indicates a potential replication problem or that too much write activity exists for the target server to handle gracefully. When MRS is convinced that the databases are healthy, it releases the move to continue to completion. MRS treats these high-availability (HA) stalls as a transient failure; if a stall lasts for more than 30 minutes, the regular MRS failure logic is invoked, and the move is marked as failed due to too many transient failures. See the "Ensuring high availability" section later in this chapter for more information on this topic.

Checking and altering move request status

To check the status of a move request, pass the name of the mailbox to the Get-MoveRequest cmdlet:

```
Get-MoveRequest –Identity 'Tony.Redmond@contoso.com'
```

If you make a mistake when creating a move request, you can cancel the move only if the move request is in Queued or Moving status or if it is autosuspended pending completion. In other words, as long as MRS is occupied with moving data and hasn't reached the stage of switching mailbox pointers in Active Directory, you can cancel a move.

However, if the move request has reached the finalization stage, you can't cancel it because MRS is updating Active Directory to point to the new mailbox. In this situation, you must

wait for the first move to complete and then remove the move request before creating a new move request to move the mailbox to the right database.

TROUBLESHOOTING

I can't remove a mailbox database

Exchange won't allow you to remove a mailbox database if any move requests for the database have not been cleared. This prevents the removal of a database if any danger exists that a mailbox move is in progress or a planned move has not completed successfully and requires administrator attention.

Clearing move requests

You cannot create another move request for a mailbox until you remove any previous move request. This rule exists whether you move mailboxes in migration batches or as individual move requests. A command like this will remove a move request:

```
Remove-MoveRequest –Identity 'Tony.Redmond@contoso.com'
```

When you remove a move request, you also remove the attributes from the mailbox's Active Directory object that MRS uses to track the progress of a move request. This indicates to MRS that it is safe to proceed with a new move request.

Being forced to remove old move requests before creating another for the same mailbox can be confusing because it is reasonable to assume that it should be possible to schedule a new move request immediately for a mailbox if the previous move was successful. However, Exchange keeps the move request in place to provide a form of audit data to show that the mailbox was moved in the past. If you don't need to keep the move request, you can delete it with the Remove-MoveRequest cmdlet. For example, to look for move requests that completed successfully and then remove them, you would run the following command:

```
Get-MoveRequest –MoveStatus   "Completed" | Remove-MoveRequest
```

Exchange 2010 move requests linger on until they are removed. Exchange 2013 introduces the *CompletedRequestAgeLimit* parameter for the New-MoveRequest and Set-MoveRequest commands to enforce an age limit for move requests. If none is specified, MRS keeps a move request for 30 days and then removes it. The Migration service uses the same default for the move requests that it creates in migration batches.

Migration batches introduce a little more complexity into the question of how best to remove move requests. Every batch is represented by a set of move requests. If you remove all move requests without removing batches (or waiting for batches to complete), you run

the risk of having batches that can't tie back to the underlying move requests, which is not a good situation. Removing a migration batch removes all the associated move requests.

Therefore, although it is a good idea to check for lingering move requests on a regular basis and remove them if necessary to allow a new move request or migration batch to run, you do need to carefully consider which requests are removed. Think before deletion is the order of the day.

Tip

It's a good idea to check the status of a move request after it is finished to ensure that it finished successfully and not with the *CompletedWithWarning* status. Moves marked with a *CompletedWithWarning* status have met a problem that might affect the mailbox, so you should extract a full mailbox move report and examine it to discover the warnings MRS reports.

Migration endpoints

Up to this point, the discussion has been about moving mailboxes between databases in the same organization. These types of moves are called *local* and are the easiest to manage because they occur within a single authentication context. In other words, all the Exchange servers involved in the move share the same Active Directory settings because they are part of the same organization.

As shown in Table 10-2, apart from local moves, Exchange 2013 supports three other types of mailbox moves, each with its own characteristics but all sharing the requirement that an endpoint must be created to allow the move to proceed. An endpoint sets up the necessary authentication context to enable MRS to connect to source and target mailboxes and move contents. An endpoint can point to the source from which MRS will fetch mailbox contents or the target on which MRS will create the new copy of the mailbox. After it is created, an endpoint can be provided as a parameter to the New-MigrationBatch cmdlet. You do not have to create a migration endpoint before you can move mailboxes between databases in the same organization. Endpoints are required only when mailboxes are migrated to or from an external source.

TABLE 10-2 Exchange 2013 move mailbox endpoints

Endpoint type	Meaning
None (local)	Between databases in the same organization
ExchangeRemoteMove	Between databases in different Active Directory forests or between on-premises and hybrid servers (hybrid configuration)
ExchangeOutlookAnywhere	From an on-premises Exchange deployment to the cloud (either cut-over or staged)
IMAP	From an IMAP4 email server to Exchange Online (Office 365)

Endpoints can be created through EAC or by running the New-MigrationEndpoint command. When it is created, the endpoint can be managed through EAC or by running the Set-MigrationEndpoint command. However you choose to manage endpoints, they must be in place before they can be used with migration batches. As an example, this command uses the New-MigrationEndpoint command to create an endpoint for mailboxes moving to or from the Contoso organization:

```
New-MigrationEndpoint -Name "Contoso EndPoint" -ExchangeRemoteMove -Autodiscover
-EmailAddress 'Tony.Redmond@contoso.com' -Credentials (Get-Credential
Contoso\Administrator)
```

The ExchangeRemoteMove parameter identifies the kind of migration that is being performed. The Autodiscover parameter tells MRS to query Autodiscover for the domain of the specified mailbox to fetch connection settings such as the CAS with which to communicate during the move. The EmailAddress parameter specifies an email address within the domain where the mailboxes are located that the Autodiscover service can use when it determines connection settings. Finally, the Credentials parameter tells the credentials for a suitably privileged account that can connect to the remote Exchange organization and access the mailbox data to be moved. In this case, use the Get-Credentials cmdlet to prompt for the password for the Contoso\Administrator account.

When this command is run, Exchange attempts to use Autodiscover to connect to the remote organization by using the supplied credentials. After a successful connection is made, details of the endpoint are stored and can be reused when MRS needs to move mailboxes to or from the remote organization.

Enabling MRSProxy

MRSProxy is the Mailbox Replication Proxy service. This is a web service that runs on CASs and is designed to enable mailbox moves in and out of the organization. The intention behind MRSProxy is that it provides a safe and secure method to transfer mailbox data

between different Exchange organizations without the need to expose Active Directory or Exchange endpoints externally. Inside the boundary of an organization, MRSProxy communicates with MRS to transfer mailbox information as required. In Exchange 2013, MRSProxy essentially redirects incoming connections from the CAS to the Mailbox server that hosts the database for either the source or target mailbox, depending on which is local.

MRSProxy is not enabled by default on an Exchange 2013 CAS, so you need to enable the service on all CASs that are involved in remote mailbox moves before commencing any moves. Figure 10-11 shows an error that is often encountered when attempting to access the MRSProxy webpage on a server.

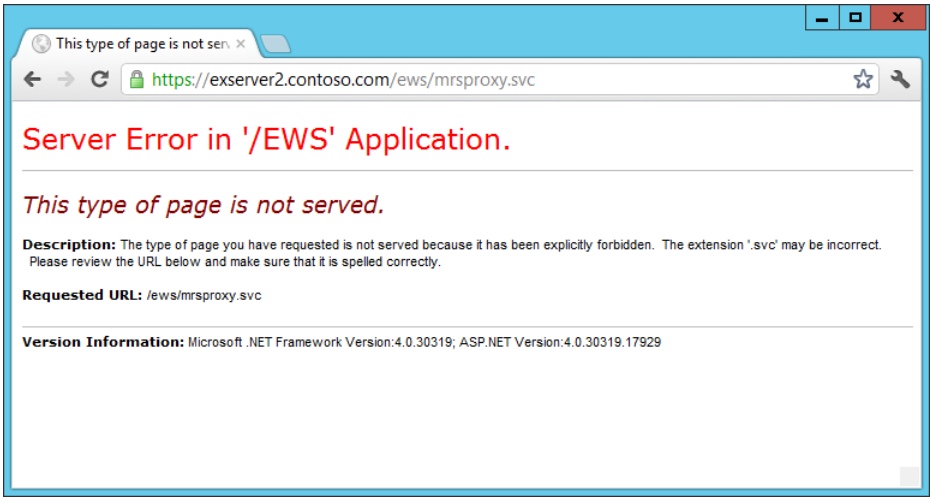


Figure 10-11 Error seen when MRSProxy is not enabled

You can run the following command to enable MRSProxy on a server:

```
Get-WebServicesVirtualDirectory -Server Exserver2.contoso.com |
Set-WebServicesVirtualDirectory -MRSProxyEnabled $True
```

After the command is run, you need to recycle the *MSExchangeServicesAppPool* pool with IIS Manager to make the new setting effective. When the pool is reset, you can test that MRSProxy is working successfully with the following command:

```
Test-MigrationServerAvailability -ExchangeRemoteMove -RemoteServer ExServer2.contoso
.com -Credentials (Get-Credential)
```

The Test-MigrationServerAvailability cmdlet tests a target server to ensure that it can support out-of-organization mailbox moves (to another organization or Exchange Online, or to move data from an IMAP4 server to Exchange Online). In this instance, you connect to the target server to validate that the server is capable of supporting remote Exchange mailbox

moves. You should see similar output to that shown in Figure 10-12. If not, a problem still exists with the MRSProxy configuration that needs to be investigated. Recycling the *MSExchangeServicesAppPool* usually cures the problem.

```
Machine: Exserver2.contoso.com
[PS] C:\temp>Test-MigrationServerAvailability -ExchangeRemoteMove -RemoteServer exserver2.contoso.com  -Credentials (Get
-Credential)

cmdlet Get-Credential at command pipeline position 1
Supply values for the following parameters:
Credential


RunspaceId          : 3dc39ebc-d2e0-4388-9c2c-513b42f36a3e
Result              : Success
Message             :
ConnectionSettings  : <ExchangeConnectionSettings HasAdminPrivilege="True" HasAutodiscovery="False" HasMrsProxy="False"
                      AutodiscoverUrl="" IncomingEmailAddress="" IncomingRPCProxyServer="exserver2.contoso.com"
                      IncomingExchangeServer="exserver2.contoso.com" IncomingNSPIServer="" IncomingDomain="contoso"
                      IncomingUserName="administrator"
                      EncryptedIncomingPassword="7A+v3GgfGcbMF4gl0xPFz2MbS9WJURqAXbbxE2i9GFvAgRNqQBHAiuiIgvG37yGB"
                      IncomingAuthentication="Basic" ServerVersion="" TargetDomainName="" />
SupportsCutover     : False
ErrorDetail         :
IsValid             : True
Identity            :
ObjectState         : New


[PS] C:\temp>
```

Figure 10-12 Validating that MRSProxy is working correctly on a server

Planning mailbox moves

It is common practice to initiate a set of mailbox moves to happen overnight when user activity and network traffic are light. You should be careful not to swamp the network with many concurrent move requests for large mailboxes that conflict with other network-intensive activities such as nightly backups. You should also distribute moves across all available servers and databases to minimize the chance of swamping a target server with the load from multiple incoming move requests, all of which generate significant CPU and I/O demand to populate the new mailboxes and update content indexes. Another issue to consider is the limits MRS imposes on moving information to different mailboxes and servers. For example, if you want to move 1,000 mailboxes to decommission a Mailbox server, MRS processes the moves faster if you spread the moves across different target servers and databases. By comparison, if you select a single mailbox database as the target, you can queue the 1,000 move requests, but MRS will process only five at a time, and the operation is likely to last much longer than you expect.

When you create migration endpoints, you can control the maximum number of mailboxes the Migration service attempts to move through an endpoint by setting the value of MaxConcurrentMigrations with EAC or by running the New-MigrationEndpoint or Set-MigrationEndpoint commands. This manages the load generated by moves to and from Exchange Online when operating in a hybrid organization or when processing migration batches between two Exchange organizations. It does nothing to manage mailbox moves within the organization, whether created as part of migration batches or as individual move requests.

Another tactic often used in migrations is to stage mailbox migrations. You schedule mailbox moves to occur a week or so before the actual mailboxes will be switched. MRS performs the initial synchronization of the new mailboxes and then pauses, waiting for an administrator to allow the final switchover to happen. In the interim period, administrators can check the mailbox move reports to make sure that nothing untoward happened when MRS copied the mailboxes. For instance, they might check what bad items were dropped and try to figure out why they were dropped.

Scheduling mailbox migrations in this manner is common practice when moving users to Office 365 and is why Microsoft originally developed the Migration service to provide the framework for batched migrations. Moving large quantities of mailbox data across the Internet to a Microsoft data center does not happen fast, so it makes sense to perform the initial mailbox synchronization well in advance of a planned move. Then, when you are ready to switch the mailboxes to use Office 365, the amount of data that has to be synchronized should be a small fraction of the total. Of course, the longer you leave mailboxes after the initial synchronization, the more change will occur in the source mailboxes, so the longer the final synchronization will take. For this reason, although you can leave mailboxes in a Suspend When Ready To Complete state for up to 30 days, it's wise to restrict the period between initial synchronization and final move to less than a week whenever possible. Using migration batches to move mailboxes rather than creating individual move requests helps because the Migration service performs a daily incremental synchronization for its moves to keep the mailboxes fresh and ready to complete.

Suspending mailbox moves

Exchange 2013 allows you to hold mailbox moves in migration batches or individually. When you create a migration batch, you have the option to start the batch processing automatically as soon as it is created or hold it until a more suitable time. You also have the option to complete the batch automatically. These options are visible in Figure 10-13. When you elect to start a batch manually, Exchange will not create the move requests until you release the batch for processing by clicking the Start button when the batch is selected in EAC.

When you choose not to let a migration batch automatically complete, Exchange creates the move requests, and MRS processes them to a point when all the mailbox content is copied and the only steps remaining are to perform a final synchronization and update the user object in Active Directory. After the mailboxes are copied, administrators can check the mailbox move reports to determine whether it is safe to proceed to completion. When ready to complete, an administrator can choose Complete The Migration Batch as shown earlier, in Figure 10-7, and MRS will complete processing.

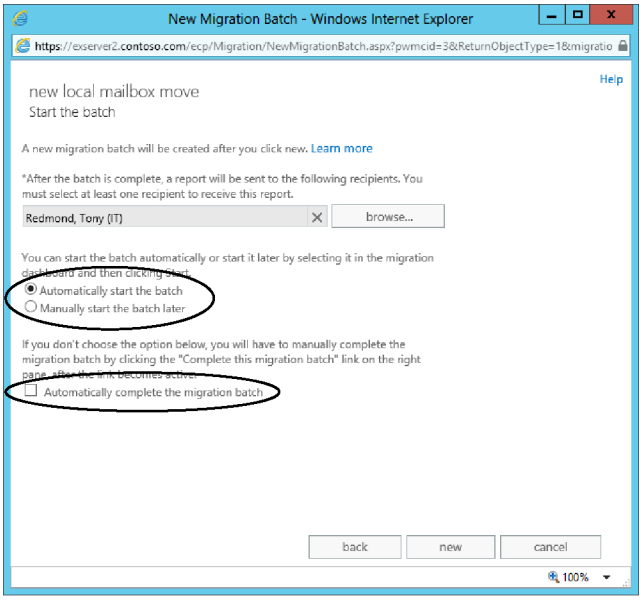


Figure 10-13 Options to control how a batch is processed

If you create individual move requests, you can do much the same. Exchange supports two parameters for move requests, *Suspend* and *SuspendWhenReadyToComplete*:

- *Suspend* tells MRS to create a queued move request but do nothing more until the move request is resumed. This is a good way to create a set of move requests to be processed as a batch, perhaps at night when user demand is low.
- *SuspendWhenReadyToComplete* tells MRS to perform the initial copy and move all the current information from the source mailbox to the target. When an administrator is ready to complete the move, he can release the move request.

For example, this command creates a new autosuspended move request:

```
New-MoveRequest -Identity 'CEO Mailbox' -SuspendWhenReadyToComplete -TargetDatabase
'DB1'
```

After MRS has performed the initial synchronization, it places the move request in the AutoSuspended state. You can verify this by examining the properties of the move request as follows (the output is edited for clarity):

```
Get-MoveRequestStatistics -Identity 'CEO Mailbox' | Format-List
```

```
UserIdentity                    : contoso.com/Exchange Users/Tony Redmond
DisplayName                     : Tony Redmond
Status                          : AutoSuspended
SyncStage                       : IncrementalSync
Flags                           : IntraOrg, Pull, Suspend,
SuspendWhenReadyToComplete
SuspendWhenReadyToComplete      : True
PercentComplete                 : 95
Message                         : Informational: The move request for
mailbox c612d271-5618-408a-81ef-50f1b3f9a5f7 is ready to complete and has been
automatically suspended because the SuspendWhenReadyToComplete parameter is set
to $true.
```

You can see from the output of the Get-MoveRequestStatistics cmdlet that the move is paused at a notional 95 percent of the mailbox. The last 5 percent is the incremental synchronization that will bring the mailbox completely up to date when you release the move request with the Resume-MoveRequest cmdlet.

```
Resume-MoveRequest -Identity 'CEO Mailbox'
```

Tip

Creating a migration batch or a set of suspended mailbox moves for later processing by MRS is a good way to schedule moves without any impact on users. When user demand is low outside normal working hours, the batch or individual move requests can be released to complete, and the users can access their relocated mailbox the next time they connect.

Assume that you have created a set of suspended move requests that are now ready to complete. To release all the suspended move requests, you'd use something like the command shown next. (Move requests that are created with the *SuspendWhenReadyToComplete* parameter go into a status of AutoSuspended when they are past the initial seeding phase.)

```
Get-MoveRequest -MoveStatus 'AutoSuspended' | Resume-MoveRequest
```

You can use much the same approach to suspend queued move requests that are in progress if you need to stop move requests for some reason such as to apply a hotfix or to avoid a backup window:

```
Get-MoveRequest -MoveStatus 'Queued' | Suspend-MoveRequest -SuspendComment 'Paused
move to allow application of hot fix'
```

Another way of scheduling moves is to let MRS create some move requests and place them in the queue but not do anything else until you are ready to proceed. This might be done

when you want to queue a set of move requests that you will release when system load is low. To do this, specify the –Suspend parameter with the New-MoveRequest cmdlet. For example:

```
New-MoveRequest –Identity 'Tony Redmond' –TargetDatabase 'DB1' –Suspend
```

Because they usually require less processing than other move requests, resumptions receive a certain priority in that MRS processes them before any other queued requests. However, they will wait until other move requests that are in progress complete. You can use this command to retrieve information about moves that are currently in progress:

```
Get-MoveRequest –MoveStatus 'InProgress'
```

You can build on this code to pipe information about the requests to the Get-MoveRequestStatistics cmdlet to discover which Mailbox servers are processing current move requests.

```
Get-MoveRequest -MoveStatus 'InProgress' | Get-MoveRequestStatistics | Format-List
Alias, MoveServerName
```

The Get-MoveRequest cmdlet retrieves move request data about a mailbox from Active Directory. This cmdlet accesses only the small set of properties maintained in a mailbox to describe a move request, whereas the call to Get-MoveRequestStatistics queries the MRS instance that is responsible for the move to retrieve comprehensive data about the current status of the move such as percentage complete, the number of items to move, and so on.

Move requests are queued in the target database, and MRS processes the requests in FIFO order. You can use the Get-MoveRequestStatistics cmdlet to view the move request queue from a database as follows:

```
Get-MoveRequestStatistics -MoveRequestQueue –Database 'DB1'
```

You can use the Get-MoveRequest cmdlet to examine the current moves known to a database.

```
Get-MoveRequest –Database 'DB1'
```

Mailbox moves from Exchange 2007 do not move the contents of the Recoverable Items structure (deleted items awaiting retention time expiry). As explained in Chapter 11, "Compliance management," Exchange 2010 and Exchange 2013 use a different approach because to meet legal retention requirements, Exchange includes the Recoverable Items structure when mailbox moves occur between Exchange 2010 and Exchange 2013 servers.

For the same legal retention reason, Exchange 2010 and Exchange 2013 can move archives and the contents of their Recoverable Items structure. Clearly, the nature of archives is that they are likely to be larger than the associated primary mailboxes, so if you use Exchange

archive mailboxes and force users to move items into the archive with retention policies or manually, you have to expect that mailbox moves will take longer.

Mailbox moves: Speed can vary

The speed of mailbox moves varies depending on system configuration and load. CPU can be a real bottleneck due to the way mailbox data are indexed as they arrive on a target server. There's no doubt that a lightly loaded server with a fast storage subsystem and multiple CPUs processes moves faster than a heavily loaded server running on a virtualized platform. Directing multiple concurrent moves at a single mailbox database also limits throughput.

Experience to date indicates that the sweet spot is to plan for between 5 and 10 concurrent moves per target server, spreading the moves across multiple databases on different disks to ensure that the I/O load generated does not affect high availability or cause throttling to occur. Attempting to process more concurrent moves will encounter high CPU usage, usually caused by the context-indexing process attempting to handle all the new items as they are transferred into databases. You can expect to process from 4 to 6 GB of mailbox data hourly, so it is realistic to expect that a 2 GB mailbox will take between 15 and 30 minutes to move. Most of the improvement in move-mailbox performance is due to the changes made to the Exchange Store over the last two versions, such as decreased I/O demand. In comparison, Exchange 2007 usually processes mailbox moves at a rate of 600 to 800 MB per hour.

A move request can be cancelled at any time by running the Remove-MoveRequest cmdlet before it enters the finalization stage. When you cancel a move request, Exchange stops transferring content to the target mailbox and then deletes the target mailbox. The contents that have been transferred when the move stopped will be cleaned from the target database immediately. The source mailbox is unlocked by removing the move request, and it reverts to its original condition. The user can continue to use it as before. If you need to move the mailbox again, you have to start from the beginning and cannot pick up from a point reached before.

Ensuring high availability

Mailbox moves performed within a DAG have to take account of the replication activity that underpins the ability of the DAG to deliver high availability. Exchange includes two checks to reduce the possibility of data loss when mailboxes move between replicated databases on DAG members.

At points during the mailbox move, including at the end before Active Directory is updated with the new mailbox location, MRS queries Active Manager to determine whether replication is in a healthy state (less than 10 logs to be replayed and 3 to be copied) for the target mailbox database and its passive copies. The action MRS takes depends on the value of the *DataMoveReplicationConstraint* property of the target mailbox database. (You can set this value with the Set-MailboxDatabase cmdlet.) The following values are used:

- **None** The move operation continues as if the target database is a standalone (non-replicated) database, and Active Manager is not consulted about replication status.
- **SecondCopy** If the database is replicated, at least one of the passive copies must have synchronized all the changes made to date to move the mailbox . This is the default action if the value of *DataMoveReplicationConstraint* is not set.
- **SecondDatacenter** If the database is replicated to a second Active Directory site, at least one of the passive copies in the other site must have replicated the changes.
- **AllDatacenters** If the database is replicated to multiple Active Directory sites, at least one of the passive copies in each site must have replicated the changes.
- **AllCopies** If the database is replicated, all passive copies must have replicated the changes.

If Active Manager confirms that replication of the new mailbox data is healthy and meets the *DataMoveReplicationConstraint* setting for the database, MRS continues the mailbox move. This checking occurs throughout the time required to move a mailbox, and if necessary, MRS will stall for 30 seconds to allow a server that is under load to complete the necessary log replication and then check to see whether the move can continue. MRS fails the move if the stalled condition is not lifted within 15 minutes. The stalled condition is recorded in the mailbox move history, where you'll see information like this:

```
9/9/2013 4:15:57 PM [ExServer2] Move for mailbox '/o=contoso/ou=Exchange
Administrative Group (FYDIBOHF23SPDLT)/cn=Recipients/cn=Redmond, Tony' is
stalled because DataMoveReplicationConstraint is not satisfied for the database
'DB1' (agent MailboxDatabaseReplication). Failure Reason: Database 1ec5194c-
90b9-43d8-aae1-01a5374438f2 does not satisfy constraint SecondCopy. There are
no available healthy database copies. Will wait until 9/9/2013 4:30:57 PM.

9/9/2013 4:16:27 PM [ExServer2] Request is no longer stalled and will continue.
```

The *DataMoveReplicationConstraint* setting is specific to a mailbox database and can obviously vary across the databases in a DAG. In addition, MRS uses two settings in

its configuration file to determine how it interacts with Active Manager for all databases. EnableDataGuaranteeCheck controls whether MRS interacts with Active Manager to determine the current replication status for databases (default is True); DataGuaranteeCheckPeriod controls how often MRS checks with Active Manager (default is five minutes with a range from 30 seconds to two hours). EnableDataGuaranteeCheck must be True and a *DataMoveReplicationConstraint* set for a mailbox database before MRS and Active Manager will work together to ensure that mailbox moves do not compromise high-availability operations.

In the unlikely event that the target server has not generated a log roll because it is waiting for enough data to fill its log buffer, MRS forces the Store to close off the current transaction log and generate a new log. This action then forces the now-closed transaction log that contains the last data for the mailbox move to be replicated to the servers that hold copies of the target database. MRS forces a log roll for each mailbox move if a move request is waiting for more than one minute to complete. This makes sure that a move is treated as a unique transaction. Forcing a log roll should be an extremely rare event because normal user and background assistant processing on the Mailbox server should easily fill the 1 MB log buffer very soon after the last data arrives for the mailbox being moved, so this is strictly a case of being sure and safe.

It's possible that a database transition for the database holding a target mailbox will occur during a move operation. When this happens, the MRS on the server that previously hosted the active database copy holding the target mailbox drops the move request because it is no longer responsible for the target mailbox. To complete the move, the MRS running on the server that now hosts the active database copy holding the target mailbox picks up the move request and processes it, hopefully to a conclusion.

MRS configuration file

Some aspects of MRS operation are controlled by settings in the *Exchange\Bin\MSExchangeMailboxReplication.exe.config* configuration file. This is a different situation from Exchange 2010, in which the MRS configuration file provided the only guidance to MRS. As you learned in the "MRS and system resources" section earlier in this chapter, Exchange 2013 uses Workload Management to control system resources, so the settings in the MRS configuration file are less important than they were before.

You can edit the MRS configuration file by using a text editor (Figure 10-14). You do not have to restart the MRS process to pick up any change you make. Table 10-3 describes the most important settings found in the MRS configuration file.

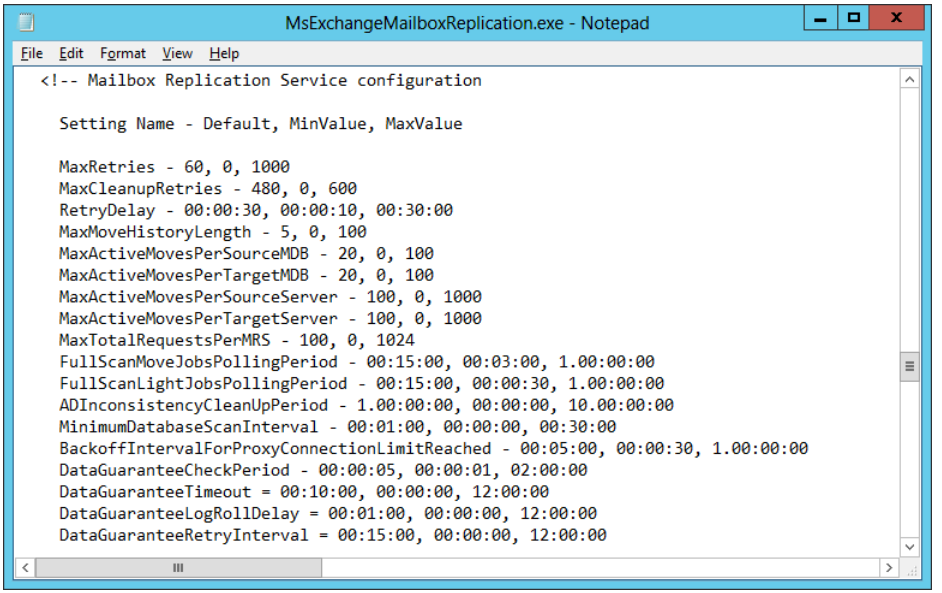


Figure 10-14 Editing the MRS configuration file with Notepad

TABLE 10-3 Properties to control MRS operation

Property	Meaning
MaxActiveMovesPerTargetMDB	Controls the number of mailboxes MRS can move concurrently to a single database. The default is 20, and the maximum is 100.
MaxActiveMovesPerSourceMDB	Controls the number of mailboxes MRS can move concurrently from a source database. The default is 20, and the maximum is 100.
MaxActiveMovesPerSourceServer	Controls the maximum number of concurrent moves a source server can perform. The default is 100, and the maximum is 1,000.
MaxActiveMovesPerTargetServer	Controls the maximum number of concurrent moves a target server will accept. The default is 100, and the maximum is 1,000.
MaxTotalMovesPerMRS	Controls the total number of concurrent moves a single MRS instance can process. The default is 100, and the maximum is 1,024.
FullScanMoveJobsPollingInterval	MRS processes move requests when they are initiated and scans databases looking for move requests to process. The default scan interval is 00:15:00 or 15 minutes.

Property	Meaning
RetryDelay	Controls how long MRS waits to retry an operation in case of transient failures. The default value is 00:00:30 or 30 seconds.
MaxRetries	Controls how many times MRS attempts an operation in case of transient failures. The default is 60. A move fails when this value is exceeded.
MaxMoveHistoryLength	Controls how many move histories MRS retains for a mailbox. The default is five, meaning that you can extract the last five mailbox history entries for a specified mailbox.

Note

If you make a change to the MRS configuration file on one Mailbox server, you should apply the same change to any other Mailbox server that is on the site so that every server functions in the same way.

Mailbox import and export

Exchange 2010 SP1 introduced a new method to import data to and export data from mailboxes based on the requests model as implemented for mailbox moves. A new PST writer library written by the Exchange development group is used, and there is no requirement to install Outlook or a MAPI library on any Exchange server to be able to import from a PST or export to a PST. Exchange 2013 completes the development for the new model by integrating mailbox import and export options into the EAC user interface to make these tasks easier.

Like mailbox move requests, mailbox import and export requests are first placed in a queue and then picked up for processing by the MRS instance running on the Mailbox server that hosts the target mailbox. The requests are processed asynchronously in the background. MRS can import data into mailboxes in any database or export data from any mailbox within its site. To facilitate mailbox import–export operations, all the PSTs MRS reads or writes must be placed in a file share that is accessible by the Mailbox servers. The file shares are secured against casual browsing by granting access only to the Exchange Trusted Subsystem. As illustrated in Figure 10-15, read/write access is required for export and read-only for import. All exports generate Unicode-format PSTs, and Exchange can read both the older ANSI and Unicode-format PSTs when it imports data.

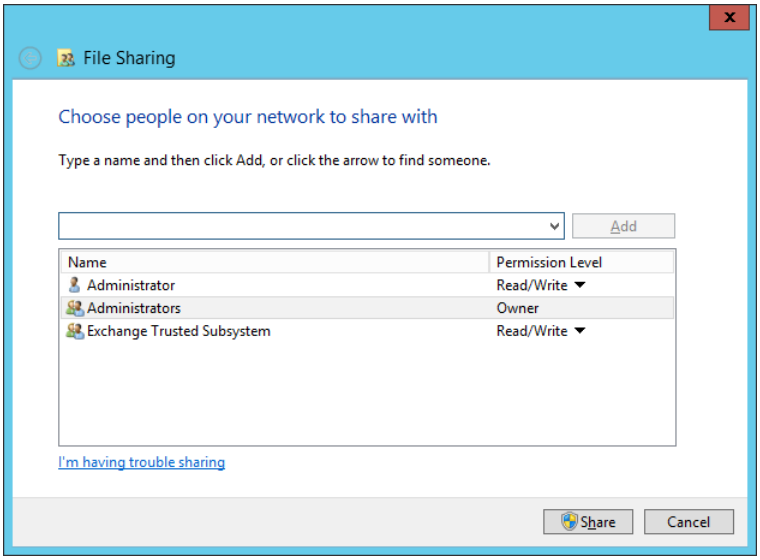


Figure 10-15 Setting the file share to allow access to the Exchange Trusted Subsystem

> **Note**
>
> **Data held in public folders or site mailboxes cannot be exported or imported using the import–export cmdlets. This information must be placed in a user mailbox before MRS can process it. For instance, you can copy items from a public folder into a user mailbox and then export the items from that mailbox to a PST.**

If you don't want administrators to use their time to manage mailbox import and export jobs, you can always revert to the time-tested basic method and ask users to move data in and out of their mailboxes by dragging and dropping items between PSTs and mailbox folders (or personal archive folders) and vice versa. However, this approach really is successful only for a small number of items because users quickly become bored with moving data from PST to mailbox and will probably never complete the task.

Gaining permission through RBAC to execute mailbox import and export

Before you can import data into or export data from a mailbox, your account must hold the Mailbox Import Export role. This role is not assigned by default even to accounts that hold the Organization Management role. A deliberate administrative action is therefore required to assign the roles to the users or groups that perform these operations. The logic

here is that accessing mailboxes is a sensitive operation that should be restricted to users who really need the permission to import data into someone else's mailbox or export data from someone else's mailbox. Although they don't hold the role by default, holders of the Organization Management role have the right to delegate the Mailbox Import Export role to users, including themselves.

> **Tip**
> **Like many other instances when you need to delegate a role before someone can perform a task, it's usually better to delegate the role to a group and then manage the group membership because users inherit or lose the role as they join or leave the group.**

You can use a command like this to assign the role to a security group. Note that the group must exist before you run the command. It is preferable to use a universal security group:

```
New-ManagementRoleAssignment –SecurityGroup 'Admins for Mailbox Import Export' –Role
'Mailbox Import Export'
```

The next time the user initializes an EAC or EMS session, role-based access control (RBAC) informs Exchange that he has the permission to use the import–export cmdlets, and the cmdlets will be loaded into the set available to the user during the session. These cmdlets are described in Table 10-4.

TABLE 10-4 Import and export mailbox cmdlets

Cmdlet	Purpose
New-MailboxImportRequest	Creates a new mailbox import request for processing by MRS
Set-MailboxImportRequest	Sets properties of a mailbox import request
Get-MailboxImportRequest	Retrieves the current status of a mailbox import request
Remove-MailboxImportRequest	Removes a mailbox import request
Get-MailboxImportRequestStatistics	Reports detailed information about a mailbox import request
Suspend-MailboxImportRequest	Suspends an import operation from a PST
Resume-MailboxImportRequest	Resumes a suspended import operation
New-MailboxExportRequest	Creates a new mailbox export request for processing by MRS
Set-MailboxExportRequest	Sets the properties of a mailbox export request

Cmdlet	Purpose
Get-MailboxExportRequest	Retrieves the current status of a mailbox export request
Get-MailboxExportRequestStatistics	Reports detailed information about a mailbox export request
Remove-MailboxExportRequest	Removes a mailbox export request
Suspend-MailboxExportRequest	Suspends an export operation to a PST
Resume-MailboxExportRequest	Resumes a suspended export operation

Planning the import of PST data

Before rushing to import data, consider how best to proceed. Among the immediately obvious challenges are the following:

- Like mailbox moves, a mailbox import request will not automatically adjust a mailbox's quota to accommodate newly imported data. For this reason, it's important for sufficient quota to exist to enable a complete PST to be imported, and you might have to adjust mailbox quotas before you start to import PSTs.

  The increase in quota can be permanent, or it can be adjusted downward again after some of or all the data are moved into the user's personal archive. Increasing the quota before an import is a relatively simple procedure to script, but it is indicative of the manual nature of the processing that surrounds PST imports. The size of a PST on disk does not equate to the amount of data that will be imported into the mailbox. The PST file structure imposes a tax or overhead of approximately 20 percent over what is required to store items in an online mailbox or archive. Even so, it's still important to ensure that sufficient quota is available in target mailboxes before you begin to import data. Roughly the same 20 percent overhead is incurred when data is exported to a PST. In other words, a 1 GB mailbox results in a 1.2 GB PST when it is exported.

- Every item that is imported creates transactions that must be captured in a transaction log. I/O demand spikes during the import as the target mailbox databases commit transactions and extend themselves to accommodate the mailboxes now swollen with PST data. Further I/O and CPU activity occurs to add items to the content indexes maintained for the target databases. More I/O is generated if you move information into personal archives afterward.

- If the import occurs inside a DAG, similar I/O spikes are experienced on servers hosting database copies due to replication, replay, and indexing activities. This has the potential to create a tsunami of I/O. The same transactional and I/O spikes in demand occur when the data are moved into the archive mailboxes.

- Mailbox data are imported from PSTs held on file shares. File shares are used because you do not know which Mailbox server currently hosts the active copy of the target database and will therefore process the import (or export) requests (unless all mailboxes are located on just one Mailbox server), so you have to provide access to the data in a location that is accessible by all the Mailbox servers. Protect the file shares against casual browsing by assigning access to only the Exchange Trusted Subsystem and place the file share on the fastest disk you can make available to ensure that operations proceed as quickly as possible.

- It's difficult to characterize how efficiently Exchange might be able to handle concurrent import operations, so scheduling the import operations to occur when they do not affect normal day-to-day user activity is often the right approach.

- The speed at which Exchange can process mailbox import and export operations varies from server to server and is highly dependent on current load and system capability. (Disk throughput is obviously important.) As an example, importing a 1.35 GB PST containing 7,450 items took a server under moderate load 15 minutes. Using a more powerful server or scheduling imports to occur at times of low user demand increases the throughput, and most servers should be able to import 8 GB per hour.

- MRS uses the MaxSendSize transport configuration setting to control the maximum size of item it can import into a mailbox. The default size is 10 MB. If you need to import larger items, you have to increase the setting. For example:

  ```
  Set-TransportConfig –MaxSendSize 20MB
  ```

- MRS ignores any passwords that users have placed on PSTs, so you do not need to ask users for these passwords. You cannot place a password on a PST when you create it during a mailbox export operation. Steps therefore have to be taken to protect the PST information held in the file share to ensure that the files cannot be opened by unauthorized clients. Remember, unlike an OST file that can be opened only when a client has knowledge of the mailbox that owns the replica folders inside the OST, any MAPI client can open a PST file. In addition, the presence of a password on a PST does not guarantee its security because many utilities are available on the Internet that can crack open a PST in a matter of seconds.

To add to the planning difficulties, before you can start to import anything, you need to understand how much data you'll actually have to deal with because this knowledge is required to understand the following:

- The resulting size of the mailbox databases after PSTs have been imported

- The quotas you have to set on primary and archive mailboxes to allow the import to occur

INSIDE OUT **Not all information about PSTs is available or reliable**

Gathering this information is a challenge because you rely on users to provide you with details such as the number of PSTs and the size of the PSTs they currently use. It's hard to scan for this information; PSTs are often inaccessible to centralized tools because they are located on disks installed in workstations or laptops that might not even be connected to the company network. For example, laptops that only sporadically connect to the company local area network (LAN) when frequent travelers are in the office might hold lots of PST information that is invisible to administrators.

Another issue to consider is what data should be imported into online mailboxes or personal archives. You might not want to import everything from user PSTs because some of it is personal and not covered by company data protection guidelines. Some data might simply be too old to worry about. Data protection guidelines often require employees to delete information after a specified period unless it needs to be retained for some reason, such as an impending lawsuit, so a survey of PSTs is a good opportunity to remind users that they should remove any unwanted information from PSTs before they import data into an online mailbox. Some companies encourage users to delete any PST older than two or three years and follow this up by implementing blocks to prevent Outlook access to PSTs after the data migration is complete.

It is obvious that the Microsoft strategy is to move away from PSTs and to get as much data into online mailboxes as possible. For this reason, and to encourage customers to begin the process of taking control over information stored in user PSTs, Microsoft introduced the PST Capture Tool in 2011 and updated it in February 2013. (See *http://technet.microsoft.com/en-us/library/hh781036(v=exchg.141).aspx* for details.) This tool enables administrators to scan for user PSTs across an organization (assuming that the PSTs are visible to the PST Capture Agent) and then import the PSTs it discovers into mailboxes that match the names of the PSTs. Similar tools exist from third-party software vendors such as Sherpa Software and Transvault, each of which has its own unique set of features and advantages over the Microsoft utility, albeit at the expense of buying a commercial product.

Although using software to discover and import information from PSTs seems like an excellent way of making this information visible to corporate compliance tools, you might end up importing an immense amount of rubbish into your brand-new online archive and having to take on the task of maintaining those data from then on, meaning that databases are larger, backups slower, and so on. When the data is in the archive, it's unlikely that users will spend much time cleaning out old and unwanted material unless they are forced to do

so. It's therefore important to have a strategy for PST import that covers important points such as:

- What data is of interest to the organization and should be held in primary or archive mailboxes.
- The tools that will be used to identify and import PST data into Exchange.
- Whether the target for the import will be the primary mailbox or archive mailboxes. If you don't already have enterprise Client Access Licenses (CALs), they are required to use archive mailboxes.
- Whether quotas need to be increased on mailboxes or archives to accommodate imported data.
- Whether additional server storage is required to hold the imported data.
- Whether the import of large amounts of PST data will affect your high-availability strategy. For instance, do you need to adjust the number of mailbox databases servers host to keep database sizes to a reasonable limit?
- When information will be imported from PSTs. You might be able to do this work over a holiday period to reduce any potential impact on server performance.
- Whether retention policies are likely to affect information imported from PSTs. For example, if your default retention policy removes all items after two years, and you import a lot of PST data that is more than two years old, the Managed Folder Assistant (MFA) will remove this information from user mailboxes soon after import.
- How to communicate to users the need for PSTs to be removed, including advice about the effective use of mailboxes and archives. For example, users who are accustomed to having PSTs available to them on their PC might not understand that information stored in archive mailboxes is not available unless an online connection to Exchange is available.
- Whether to implement any restriction in software to prevent people from creating and using PSTs in the future.

Importing and exporting mailbox data with EAC

Assuming that you have the PST to hand in a suitably protected file share, importing a PST with EAC is very straightforward. First, select the target mailbox and then click the ellipsis (...) button to reveal the additional actions you can take for mailboxes, among which is Import PST (Figure 10-16).

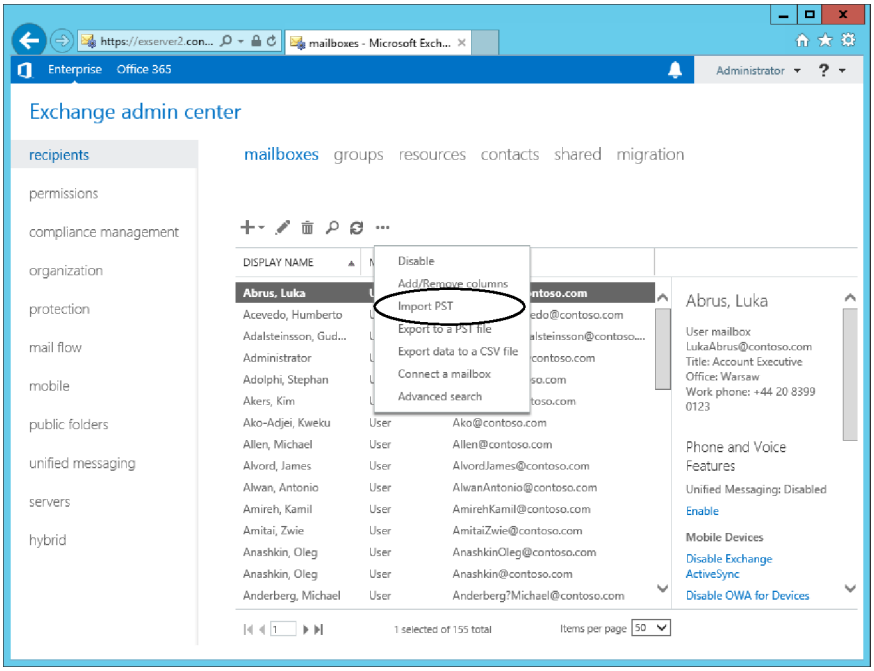


Figure 10-16 Selecting a mailbox as the target for a PST import

The next step is to identify the network file share that holds the source PST (Figure 10-17). Note that no browse button is provided to look for a PST file to import. This is because the network file share could be located on any server, and it would be impractical to browse disks on a remote server. You're then asked to confirm the target mailbox and decide whether the import should go to the primary or archive mailbox.

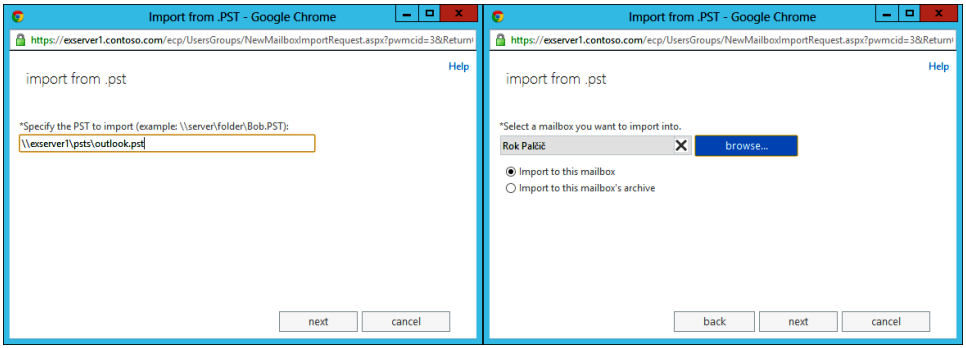


Figure 10-17 Identifying the PST to import

After identifying the source PST and the target mailbox, EAC presents another screen to ask whether you want to have a message sent to one or more recipients after the PST has been successfully imported. By default, EAC inserts the account under which it is running in the list of addresses. You can replace this address with another recipient or group. This isn't the only way to discover that a PST import has finished; EAC also signals this information as an alert to the administrator.

After confirming whether MRS should notify anyone when the import job is complete, clicking Finish submits the job for processing in the background. The exact amount of time an import job takes depends on server load and capability. As shown in Figure 10-18, the message notification MRS sends is pretty sparse in terms of detail. If you want more information about exactly what happened during a PST import, you have to use the Get-MailboxImportRequestStatistics cmdlet to create an import report (see the "Retrieving information about import jobs" section later in this chapter).

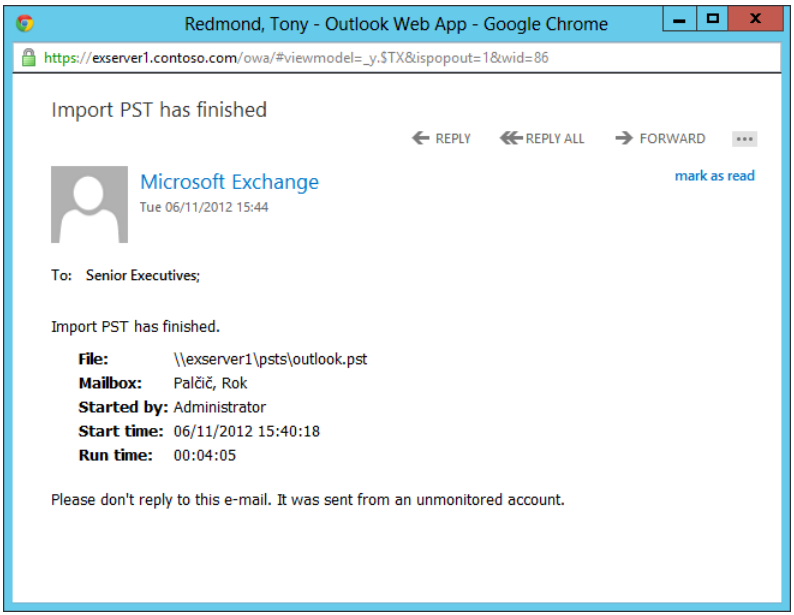


Figure 10-18 PST Import completion notification

Exporting a PST from EAC follows broadly the same sequence. You identify the mailbox to be exported, either the primary or archive mailbox. You cannot export a user's primary and archive mailboxes in one step. Instead, you first export data from the primary mailbox to a PST and then (in a separate job) export data from the archive to a different PST. Both jobs can be scheduled and processed concurrently.

You then identify the name of the target PST to which MRS will export the data. This must be in a network file share that is read/write accessible to the Exchange Trusted Subsystem. MRS will create a Unicode-format PST.

Importing and exporting mailbox data with EMS

To gain a better understanding of what happens when you create a mailbox import request, review how to manage the entire process using EMS. As you read through this description, notice that EAC does not use all the parameters that are available to mailbox import and export requests. This means that you can control mailbox import and export requests more tightly and accurately through EMS than you can through EAC. In other words, EAC is great for general-purpose import and export jobs, but when you need to exert more control or want to automate processing, you have to use EMS.

To begin, you can use the New-MailboxImportRequest cmdlet to create a new mailbox import request. For example:

```
New-MailboxImportRequest –Mailbox JSmith –FilePath
'\\Exserver1\Imports\JSmith -Outlook.pst' –Name 'JSmith PST'
–ConflictResolutionOption KeepLatestItem   –BadItemLimit 10
```

This command creates a new import request for the JSmith mailbox and identifies that the source PST is located on a file share on the ExServer1 server. Because the bad-item limit is nonzero, Exchange warns you that any items deemed corrupt will be dropped during the import. Just like mailbox moves, if you specify a value higher than 50 for the bad-item limit, MRS requires you to specify the *AcceptLargeDataLoss* parameter to indicate that you understand that a large amount of data might not be imported due to corruption.

After Exchange accepts the import request, it is queued and will be processed by the MRS running on the Mailbox server on which the mailbox to be imported or exported is located.

By default, Exchange checks for duplicate items when it imports data into a mailbox and doesn't create a copy of an item if it already exists in the target mailbox. (The message identifier detects duplicates.) In this case, the *ConfictResolutionOption* parameter specifies that if a duplicate is detected during the import, Exchange should keep the latest version of the item. The other options are KeepAll (keep all versions) and KeepSourceItem (keep the version of the item from the import PST).

You can also see that a unique name is provided for the import request. This is an optional parameter, and if you don't provide a name, Exchange uses a default job name of *MailboxImport*. If you create multiple import requests for a mailbox, Exchange uses names such as MailboxImport1, MailboxImport2, MailboxImport3, and so on to identify each import operation uniquely. The mailbox name and the request name are combined and used to retrieve information about the import request. Note that only 10 autogenerated

job names are supported per mailbox. You'll see an error if you attempt to create an eleventh import request for a mailbox without specifying a job name.

Running multiple concurrent imports

Assigning a specific name to an import request becomes important if you want to run multiple concurrent imports for the same mailbox, each of which processes data from a different PST. (You can't run concurrent imports from the same PST.) In this scenario, the default names Exchange assigns work perfectly well, but it's easier to track the progress of each job and troubleshoot any errors that occur if you assign more meaningful names such as the name of the source PST.

You don't have to import everything in a PST because the *ExcludeFolders* and *IncludeFolders* parameters enable you to control exactly which folders Exchange imports. For example, to import just a few named folders, you can pass their names as follows:

```
-IncludeFolders "Project A", "Project B", "Project C"
```

If you want to include all the folders under a specific root folder, you would pass the name of the root folder as follows:

```
-IncludeFolders "Projects/*"
```

If you have to navigate to a specific folder deep in the folder hierarchy, you can pass its name like this:

```
-IncludeFolders "Projects/2013/Secret Project A"
```

PSTs often contain items known as folder-associated items (FAIs). These are hidden items Outlook uses to store data such as rules, forms, and views. Exchange does not import associated items unless you instruct it to do so by setting the *AssociatedMessagesCopyOption* parameter to Copy. In most cases, you can avoid copying associated items from a PST because an equivalent associated item might exist in the mailbox. One exception might be when you know that these items include forms required by an application that do not exist in the mailbox.

Retrieving information about import jobs

In the next set of examples, you can see how the name assigned to the import request is combined with the mailbox identifier to retrieve information about a specific import request with which to work:

```
Get-MailboxImportRequest –Identity 'JSmith\JSmith PST'
```

The Get-MailboxImportRequest cmdlet supports a number of parameters that enable you to retrieve the status of different groups of jobs.

- The *BatchName* parameter fetches details of all requests that belong to a named batch.
- The *Database* parameter fetches details of all requests belonging to mailboxes in a named database.
- The *Status* parameter fetches details of all requests with a specified status. Valid status codes include Completed, InProgress, Queued, Suspended, and Failed.

To report the progress of an import, you can use the Get-MailboxImportRequestStatistics cmdlet to discover how much data has been transferred. Initially, you'll see that MRS creates the folder hierarchy in the target mailbox to accept the imported data, and then you'll observe an increasing count of transferred items as MRS moves data from the PST into the mailbox, for example:

```
Get-MailboxImportRequestStatistics -Identity 'JSmith\JSmith PST' | Format-List
```

Get-MailboxImportRequestStatistics reveals a lot of information, so it's a good idea to reduce the properties returned to reveal the essential data about the import operation. Here's what I normally use:

```
Get-MailboxImportRequestStatistics -Identity 'JSmith\JSmith PST' | Select Name,
Status, StatusDetail, BytesTransferred, ItemsTransferred, EstimatedTransferItemCount,
BytesTransferredPerMinute
```

When the import finishes, you can retrieve a report of everything MRS did to populate the mailbox with data from the PST by using the Get-MailboxImportRequestStatistics cmdlet. Here's how:

```
Get-MailboxImportRequestStatistics -Identity 'JSmith\JSmith PST' -IncludeReport |
Format-List
```

The report is dumped to screen, but it is more convenientand is easier to read the report by piping the output to a text file. The mailbox import report is divided into summary information at the start of the report, followed by detailed information about the processing of each folder from the source PST into the target mailbox. The following example illustrates the typical summary information about a mailbox import. The important information you will see includes the following:

- The name of the source PST
- The name of the target mailbox and the database in which it is located
- The current status of the job (completed, with no warnings)

- The number of bad items encountered during processing (three, below the five-item limit)
- The start and end times for the job and the name of the MRS that processed the job
- The total number of items and their size transferred from the PST into the mailbox
- Whether any folders were explicitly excluded or included

Your import job encountered some bad items, those that Exchange cannot process for some reason. They could be corrupt or missing some properties that Exchange requires. You can discover details about the bad items (and any large items) by running a command such as:

```
$Rpt = Get-MailboxImportRequestStatistics -Identity 'JSmith\JSmith PST'
-IncludeReport
$Rpt.Report.BadItems
$Rpt.Report.LargeItems
```

If you want to cancel an import operation, you first suspend the import request and then remove it; for example:

```
Suspend-MailboxImportRequest –Identity 'JSmith\JSmith PST'
Remove-MailboxImportRequest –Identity 'JSmith\JSmith PST'
```

Importing directly to an archive

If a mailbox has an archive, you can import PST data directly into the archive without going through the primary mailbox. This is an important functionality because the personal archive is the natural storage location for much of the information users hold in PST files.

To import PST data into a personal archive, you must specify the IsArchive parameter when you create the import request with the New-MailboxImportRequest cmdlet, for example:

```
New-MailboxImportRequest –Mailbox 'Tony Redmond' –IsArchive –FilePath
'\\ExServer1\ImportData\TR.PST' –Name 'Import-Archive'
```

Exchange will not accept the import request if the mailbox does not have an archive.

Exporting mailbox data

Many circumstances make it necessary for an administrator to export information from a mailbox. One example is when a user is transferring responsibility for a project to someone else and wants to provide all the information about the project to that person. For anything

but a few items, it would be too onerous and boring to email each item, so it's more convenient to export a complete set of items to a PST. Of course, users can export information to a PST by dragging and dropping the items, but this can be time consuming if there are hundreds of items to transfer. Further problems can then occur if the two users are not in the same geographic location. It's just easier if an administrator (or even better, a help desk representative) can do the work from a central point. Another reasonably common situation is when a company has to provide mailbox data to lawyers as the result of a subpoena.

Exporting mailbox data with EMS

A similar approach is followed to export data from Mailbox servers as described for import operations, except that you use a different set of cmdlets. Start with the New-MailboxExportRequest cmdlet to create a new export request. For example, to export a complete mailbox to a PST, you might use a command like this:

```
New-MailboxExportRequest -Mailbox 'Jeff Smith' -Name 'JSmith Export' -BadItemLimit 5
-ExcludeDumpster:$True -FilePath '\\ExServer1\PST\JSmith.PST'
```

This command writes all the content from the nominated mailbox to a PST in the file share location. The contents of the Recoverable Items folder are excluded from the operation. If the PST is not present, Exchange creates a new file; otherwise, if you pass the name of an existing PST, it writes the exported data into it.

Experience demonstrates that it is more common to apply qualifiers to restrict the information when you export to a PST from a mailbox than it is when you import a PST into a mailbox. Exchange provides a number of parameters to control which information is exported to a PST.

- The *SourceRootFolder* parameter specifies a folder in the mailbox to use as the base of the export. If this parameter is not passed, Exchange exports the complete contents of the mailbox. For example, this command exports only the items that are stored in the Project Athena folder and any of its subfolders. In this instance, the Project Athena folder is a subfolder of the Projects folder.

  ```
  New-MailboxExportRequest -SourceRootFolder 'Projects/Project Athena'
  ```

- The *TargetRootFolder* parameter specifies a root folder in the target PST to create the folders exported from the mailbox.

- The *IncludeFolders* parameter specifies one or more folders that are to be exported. For example:

  ```
  New-MailboxExportRequest -IncludeFolders 'Projects', 'Planning/Budgets'
  ```

- The *ExcludeFolders* parameter specifies one or more folders that are excluded from the export.

- The *IsArchive* parameter specifies that the export should be done from the user's archive rather than from the primary mailbox.

Just like import requests, you can trace the status of an export request and fetch information about the work it is doing. The Get-MailboxExportRequest cmdlet returns the status of a request so you can see it moving from Queued to InProgress to Completed (or, indeed, to Failed if MRS encounters a problem). For example, you'd use the following command to retrieve the status of the export request you created previously:

```
Get-MailboxExportRequest –Identity 'Jeff Smith\JSmith Export'
```

The Get-MailboxExportRequestStatistics cmdlet returns detailed information about an export request. For example, this command fetches slightly different information from the data returned from the example used for an import request:

```
Get-MailboxExportRequestStatistics –Identity 'Jeff Smith\JSmith Export' | Select
Name, Status, StatusDetail, EstimatedTransferSize, EstimatedTransferItemCount,
BytesTransferred, ItemsTransferred, PercentComplete
```

At the end of a job, you can use the IncludeReport parameter to fetch a complete narrative of the export operations.

```
Get-MailboxExportRequestStatistics –Identity 'Jeff Smith\JSmith Export'
-IncludeReport | Format-List
```

As with import requests, the amount of information Exchange provides about an export to a PST is too much to be viewed comfortably on screen, so it's best to pipe the output from this command to a text file to allow a more leisurely review of the data.

Just like mailbox move requests, it's possible that Exchange will encounter a problem with a corrupt item in a mailbox and won't be able to write it into the PST. The *BadItemLimit* parameter governs what happens next. If you don't set a value for *BadItemLimit* when you create the export request, Exchange aborts the export operation after it encounters one corrupt item. The export request is then in a Failed status. To determine what happened, use the Get-MailboxExportRequestStatistics cmdlet to generate a report and then extract details of the bad items as explained in the preceding discussion about mailbox imports. You could delete the offending items from the source mailbox or leave them in place and resume the export from the point at which it failed. To do this, you first set a higher tolerance for corrupt items and then resume the export request; for example:

```
Set-MailboxExportRequest -Identity 'Jeff Smith\JSmith Export' –BadItemLimit 10
Resume-MailboxExportRequest –Identity 'Jeff Smith\JSmith Export'
```

Alerts

Mailbox migration, import, export, and restore operations all signal alerts to EAC. The alerts notify administrators when operations start and finish and whether any problems occur. This is a new feature in Exchange 2013. Any component can create an alert by writing it into the arbitration mailbox called *FederatedEmail.4c1f4d8b-8179-4148-93bf-00a95fa1e042*. (The GUID differs from organization to organization.) Apart from MRS, the only other Exchange component that currently signals alerts to EAC is a check for expired certificates that is performed every 24 hours or any time the Exchange Service host service restarts. The idea is that alerts bring issues to the attention of administrators that they might otherwise overlook, and it is expected that other Exchange components will add their own alerts over time.

Components control how they write alert information into the arbitration mailbox. Some information is written as soon as an operation starts, as when a migration batch or PST import begins, and is updated as the operation progresses. For instance, if you start a migration batch, the Migration service marks it as InProgress if you allow the batch to start automatically or Queued if it's awaiting administrator approval to proceed. When the batch finishes, the Migration service updates its notification to Complete. Others take a more leisurely approach because their information is not quite so time critical, which is why the scan for expired certificates occurs daily.

EAC checks the arbitration mailbox for alerts it should signal every 30 seconds. When an alert is discovered, EAC signals it to the administrator. In Figure 10-19, you can see that EAC has discovered the presence of an expired certificate. If the administrator clicks the View Details link, EAC brings him to the Certificates section (under Servers) so he can deal with the problematic certificate. However, clicking the link does nothing if you run EAC using an account that does not hold an RBAC role that allows access to certificates.

You cannot create new types of alerts, but you can fetch information about alerts or create a request for Exchange to notify specific recipients by email when a task is complete. The Get-Notification command reads information from the arbitration mailbox and provides insight into the information available to EAC. You can compare the following output to the alerts displayed in Figure 10-19.

```
Get-Notification
```

```
Display Name               Type        Status       StartedBy           StartTime
------------               ----        ------       ---------           ---------
January 16 Batch (to VIP)  Migration   Queued       Redmond, Tony (IT)  16/01/2013 12:23:00
Exchange 2013 Test         CertExpiry  CertExpired  Microsoft Outlook   14/01/2013 13:46:02
```

The Set-Notification command creates a request to be notified when a task is complete. For example, this command instructs Exchange to send email to two users when migration batches are complete.

```
Set-Notification -NotificationEmails "TRedmond", "JSmith" -ProcessType Migration
```

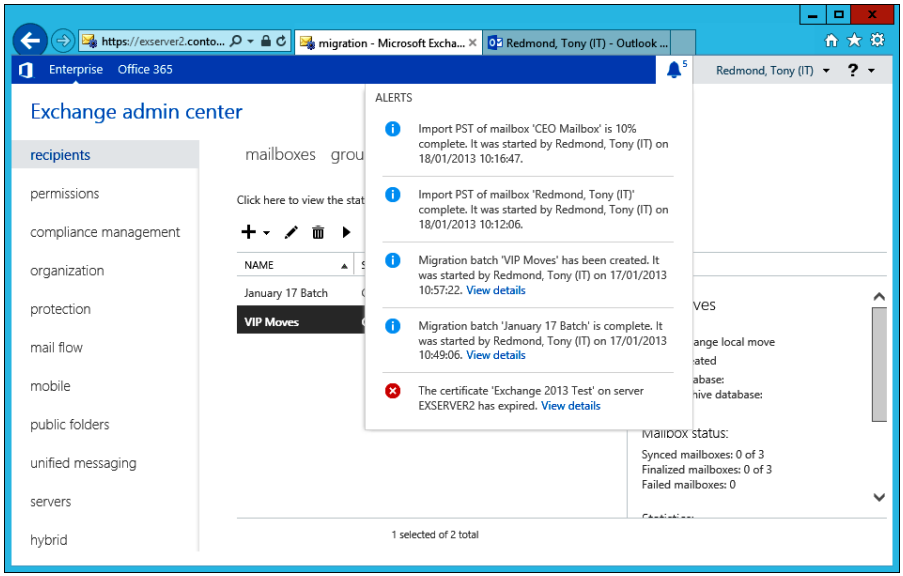


Figure 10-19 Alerts that EAC displays

Note that the Migration service enables you to enter a notification address for migration batches through EAC or by running the New-MigrationBatch or Set-MigrationBatch commands. This address receives the migration report rather than a simple notification that a job has finished.

Notifications remain in the arbitration mailbox until they are removed or expire. It is the responsibility of the component that creates a notification to remove it, and this is usually done when the associated task is complete. For instance, if you remove a migration batch, the notification for that batch is deleted from the arbitration mailbox. If notification items are left in the arbitration mailbox, they will be removed by the MFA after 30 days because they are stamped with the AsyncOperationNotification system retention tag. The next chapter discusses more about how the MFA processes retention policy tags.

Time to comply

Some of the material covered in this chapter has touched on compliance. Given the modern world's reliance on electronic communications and email in particular, the topic of compliance deserves far more in-depth coverage, especially when Microsoft has dedicated so much effort in Exchange 2010 and Exchange 2013 to make the product flexible and powerful when meeting the regulatory and legal compliance requirements that so many companies have to cope with today. Let's move to comply!

CHAPTER 11

Compliance management

The joy of legal discovery 642
Archive mailboxes 645
Messaging records management 657
How the Managed Folder Assistant implements retention policies 689
Preserving information 696
The value of the Recoverable Items structure 737
Auditing administrator actions 748
Auditing mailbox access 754
Other compliance features 763

The need to comply with legal and regulatory requirements is a fact of corporate life today. Legislation such as the Sarbanes–Oxley Act in the United States has influenced many other countries to introduce similar requirements to keep records that show when something was done and by whom. Microsoft started on the process to build messaging records management (MRM) capability into Exchange Server 2007. However, users (and many administrators) didn't understand the purpose of MRM, and take-up was weak. To address these issues and to provide a true basis for compliance, Exchange 2010 introduced a wide range of new features, including a new implementation of MRM and the first appearance of archive mailboxes. Exchange 2013 builds on the foundation laid by Exchange 2010 to refine and improve the features, including the following:

- The ability to audit administrator and mailbox actions.
- The provision of archive mailboxes as a replacement for the much-loathed (at least by me) PSTs.
- The ability to create and apply retention policies to items and folders.
- The ability to recover items without requiring a backup to be restored.
- The ability to conduct searches against mailbox content, including deleted items information held in the Recoverable Items folder. If Exchange 2013 is integrated with Microsoft SharePoint 2013, searches can also be conducted against information held on SharePoint sites through the eDiscovery Center. If Exchange 2013 operates on a standalone basis, eDiscovery searches are conducted through the Exchange Administration Center (EAC).
- The ability to place mailboxes on hold so that items are retained even when the user might prefer them not to be.

The features just listed are not exhaustive because other Exchange features can be associated with compliance. For example, transport rules allow a disclaimer to be appended to every outgoing message that can limit liability by complying with rules that say messages from a company must contain specific contact or other information about the company. Data loss prevention (DLP) is a new feature of Exchange 2013, which uses a special form of transport rules and some special processing to detect patterns in email content that might represent different forms of confidential information, to stop users from sending this information through email when they should not. The transport system also supports journal rules and enables servers to capture copies of messages in a legally robust manner so that those copies can be cited in court cases. (Capturing information through an immutable hold is another way to achieve the same goal.) Transport, journal, and DLP rules are covered in *Exchange Server 2013 Inside Out: Connectivity, Clients, and UM*, by Paul Robichaux (Microsoft Press, 2013).

When you examine the range of compliance features incorporated in Exchange, you understand that compliance is an area that continues to evolve, largely because changing legal and regulatory needs drive such evolution. Microsoft will update the Exchange feature set over time to satisfy the broadest possible set of requirements. However, it is unreasonable to expect any software to deliver a complete answer. For example, although you can place mailboxes on hold or establish an extended deleted item retention period to ensure that important information is not deleted, you might also need to develop administrative procedures to handle situations such as mailbox retention following the death or dismissal of an employee. Handling situations such as this could be reasonably straightforward—disable the account, hide the user's mailbox from the Global Address List (GAL), and keep it and any archives, including PSTs that you recover from the user's PC, until any legal hold period has passed—as long as you think everything through. With that thought in mind, let's review the compliance features in Exchange 2013.

The joy of legal discovery

Legal discovery actions have been around for centuries. Over the past two decades, the focus of discovery or searches for information pertinent to a legal case has begun to shift from paper evidence to electronic evidence. This shift reflects the different manner in which organizations store data today. Filing cabinets are still stuffed with paper, but much of the correspondence companies conducted by letter, fax, and telex are now sent by email, so the focus of discovery has to accommodate both paper and electronic media.

Discovery actions for email systems first began in the mid-1980s. Messages were recovered from backup tapes and printed for lawyers to review. The process was expensive and time consuming. The only mitigating factor was that it was much easier to determine who might have sent an incriminating message because relatively few people in a company had email, and the overall volume of email was low. Messages were text only and tended to be short.

It was therefore possible to satisfy a judge's order to retrieve all messages for 10 specific users over a month without running up an extraordinarily high bill.

Today's environment is different. Many more users are typically hosted on each server, they send and receive an ever-increasing volume of messages, and those messages contain many types of attachments, including video and audio files. The result of living in the age of electronic communication is that the cost of legal discovery is higher because there is more information to process. In March 2009, *Fortune* magazine reported that the court-appointed trustee of bankrupt Lehman Brothers Inc. had captured 3.2 billion email and instant messages, occupying 1.4 terabytes (TB). This isn't an unusual amount; the FBI investigation of Enron in 2001 reviewed 31 TB of data and used 4 TB as evidence. Email is a critical means of business communication that has replaced telexes, faxes, and written letters in many respects, so legal discovery of email has moved from an out-of-the-ordinary situation to a form that is extremely common, whether it is to satisfy a legal or regulatory requirement, respond to a subpoena, or deal with an internal matter concerning employee ethics, harassment, or discipline.

The first generation of Exchange offered no way to store mail after it was deleted, so you had to restore a database from a backup if you wanted to recover a message, whether it was needed to satisfy a legal order or because a user had deleted it in error. Gradually, Microsoft began to add new features to Exchange to help. The original version of the dumpster (the official term now used is the "Recoverable Items" structure), as implemented in Exchange 2000 through Exchange 2007, provides a two-phase delete process by which messages are marked as deleted but kept in the database until their retention period expires, at which time they are removed. As discussed in the "The function of the Recoverable Items structure" section later in this chapter, Exchange 2013 uses an enhanced set of folders to underpin a number of compliance features such as in-place hold.

Journaling appeared in Exchange 2003 and was upgraded in Exchange 2007. However, the journaling functionality Exchange offered was basic, and most companies that invested in products to capture copies of messages preferred purpose-designed products such as Symantec's Enterprise Vault or Iron Mountain's NearPoint. As mentioned earlier, Microsoft added managed folders in Exchange 2007 with the idea that administrators could create folders that are distributed to mailboxes for users to store important items. However, the reality is that most organizations ignored managed folders.

The compliance features in Exchange 2007 were a start. However, the overall experience was not compelling enough to generate widespread usage, which then led Microsoft to create a new set of features that have been rolled out over Exchange 2010 and Exchange 2013.

Although Exchange 2013 includes a wide range of compliance features, Microsoft must convince customers that having integrated archiving and search incorporated in an email

server is a better solution than dedicated archiving and search applications that have been in use and developed over many years. It can be argued that cost is one key Microsoft advantage because archiving is available at the price of an enterprise Client Access License (CAL) that might be already acquired. Another obvious advantage is the integration of the compliance features into the core of Exchange, meaning that customers do not have to pay for and manage an additional system to gain compliance features.

The cost of an enterprise CAL for each user is often lower than the cost of dedicated archiving software plus any additional hardware that is required to run the archiving software. This argument works only if the functionality available in Exchange meets your requirements. Microsoft acknowledges that many vendors have been actively selling compliance solutions for Exchange for nearly a decade. Some offer different functionality than Exchange, especially in areas such as workflow, the ability to archive information taken from other sources, and the experience that companies have with these products in integrating compliance processes with various regulations.

If SharePoint and Exchange are the most important repositories of information within your company, the two will serve as an excellent platform to enable compliance, provided that you can deploy the necessary software versions to use features such as site mailboxes and conduct searches across both repositories. (See *http://technet.microsoft.com/en-us/library/jj218665.aspx* for information about how to configure Exchange for use with the SharePoint eDiscovery Center.) If Exchange 2013 is used without SharePoint, then the focus needs to be on how to extract maximum advantage from its compliance features. Based on this premise, you then focus on:

- Deploying archive mailboxes in an attempt, perhaps in vain, to eliminate the sprawl of PSTs used across the company. The aim is to use large mailboxes to enable users to keep all their data online, which is an advantage for both users and the company. After it is online, the data is exposed to indexing and search.
- Deploying suitable retention policies to help users keep control of their (now larger) mailboxes. Retention policies can sweep unwanted items out of user mailboxes on a regular and automatic basis while moving items that need to be retained into archive mailboxes.
- Working with the company legal department to determine appropriate policies to govern:
    - When users are placed on hold (when they are prevented from deleting items from their mailboxes or making any other alteration to mailbox content). Exchange captures attempts to delete or edit information in the backup without interfering with the user's ability to work with her mailbox.

- When and how eDiscovery searches are performed, who can authorize these operations, who has access to the data recovered by searching, how long this data is retained, and how and when it is removed from servers.
- When administrator and mailbox auditing is used and who has access to reports generated from this data.

Having some focused goals for compliance is a good way to begin complying. With that point in mind, the following discusses some of the ways you can comply.

Archive mailboxes

An archive mailbox, or personal archive, is a logical extension of a user's primary mailbox that provides an online archive facility. The name might cause some confusion with the personal archives users create with Microsoft Outlook. The big difference is that the Exchange archive is tightly integrated in the Information Store, and the data held in the archive are therefore accessible using all the features available to mailboxes, including eDiscovery searches. By comparison, PST archives are usually confined to an individual PC, and the data that they contain are inaccessible to server-based processing. (It is possible, but not recommended and not supported, to place PSTs on network file shares; see *http://support.microsoft.com/kb/297019/en-us*.)

An archive mailbox can be stored in the same database as the primary mailbox, or it can be in a different database. Some deployments have created special archive servers that host databases containing only archive mailboxes. This is a perfectly acceptable solution that offers some advantages because the hardware can be tailored to the lower demands that exist for access to archive information. Usually, people don't access their archive mailboxes as frequently as they do a primary mailbox, which is constantly busy with the process of receiving and sending messages. In essence, therefore, an archive is infrequently used but always available online.

INSIDE OUT **Outlook cached Exchange mode and archive mailboxes**

Outlook maintains local copies of all the folders in the primary mailbox through synchronization when it operates in cached Exchange mode to enable the user to work offline or to work unhindered by any temporary network outage. Outlook never caches local copies of folders from archive mailboxes. An archive is designed to hold information that is important and has to be retained but is not accessed frequently. Therefore, you can access information in an archive mailbox only when you are connected online to Exchange.

If you use Microsoft Office 365, archive mailboxes can be stored in the cloud, an option that has proven increasingly attractive as companies gain more experience and confidence with cloud-based services. It is attractive to hive off archives to a cloud-based service because this enables you to remain focused on the care and maintenance of production mailboxes while the hosting provider takes care of the archives. Whatever option you choose, a mailbox can have just one personal archive, and each mailbox that has an on-premises archive requires an enterprise CAL. Mailboxes that use cloud-based archive mailboxes in the Microsoft Exchange Online Archive service do not need enterprise CALs.

Microsoft views archive mailboxes as the natural replacement for PSTs. The growth of messages and the reluctance of administrators to increase mailbox quotas coupled with the inability of Exchange and its clients to deal elegantly with very large mailboxes (5 GB and up) meant that most organizations were forced to use PSTs to offload data from the online store. Users do like to keep messages, even if they never look at them again. (Some conference speakers have opined that a message filed in a PST has a 99 percent chance of never being looked at again after six months; my personal experience tallies with this estimate.) Other problems with PST management typically cited in corporate messaging deployments include the following:

- **Reduced security** PSTs are personal stores, but users keep just about anything in them, including sensitive and usually unencrypted corporate information ranging from budgets to presentations about new products to performance reviews. If someone loses a laptop—or even a USB device that has a PST on it—that information is immediately exposed and potentially available to anyone who finds the device and accesses it. Even if protected by a password, the PST file structure is insecure and can be quickly accessed by using utilities commonly available on the Internet. After the password is bypassed, a PST can be opened using any Microsoft Outlook client.

- **Inability to respond to discovery actions** Information held on a PST is usually invisible to searches that a company performs to respond to discovery requests. This is fine if the information is personal or irrelevant to the discovery request, but it could be very expensive if required information is not disclosed to a court and is subsequently discovered.

- **Inability to apply policy** Many companies have a data retention policy that requires users to delete documents and messages after a certain period. The period can vary, depending on the type of information contained in different items. In any case, the company loses any ability to apply policy centrally after a user moves an item from his mailbox into a PST.

- **Exposure to data loss** Laptop disks are notoriously prone to failure. If users don't back up their data, any disk crash exposes them to potential data loss, and that information might be important.

The alternative solution to increasing disk quota for mailboxes in previous versions of Exchange was to buy and deploy a dedicated third-party archiving solution. Using PSTs is obviously far cheaper for a company. It's also easier for users because they control how many PSTs they create and how they use them. Some create a separate PST for each year; some create a PST for each major project. The big downside is that PSTs then expose the company to the risks previously described. Even so, it will take time to pry user fingers from their beloved PSTs.

Exchange archive mailboxes are not perfect, and a number of limitations exist that could hinder deployment, including the following:

- You cannot transfer an archive to another mailbox. If a user leaves and you delete her mailbox, the archive is also removed. You can save data by exporting items from the archive (and the primary mailbox) to a PST and then importing it back into the personal archive of another user, but it would be more elegant just to transfer the archive intact.
- You cannot copy or move sections of the archive to transfer it to another user. For example, a user who wants to transfer responsibility for a project to another user has to extract the folders and other items relating to the project from her archive and provide them to the other user. Again, the workaround is to export selected folders from the personal archive to a PST and provide the PST to the other user (or import the PST into her archive). Alternatively, a site mailbox or public folder might serve as a better repository for information that has to be shared between different project members.
- You cannot assign permissions on a folder level within the archive to allow users to give access to parts of their archive to other users. Delegates who have full access to a user's mailbox can access the complete archive for that mailbox.
- Archive mailboxes are inaccessible from mobile clients and from Outlook for Mac. Given the use of mobile devices today, this can be an issue for some users.

These are examples of functionality Microsoft will doubtless consider enhancing in the future. It's likely Microsoft will wait to see how archives are used in practical terms within customer deployments before it plans how archives evolve in future releases of Exchange.

Enabling archives

Before you can create and use archive mailboxes with Exchange, you have to deploy clients that support the feature. These are as follows:

- Outlook 2007 Service Pack 3 (running the November 2012 cumulative update or later; see *http://blogs.technet.com/b/exchange/archive/2010/12/20/3411710.aspx*)

- Outlook 2010 Service Pack 1 (running the November 2012 update or later)
- Outlook 2013
- Outlook Web App

Outlook Professional Plus and Outlook standalone support archive mailboxes (see *http://office.microsoft.com/en-us/outlook-help/license-requirements-for-personal-archive-and-retention-policies-HA102576659.aspx)*. The other editions (such as Office Home and Business 2013) do not include the code necessary to open and reveal archive mailboxes. No mobile clients currently support access to archive mailboxes.

You can enable an archive when you create a new mailbox by clicking the More Options link at the bottom of the screen used to enter new mailbox details. This reveals the check box by which to indicate that an archive should be created alongside the primary mailbox (Figure 11-1). You can also select a specific mailbox database to hold the archive or just click Save to have Exchange use its auto-provisioning feature to select a database to hold the archive.

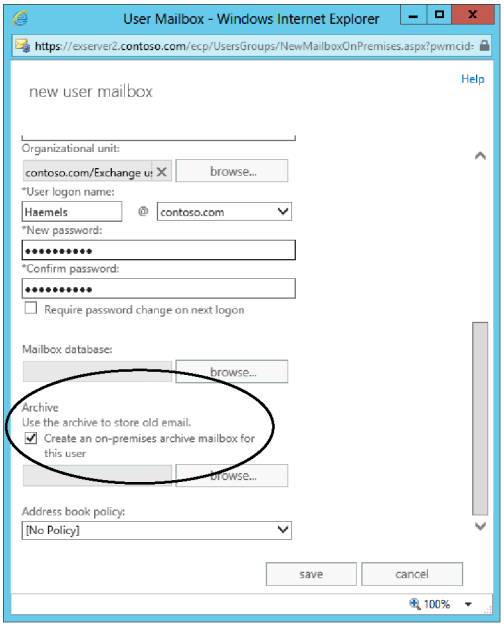


Figure 11-1 Opting to create an archive for a new mailbox

To enable an archive when you create a mailbox with Exchange Management Shell (EMS), you just add the –Archive parameter to the New-Mailbox cmdlet and the –ArchiveDatabase

parameter to select a database for the archive mailbox. (Again, this is not necessary because Exchange will pick a database for you if you omit the –ArchiveDatabase parameter.)

See Chapter 5, "Mailbox management," for a full discussion about how to create new mailboxes.

INSIDE OUT **You won't lose access to your archive if there is a server failure**

EAC restricts the databases you can choose to hold an archive for a mailbox to those that are mounted on servers in the same site. However, Exchange supports an exception to this rule when a database transfers to a server in another site following a failure. In this case, the CAS redirects clients to the archive in the database on the other site by using a cross-site connection for as long as the database is hosted on that site. You won't want to use cross-site connections for an extended period, and normal connections will resume after you switch the database that contains the personal archive back to a server on its original site. If you have a tenant subscription for Office 365 and have configured the necessary hybrid environment to support data moving between on-premises and cloud servers, you will also see the option displayed by EAC to allow Office 365 to host an archive.

You can also enable an archive for existing mailboxes by selecting a mailbox in EAC and then selecting Enable under the In-Place Archive section in the action pane (Figure 11-2). EAC displays a dialog box to enable you to select a database for the archive and to warn you that enabling this feature requires an enterprise CAL. If you click OK, the mailbox is enabled with an archive. You can also enable a personal archive for an existing mailbox with EMS. For example:

```
Enable-Mailbox –Identity 'Tony Redmond' –Archive
```

As soon as an archive has been enabled for a mailbox, it becomes available to Outlook Web App and Outlook the next time the client refreshes its resource information through the Autodiscover process, which provides Outlook with information about the new archive. This usually takes a few minutes for on-premises mailboxes but might need up to an hour for an archive hosted in Office 365. Outlook Web App retrieves information about the archive when it connects to the mailbox online. At this point, the new archive will hold only a Deleted Items folder.

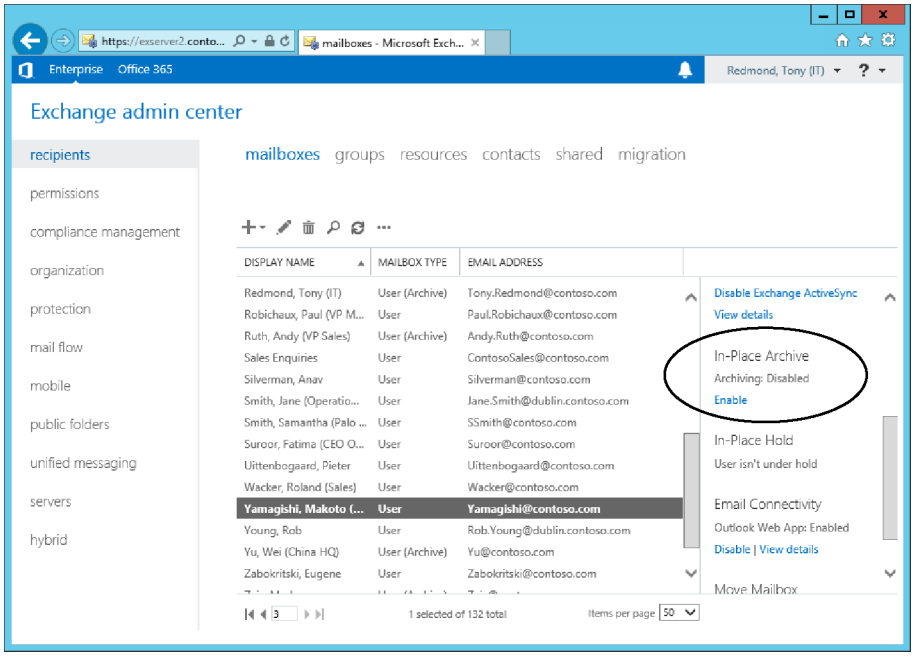


Figure 11-2 Enabling an archive for an existing mailbox

After you enable an archive for a mailbox, you'll notice that EAC displays User (Archive) instead of just User in the mailbox type column. Unlike its Exchange Management Console (EMC) predecessor, EAC does not include a prewritten query to display all the mailboxes quickly that currently have an archive. Because EAC regards archive-enabled mailboxes as having a different type, you can click the mailbox type heading to have EAC sort the mailboxes and present them together. Alternatively, you can use the Get-Recipient or Get-Mailbox cmdlets to search for mailboxes that have an archive. For example, this command looks for mailboxes that have archives enabled and reports the mailbox name, the archive name, and the databases in which the mailbox and archive are located:

```
Get-Mailbox -Archive | Format-Table Name, ArchiveName, Database, ArchiveDatabase
```

Mailboxes that use MRM 1.0 cannot have an archive

Although Microsoft does not recommend it, you can enable an archive for room, equipment, and shared mailboxes, which seems a little strange. However, you cannot enable an archive for a mailbox that has been assigned a managed folders policy. Managed folders provide the basis for messaging records management in Exchange 2007, but they are superseded by retention policies from Exchange 2010 onward and therefore do not support archive mailboxes.

Managing archive properties

Behind the scenes, EAC calls the Enable-Mailbox cmdlet to enable an archive. These commands first enable the personal archive for a mailbox and then retrieve the properties that Exchange maintains for an archive.

```
Enable-Mailbox -Identity 'Andy.Ruth@contoso.com' -Archive
Get-Mailbox -Identity 'Andy.Ruth@contoso.com' | Select Name, Arch*
```

```
ArchiveDatabase         : DB2
ArchiveGuid             : 03c8b429-5160-4418-868c-2816b8a31d71
ArchiveName             : {Personal Archive - Ruth, Andy (VP Sales)}
ArchiveQuota            : 50 GB (53,687,091,200 bytes)
ArchiveWarningQuota     : 45 GB (48,318,382,080 bytes)
ArchiveDomain           :
ArchiveStatus           : None
ArchiveState            : Local
ArchiveRelease          :
```

The first four archive properties are always populated for a mailbox when it is archive-enabled. The globally unique identifier (GUID) identifies the archive mailbox within the database where it is stored. The default name for the archive is derived from the Personal Archive prefix plus the mailbox's display name and can be changed afterward to whatever name you prefer. The archive quotas are inherited from the default values set for the database and reflect the values Exchange uses to limit the amount of information in the archive and when it starts to issue warning messages.

You can alter these values with the Set-Mailbox cmdlet. For example:

```
Set-Mailbox -Identity 'Andy.Ruth@contoso.com' -ArchiveName "Andy's Splendid Online
Archive" -ArchiveQuota 2GB -ArchiveWarningQuota 1.9GB -UseDatabaseQuotaDefaults
$False
```

The last four properties have the following meaning:

- *ArchiveDomain* is used only if the personal archive is stored on an Exchange Online server (Office 365). If used, the property holds the name of the tenant domain.
- *ArchiveStatus* contains a status value to indicate whether the personal archive has been created on an Exchange Online server.
- *ArchiveState* is Local, in this case meaning that the archive is on a local, on-premises server.
- *ArchiveRelease* is reserved for Microsoft purposes and might be used to indicate that an archive depends on a particular version of Exchange in the future. For now, it remains blank.

Checking space usage

The amount of space used in an archive mailbox can be checked with the Get-MailboxStatistics cmdlet, which supports the –Archive parameter to tell it to report details of the archive mailbox rather than the primary mailbox. For example:

```
Get-MailboxStatistics -Identity 'John Smith' -Archive | Format-Table DisplayName,
ItemCount, TotalItemSize, LastLogonTime -AutoSize
```

Updating properties of an archive mailbox

You can update some archive properties, including its name, through EAC. To do this, select the mailbox, click Edit, navigate to Mailbox Features, and then select View Details for the Archiving section. You can update the name (Figure 11-3) and the quotas assigned to the archive. EAC also displays details of the quota currently used in the archive.

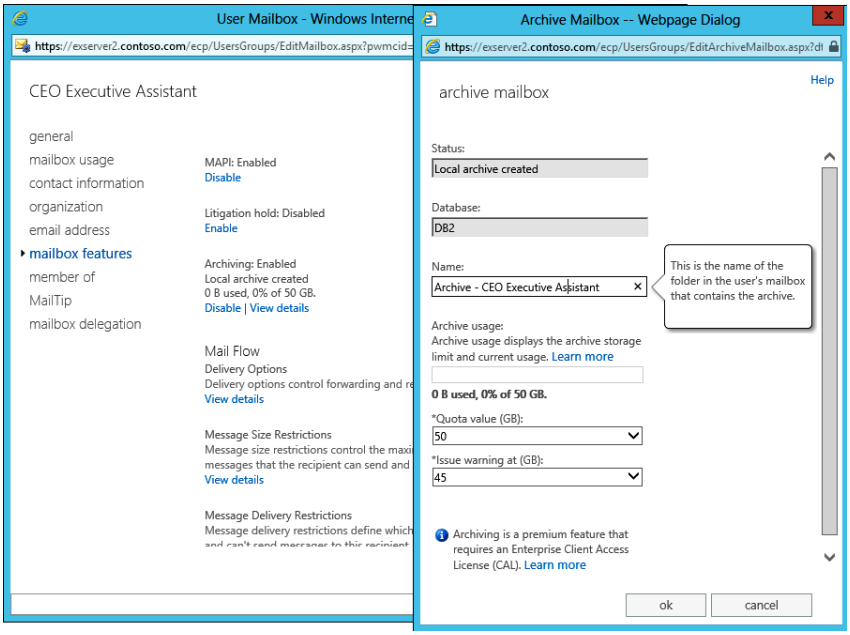


Figure 11-3 Updating the name of a personal archive through EAC

The default archive and retention policy

When you enable a personal archive for a mailbox, Exchange assigns a retention policy called Default MRM Policy to the mailbox unless the mailbox has a retention policy already assigned. This policy is designed to help the mailbox's owner use the archive by automatically moving items from the primary mailbox into the archive as their retention period

expires. Assigning a retention policy automatically to a mailbox might be considered helpful, but the action has some consequences for users, which are discussed in the following paragraphs.

The retention period applied by the default tag in the policy is two years, so the effect of applying the policy is that any item in the mailbox that is not stamped with another tag will be moved into the archive after it is two years old. The retention policy assigned to the mailbox becomes effective the next time the Managed Folder Assistant (MFA) processes the mailbox. The default policy is not assigned if the mailbox is already under the control of another retention policy. How to manipulate retention policies and tags is discussed in the "Messaging records management" section later in this chapter.

The name of the default retention policy Exchange supplies has changed over the versions; if you previously deployed Exchange 2010, you will find that another policy, the Default Archive and Retention Policy, has been applied to mailboxes that had archives enabled with Exchange 2010. In fact, an even earlier policy, Default Archive Policy, might also be present. This policy was provided with the original release to manufacturing (RTM) of Exchange 2010. Even if the other two policies are available, Exchange will apply the Exchange 2013 version to mailboxes that are newly enabled with archives.

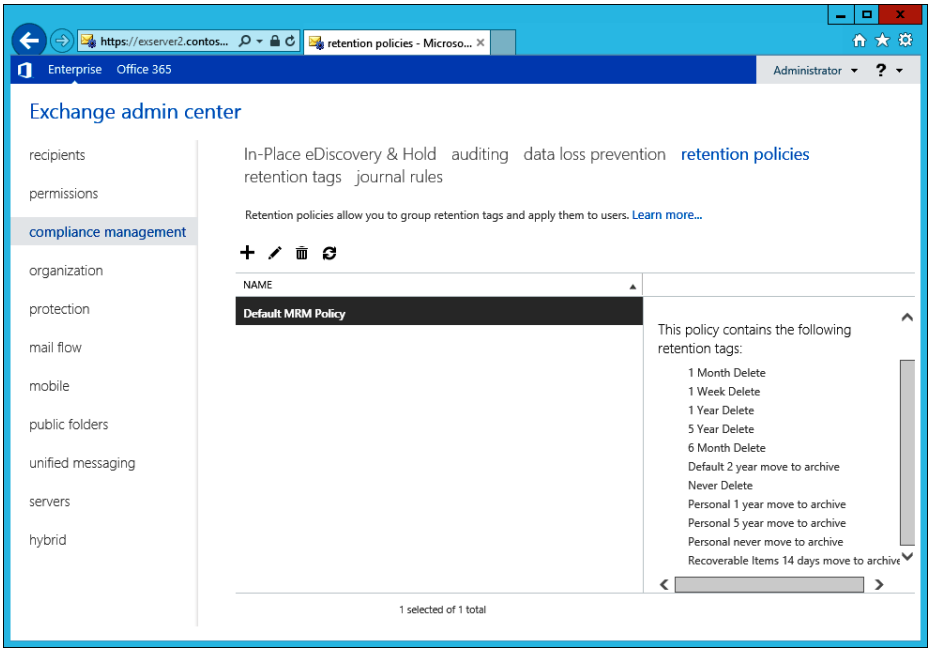


Figure 11-4 The default archive and retention policy Exchange 2013 provides

It might be strange to have multiple default retention policies within an organization. In fact, apart from their name, the policies differ slightly in terms of the retention tags they contain. The Default Archive Policy contains only archive tags and therefore does not function well in terms of an overall retention policy, which ideally should contain tags to help remove unwanted items from a mailbox and archive items that need to be retained. However, although a good reason exists for the change made from Exchange 2010 RTM to Exchange 2010 SP1, the two default retention policies in Exchange 2010 SP1 and Exchange 2013 appear functionally identical, apart from the name. The only reason Microsoft changed the name seems to be to align the default retention policies across Exchange Online and Exchange on-premises because both now use Default MRM Policy.

In any case, the Exchange 2010 SP1 version of the default retention policy will continue to work with Exchange 2013, and there is no reason to replace it with the Exchange 2013 version for the mailboxes that use the older policy. Table 11-1 describes the retention and archive tags that are included in the default policy. You can add retention archive tags to or delete them from the policy if required. You can also see these tags listed in Figure 11-4.

TABLE 11-1 Tags included in default archive and retention policy

Tag name	Type	Purpose
Default 2-year move to archive	Default	Automatically move items to the personal archive when they are two years old. This tag is applied to any item in the mailbox that does not have an explicit tag applied by the user or is inherited when an item moves into a folder that has a default policy.
Personal 1-year move to archive	Personal	Tag that the user can apply to items to instruct the MFA to move the items into the personal archive after they are one year (365 days) old.
Personal 5-year move to archive	Personal	Tag that the user can apply to items to instruct the MFA to move items into the personal archive after they are five years (1,825 days) old.
Personal never move to archive	Personal	Tag that the user can apply to items to block the MFA from ever moving the items into the personal archive.
Recoverable Items 14-days move to archive	Recoverable Items folder	Move items placed in the Recoverable Items folder to the personal archive after 14 days.
1 Month Delete	Personal	Move items into the Deleted Items folder after one month.
1 Week Delete	Personal	Move items into the Deleted Items folder after one week.
6 Month Delete	Personal	Move items into the Deleted Items folder after six months.

1 Year Delete	Personal	Move items into the Deleted Items folder after one year.
5 Year Delete	Personal	Move items into the Deleted Items folder after five years.
Never Delete	Personal	Disabled tag that prevents the MFA from processing the item; the effect is to stop the item from ever being deleted.

Notice that the default MRM policy contains only a single default tag to move items into the archive after two years; no default tag is present to delete items. Therefore, the impact of applying this policy to mailboxes is that every item more than two years old will be moved by MFA into the archive the first time MFA processes the mailboxes after the policy is applied. And because Exchange automatically applies the default MRM policy to mailboxes after they are enabled for archives, this sequence of events becomes immediately apparent if items exist in the mailbox that are more than two years old. In effect, this results in the disappearing-items syndrome in which users log problem reports for missing mailbox items. In the majority of cases, the missing items are found in folders in the archive mailbox. It just takes time for users to realize that Exchange moves items automatically after they reach a certain age; this underlines the importance of communication with the user community as you implement archive mailboxes.

INSIDE OUT **Don't delete the default policy!**

You should not delete the default archive and retention policy because this will affect the processing MFA performs for the mailboxes to which the policy is already assigned. It's a better idea to create a custom archive and retention policy tailored to the needs of the company or to different groups of users and apply that policy to their mailboxes. In this case, the custom retention policy replaces the default archive policy. How to manipulate retention policies and tags is discussed in the "Messaging records management" section later in this chapter.

Using an archive mailbox

Assuming that your mailbox is archive-enabled and a suitable client is at hand, working with items in the archive is just like working with items in the primary mailbox. You can create new items, reply to messages, move items around, and so on. After the archive mailbox is created, it is up to the user to populate it, most likely by using drag and drop to move folders or items from his primary mailbox. Administrators can import the complete

contents of PSTs into a mailbox, but there are some limitations with this approach, as discussed previously.

Exchange doesn't support offline access for data held in archive mailboxes. When Outlook is configured to use cached Exchange mode, it has access only to the offline copies of the folders from the primary mailbox that are stored in the OST and uses background synchronization to keep those folders updated. This arrangement enables Outlook to work through transient network interruptions. Outlook has to be able to connect to the server before it can work with the data stored in an archive.

TROUBLESHOOTING

I can't access my archive mailbox when I'm offline.

If you want something to be available offline (or available to a mobile device), you have to store it in the primary mailbox. The archive is designed to hold information that isn't always required immediately and that you can wait to access until you can get back online, so if you need something from the archive and know that you have to work offline (for example, on a road trip), you have to plan ahead and move the desired items from the personal archive into the primary mailbox beforehand.

Disabling an archive mailbox

If present, you can disable the archive for the selected mailbox in EAC by clicking the Disable link under In-Place Archive shown in the action pane. Alternatively, you can edit the mailbox's properties, open the Mailbox Features section, and disable the archive there. If you have an active EMS session, you can run the Disable-Mailbox cmdlet. For example:

```
Disable-Mailbox –Identity 'Smith, John' –Archive
```

EMS prompts for a confirmation before it proceeds unless you add the –Confirm:$False parameter. This is not a good idea unless you are absolutely sure that you want to disable the archive. If the archive's owner is logged on and connected to her mailbox when the archive is disabled, she can continue to access data in the archive until the next time the client disconnects.

When it disables an archive mailbox, the Information Store disconnects it from the primary mailbox and keeps it in the database until the deleted mailbox retention period expires. If you make a mistake and have to re-enable the archive before it is removed from the database, the Store reconnects the original archive, and all the information in it becomes available to the user again. Note that you cannot disable a mailbox when it is on hold; this action can interfere with the ability to preserve information that might be required

for discovery purposes. If necessary, you can specify the IgnoreLegalHold switch with the Disable-Mailbox cmdlet to force Exchange to disable a mailbox. It would be wise to seek approval from your legal department before taking such an action.

Messaging records management

Exchange 2007 introduced the messaging records management (MRM) system as its business email governance strategy to help users comply with regulatory and legal requirements. The idea is to provide a method for users to retain messages and attachments that are required business records. Another way of thinking about MRM is that it helps users keep control over mailboxes by automating the retention process; marked items are kept as long as required, whereas others can be automatically discarded when their retention period (otherwise known as the expiration limit) expires. The key to success for any scheme that aims to alter user behavior is to make it as simple as possible while achieving maximum functionality. Exchange 2007 didn't quite meet this goal, largely because users did not like being forced to change how they worked. The lack of automated processing spelled disaster for this implementation.

Microsoft therefore needed to change its tactics to provide a workable implementation of MRM. The change occurred in Exchange 2010, and the same approach to messaging management is further developed in Exchange 2013. Managed folders are deprecated in Exchange 2013. At this point, any remaining vestige of managed folders should be eliminated from Exchange deployments as quickly as practicable. See the "Upgrading from managed folders" section later in this chapter.

MRM depends on retention tags that are applied to items in mailboxes through policy and are automatically processed thereafter by the MFA. Retention tags can be applied to any item in any folder to specify what action Exchange should take for the item when its retention period expires. Supported actions include the hard (permanent) or soft (recoverable) deletion of the item, moving the item to a personal archive, or flagging the item for user attention. Retention policies group retention tags together in a convenient manner so administrators can apply policies to mailboxes rather than having to assign individual retention tags to folders. Retention tags and policies are organization-wide objects that are stored in Active Directory and can therefore be applied to any mailbox in the organization after they are created. Just as with Exchange 2007, the MFA is responsible for checking mailbox contents against policy and taking whatever action is determined by policy for items that exceed their retention period.

Types of retention tags

Table 11-2 describes the three types of retention tags Exchange 2010 and Exchange 2013 support. The type shown in the third column is a value passed to the –Type parameter

when you create a new tag with the New-RetentionPolicyTag cmdlet. Exchange uses this value to understand the scope of the items in a user mailbox to which it can apply the tag.

TABLE 11-2 Types of retention tags.

Tag type	Context	Target
Retention policy tags (RPT)	Administrators can apply these tags to default mailbox folders such as the Inbox, Sent Items, and Deleted Items. If an RPT is assigned to a default folder, all items in the folder automatically come under the control of the tag unless the user applies a personal tag to the item. Only one RPT can be assigned per default folder.	Supported for Exchange default mailbox folders such as the Inbox, Calendar, and Sent Items. See *http://technet.microsoft.com/en-us/library/dd297955(v=exchg.150).aspx for an up-to-date list of supported folders.*
Default policy tags (DPT)	A catch-all tag the MFA applies to any item that does not inherit a tag from its parent folder or has not had a tag explicitly applied to it by the user. In other words, if no other tag applies to an item, Exchange will respect the instructions contained in the default tag. A retention policy includes only a single DPT that is used to delete items; you can include another DPT to control the movement of items into the archive. It's logical but sometimes overlooked that if you specify two DPTs in a policy, the tag that moves items into the archive must have a shorter retention period than the tag that deletes items.	*Supported for any folder in a mailbox.*
Personal tags	Users can apply these tags to nondefault folders and individual items in a mailbox. Personal tags that move items into the archive can also be applied to default folders. Personal tags mark an item with an explicit retention, usually to comply with a business requirement. For example, you might use an Audit tag to mark items that users are compelled to retain for audit purposes. A retention policy can include many personal tags.	*Supported for any folder, item, or conversation in a mailbox.*

Microsoft originally restricted the set of default folders to which you could apply a retention policy tag to a smaller set including the Inbox, Sent Items, and Deleted Items. The set has gradually expanded, and you can now define a retention policy tag for just about every default folder, including those such as the Calendar, Contacts, and Tasks, where considerable care must be taken not to interfere with items that users often want to retain for a

considerable time. After all, no one will thank you if you clean out the CEO's calendar after 120 days!

The set of default folders includes those that often accumulate debris within mailboxes. Sync issues, Junk E-Mail, and RSS Feeds are particularly interesting in this respect. It's good to have these folders cleaned out automatically because the items stored here aren't typically needed after a day or so.

When you create a new retention tag with EAC, you select the type of tag through a drop-down list (Figure 11-5).

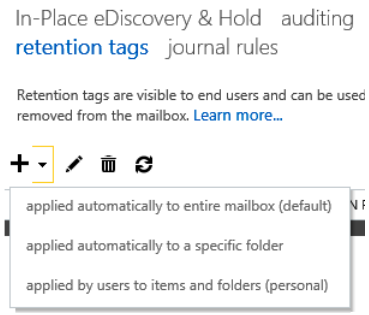


Figure 11-5 The EAC drop-down list for retention tags

The three values are:

- Applied automatically to entire mailbox This is a default policy tag (DPT). You can have two of these in a policy, one for deleting items, the other for archiving items. As pointed out in Table 11-2, if you use two default tags, the default tag used to archive items must have a shorter retention period than the tag that deletes them. Exchange applies default tags to any untagged item in a mailbox. Untagged items are those that do not inherit a tag based on the folder in which they are stored or have not had a tag placed on them by a user. Because of its influence over all untagged items in a mailbox, the default tags are critical in terms of how long items remain in a mailbox before they are deleted or archived.

- Applied automatically to a specific folder This refers to retention policy tags (RPTs) that are associated with one of the supported default folders such as the Inbox, Sent Items, and so on. You can have as many RPTs as you like for a default folder, but only one RPT for a folder can be included in a policy. EAC signals the error shown in Figure 11-6 if you attempt to add two RPTs for a default folder to a retention policy.

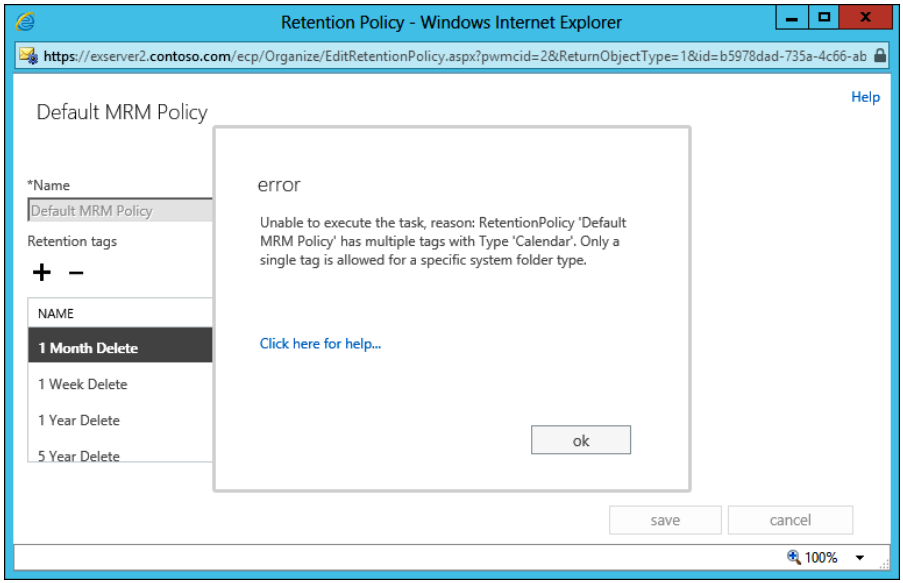


Figure 11-6 Error when attempting to add two RPTs for the same folder to a policy

- Applied by users to items and folders (personal) As the name implies, these tags are placed on folders or individual items as a result of users making a personal decision that these contain information that needs to be retained for some out-of-norm period. For instance, a folder containing items required for financial audits might need to be retained for six years and then deleted. Users could accomplish this goal by placing a personal tag with a retention period of six years and a retention action of *DeleteAndAllowRecovery* on the folder that holds the items required for audit purposes. Other items in the mailbox that are not in the folder can also be tagged with the personal tag to retain them for six years.

The range for a retention period is from 1 day to 24,855 days or, roughly, 68 years. Probably no one reading this book will worry whether an item that is 68 years old is deleted or moved in an archive, but it is good to know that such an extended period is feasible. Or maybe it's just a question of Exchange trivia.

INSIDE OUT Some items are timeless

Items in some folders tend to be more timeless than general-purpose messages, so you should think carefully through the potential consequences when you create retention policy tags for folders. For example, contacts tend not to expire, and people usually want to keep them for a long time, so it might be best to create retention policy tags that enable users to mark items not to expire. To do this, they should create a personal tag with a Never retention period, which indicates to Exchange that the item should be held indefinitely and neither deleted nor archived by MFA.

INSIDE OUT Creating a default tag for voice mail

Although I have said that you can have only two default tags in a retention policy (one to archive items, one to remove them), Exchange allows a special condition for voice mail. You cannot create a default policy tag for voice mail through EAC. Instead, you run the New-RetentionPolicyTag cmdlet through EMS to create the new default tag. After it's created, the new tag appears in EAC and can be included in retention policies. Here's an example of creating a new default tag that removes voice mail after 14 days.

```
New-RetentionPolicyTag -Name "Voicemail 14 days" -Type All -MessageClass
Voicemail -AgeLimitForRetention 14 -RetentionAction DeleteAndAllowRecovery
```

Retention tags cannot be applied to items directly. First, they have to be assigned to a retention policy and the retention policy assigned, in turn, to the mailboxes whose content you want to manage. A retention tag can be reused several times in different policies. Although there is no theoretical limit to the number of retention tags you can define for an organization, it makes sense to create a set of tags that can be shared and reused between retention policies rather than creating separate tags for each policy.

Exchange can apply only one retention policy tag and one archive tag to an item. Two simple rules are enforced when Exchange evaluates policies that it can apply to an item. The first rule states that the policy with the longest retention period always wins and is intended to ensure that Exchange never deletes an item before its time truly expires. The second rule is that an explicit policy is always respected before an implicit or default policy. If you apply a personal tag to an item to retain it for six years and the default retention policy for the folder requires deletion after 12 months, the item will be kept for six years. Personal tags

can be placed on items, conversations, or complete folders, and they are transferred with items if you move them between folders.

> **Note**
>
> **When you apply a tag to a conversation, you really just apply the tag to the items that make up the conversation at that point in time. Exchange knows that the items are part of the conversation and can apply the tag, but it won't look for and tag new items as they arrive and join the conversation. This is because a conversation is not a real storage container within a mailbox and therefore cannot be permanently tagged. In short, tags only exist in a persistent manner for folders and individual items.**

Of course, to make any sense of retention policies, you also need to deploy clients that include the necessary intelligence and user interface. The only clients in this category are Outlook 2010, Outlook 2013, and Outlook Web App. At the time of writing, no mobile client has any ability to display or set retention tags (this situation might change with updates to the Outlook Web App for Devices app). As you'll see when you review how retention policies function from a user perspective, the Outlook user interface provides the richest views of retention policies and tags. Outlook Web App is less capable but still highly usable.

System tags

Exchange 2013 supports two types of retention tags: system tags and nonsystem tags. Exchange uses system tags for its own purposes, and they are not shown when you run the Get-RetentionPolicyTag cmdlet unless you specify the –IncludeSystemTags parameter. By default, Get-RetentionPolicyTag lists only nonsystem tags (those created to be used with normal retention policies). To see the system tags defined in an organization, you can execute the following command. (Nonsystem tags are listed afterward.)

```
Get-RetentionPolicyTag –IncludeSystemTags | Format-Table Name, Type, SystemTag
```

The first three entries you will see (AutoGroup, ModeratedRecipients, and AsyncOperationNotification) are system tags that Exchange uses to prevent items from accumulating in arbitration mailboxes. The other entries are nonsystem tags, which instruct the MFA to clean out these mailboxes as items expire. To see details of the retention policy used for arbitration mailboxes and its links to the two system tags, run these commands:

```
Get-RetentionPolicy –Identity 'ArbitrationMailbox'
Get-RetentionPolicyTag –Identity 'AutoGroup'
Get-RetentionPolicyTag –Identity 'ModeratedRecipients'
Get-RetentionPolicyTag –Identity 'AsyncOperationNotification'
```

CAUTION!

You cannot add system tags to a retention policy that's applied to user mailboxes. Deleting a system tag is also a bad thing because you have no idea of what consequences might follow from this event.

Designing a retention policy

Many retention policy tags can exist within an organization. This allows great flexibility in creating appropriate policies for different groups that work within a company. For example, the finance department might want Exchange to delete everything in the Deleted Items folder permanently that is more than three days old (the shred principle), whereas users in other departments might not be concerned whether items survive in the Deleted Items folder for 30 days or more. You can apply a retention policy to members of the finance department that includes a retention policy tag for the Deleted Items folder that instructs the MFA to remove items after three days. The same policy might include a personal tag that allows members of the finance department to mark items that have to be archived for audit purposes after a month in the primary mailbox. The MFA moves items with this tag to the archive mailbox when it processes the mailbox.

Why are you creating this retention policy?

Before you rush to create a retention policy for anyone—even the finance department—you should sit down and determine the why, when, and how for the policy:

- **Why you are implementing the policy? What business need will the policy serve, and how does the policy contribute to the governance of information required by the business? How will the policy assist the business in satisfying its legal and regulatory requirements?**
- **When will you implement the policy? To what mailboxes will the policy be applied? How will you communicate the policy to end users so that they understand the purpose of the policy and how it will affect the contents of their mailboxes?**
- **How will you implement the policy? What tags and types of tags are required? What actions will you enforce through tags, and what retention periods are used? Do any restrictions exist as a result of other aspects of your deployment? For example, if you use an archiving product from another vendor, you cannot deploy tags to move items into an archive mailbox after a designated period.**

The design for a retention policy might be captured in a simple table format that makes it clear which tags are included in the policy, their purpose, and the folders MFA processes. Apart from its other advantages, capturing the design like this makes it easier to communicate the policy to users. Table 11-3 lays out a simple policy that could be applied to help managers cope with overloaded mailboxes.

TABLE 11-3 Laying out a retention policy

Retention Policy Name	***Management Retention Policy***
Applies to	***Mailboxes with CustomAttribute7 = Management***
General purpose	Automatic clean-out of Inbox and Sent Items folders to encourage users to keep these folders tidy. Items in most other folders can remain in place for a year, but calendar items are kept for five years. Removal of items from the Deleted Items folder after a week. Tags are provided to keep the contents of several utility folders under control. A tag is provided to enable users to mark items for retention for 10 years. Another is available to tag audit items so that they are held indefinitely.

TAG NAME	TAG TYPE	APPLIES TO	ACTION
Inbox 30	RPT	Inbox folder	Move items to Recoverable Items after 30 days.
Sent Items 30	RPT	Sent Items	Move items to Recoverable Items after 30 days.
Calendar 5 years	RPT	Calendar	Move calendar items to Recoverable Items after five years.
Deleted Items 7	RPT	Deleted Items	Permanently remove items after seven days.
Junk Mail 3	RPT	Junk Mail	Permanently remove items after three days.
RSS Feeds 3	RPT	RSS Feeds	Move items to Recoverable Items after three days.
Sync Issues 1	RPT	Sync Issues	Clean out synchronization error logs daily.
Delete after 5 years	DPT	All folders	Move items to Recoverable Items after 1825 days.
Archive after 2 years	DPT	All folders	Move items into the archive mailbox after 730 days.
Retain for 10 years	PER	All folders	Move items to Recoverable Items after 3650 days (10 years).
Keep for Audit	PER	All folders	Never delete—item held indefinitely for audit purposes.

Logically, you can have only a single RPT for each default folder within a retention policy. It would be very confusing to have two retention policies compete within a single folder! In addition, a retention policy can have only one default retention tag and one default archive policy tag that apply to all folders. If you use two default tags (which implies that you use

archive mailboxes), you should make sure that items can move into the archive before they are deleted. This is done by giving the default archive tag a shorter retention period than the default retention tag. For example, this will work:

- **Default archive tag retention period 2 years (730 days)** MFA will move all items that do not have another tag on them into the archive after items are more than two years old.
- **Default retention tag period 5 years (1,825 days)** MFA will remove the items (either allowing recovery or removing them permanently) after they have been in the mailbox for five years. In fact, the items were in the primary mailbox for two years, then moved to the archive, and then aged out of the archive after another three years.

Things would not run so smoothly if the two retention periods were reversed, because MFA would then delete items before they had a chance to be archived!

INSIDE OUT **Keep it simple**

Exchange enables you to create and apply as many retention policies as you want, but the question of long-term supportability arises. You should also consider the question of how many retention policies are really required for the organization as a whole and attempt to restrict the number to the minimum necessary to meet business needs. A couple of well-designed, logical policies that satisfy the vast bulk of requirements will be easier to create, deploy, and manage on an ongoing basis than a mass of granular policies generated to meet the specific needs of a department or other business group that might disappear following the next corporate reorganization. The more policies exist, the more potential there is to confuse administrators and users alike.

Managed Folder Assistant and retention policies

The MFA is responsible for implementing the actions specified in retention and archive tags when it processes a mailbox. For example, if the retention period for the Inbox is 30 days, the MFA will tag any item aged up to 30 days and take the specified action for items aged 30 days. Therefore, before you implement a policy that could affect thousands of items in user mailboxes, it is critical to communicate clearly what will happen, when it will happen, and how users can prepare for the implementation of the retention policy and respond to its actions afterward. You might have to communicate several times before the retention policies are implemented to avoid a deluge of calls to the help desk the morning after the

MFA runs. Having a well thought-out retention policy that the business agrees on and that is then clearly communicated to business leaders and end users well in advance is a good way to avoid problems when you implement this part of your compliance strategy.

Exchange uses the date and time when an item is created in a user's mailbox as the baseline to calculate the age of the item for retention purposes, so an age limit of 30 days for the Inbox default retention tag means that items become eligible for processing by the MFA 30 days after they are delivered to the Inbox. The creation date is used for retention purposes even for modifiable items such as posts. As noted earlier, you can create a tag to mark items never to be processed by the MFA. Such a tag has no value set for its AgeLimitForRetention property, and its RetentionEnabled property will be set to $False. For example, to see the set of retention tags that can be added to retention policies to allow users keep items indefinitely, you can run this command:

```
Get-RetentionPolicyTag | Where-Object {$_.AgeLimitForRetention -eq $null} |
Format-Table Name, AgeLimitForRetention, RetentionEnabled -AutoSize
```

Tags with a null age limit might have different retention actions (move to archive or delete), but in effect, they are simply different names for the same effect, which is to tell MFA to ignore them when it processes items. Having tags that do the same thing with different names is generally frowned upon because it increases the number of tags that have to be managed and can cause a little confusion, but sometimes it is justified when you want to communicate to the user the exact nature of the tag and what it will do. Keep For Audit is a good example here; the user should be under no illusion about what this tag means and what it should be used for.

Naming retention tags

When we started to work with retention policies and tags in Exchange 2010, the initial advice given from the Exchange development group was that tags should be named in a way that made their purpose immediately obvious. The idea was that you could use a prefix of RPT for retention policy tags, DPT for default policy tags, and PER for personal tags. Following a naming scheme like the one outlined earlier is logical, but it runs contrary to the way human beings absorb information. Having a tag called RPT-Inbox-30 makes perfect sense to those who are used to interpreting cryptic names that include acronyms. Thus, an administrator who is accustomed to working with retention tags understands at a glance that the name means the tag is a retention policy tag for the Inbox that likely sets a retention period of 30 days.

Client user interfaces expose the names of tags to end users. Outlook Web App exposes less than Outlook does, but the fact remains that users can become aware of tag names. It is for this reason that you need to pay attention to the names given to tags and make sure that the names are logical and convey the meaning of the tag to users.

You must determine a tag naming scheme because the retention policy menu that Outlook 2010, Outlook 2013, and Outlook Web App display lists tag names and their retention period (such as six months) to end users but doesn't display any other detail, such as the action that will be taken when the tag's retention period expires. This is apparent from the screen on the left in Figure 11-7, which is revealed when a user right-clicks an item. (We know the item is in the Inbox because the folder policy is Inbox 30.) Tags can have a variety of associated actions, from permanent deletion to merely warning that the retention period has expired, but the action the tag invokes is not immediately apparent here. Outlook Web App displays retention tags in much the same way. Logically, archive policy tags appear only if a mailbox has an archive.

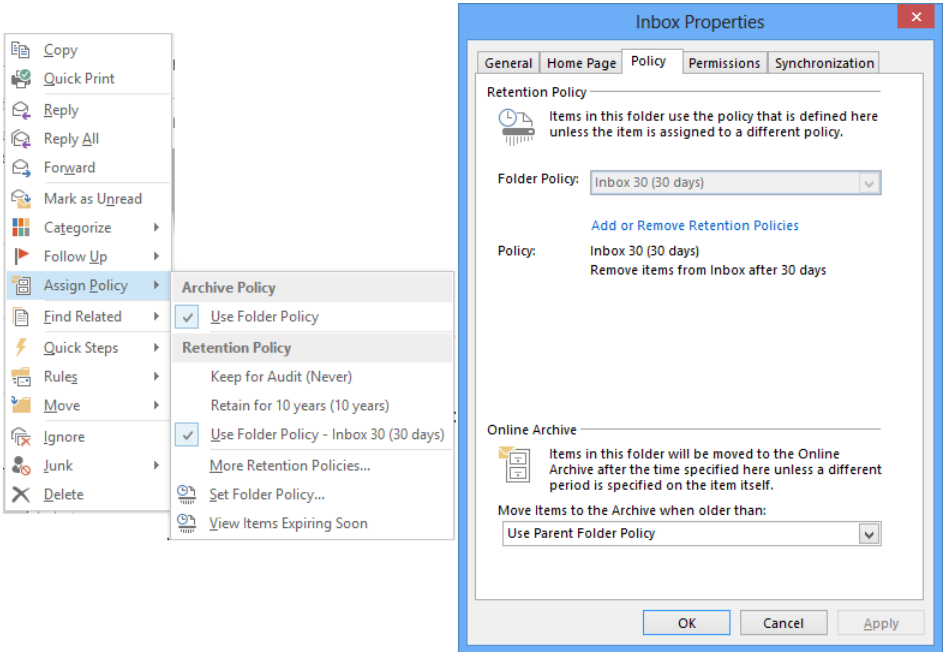


Figure 11-7 Available retention tags that Outlook 2013 lists

Outlook 2010 and Outlook 2013 users can view the actions for the default tag (if one exists) for a folder by viewing the folder properties (the right-side screen in Figure 11-7). Here you can see the action dictated by the tag, in this case to remove items from the Inbox after 30 days. However, this information is not available through the Outlook Web App user interface, and nothing at all is exposed about retention tags in the Outlook 2007 user interface. It can be argued that the tags used in an archive policy and displayed in the archive menu are an exception because users should know that the purpose of these tags is to move items into the personal archive when their retention period expires, but that's still no reason to use cryptic tag names.

The question, therefore, has to be asked whether you should use a more user-friendly naming scheme for retention tags. For example, would RPT-Inbox be better named Inbox Retention Policy, and should PER-Retain be called Retain For Five Years? Some prefer the structure of the first approach, but users usually find the second approach easier to understand.

Another approach that is often taken is to use names that give clear business directives for retention tags. For example, you might use names such as these:

- Business Critical
- Partner Negotiations
- Legal Retention

Tags named like this are usually more specific to departments or groups than more generic names such as Keep For Five Years or Required For Annual Audit, so you might have to define a set of retention tags for each department to match its work practices. The Keep For Audit personal tag specified in the prototype retention policy discussed earlier is an example of a business-specific tag name.

It's *impossible* to give a definitive answer about a naming convention that is suitable for all deployments. Some organizations are happy with cryptic tags because they are a standard that is valid no matter what language is used to connect to Exchange; others will elect to use more user-friendly names because it's easier to communicate the purpose of a retention policy to users, and they feel that this will both ease the introduction of retention policies within the organization and avoid some calls to the help desk. The important thing is to make a decision before you start to design and implement retention policies because changing the names of tags halfway through a deployment is guaranteed to cause maximum confusion.

Creating retention tags

Retention policies and tags are managed through the Compliance Management section of EAC. To create a new retention tag, click Retention Tags and then click New (+). You then select the type of tag (default, folder, or personal) you want to create. In this example, you create a tag for the Inbox folder. Figure 11-8 shows the fields you need to complete, split across two screens.

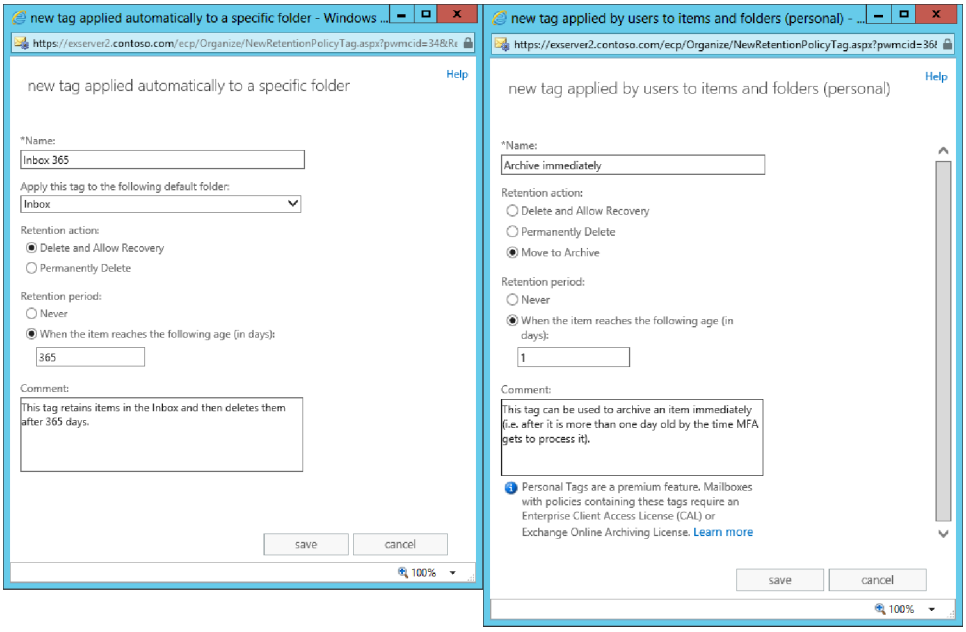


Figure 11-8 Creating a new retention policy tag

A folder tag does not offer you the opportunity to move the item into the archive because Exchange limits this option to a default tag that applies to all untagged items in a mailbox or a personal tag that a user can apply himself. If you were to create a new personal tag, for instance, EAC displays a screen similar to that shown in the right side of Figure 11-8, and you can see that Move To Archive is now available.

After creating all the retention policy tags you need, you should have something like the situation illustrated in Figure 11-9. This is the complete set of the retention policy tags defined for the organization. You can immediately see the advantage of following a well thought-out naming convention for tags. In addition, you can see how it is possible to accumulate a relatively large number of tags quickly, which different retention policies use in an organization. With some forethought, it is possible to reduce the total number of tags by designing some utility tags that are included in every policy, which then means that the only additional tags you need to define are those specifically required by a policy. For example, you can probably define utility tags to clean out folders such as Junk E-Mail and RSS Feeds that apply the same retention period and action for every policy. You might not be as successful in defining utility tags for default folders such as the Inbox or Sent Items because different sets of users might need to keep items in these folders for different periods.

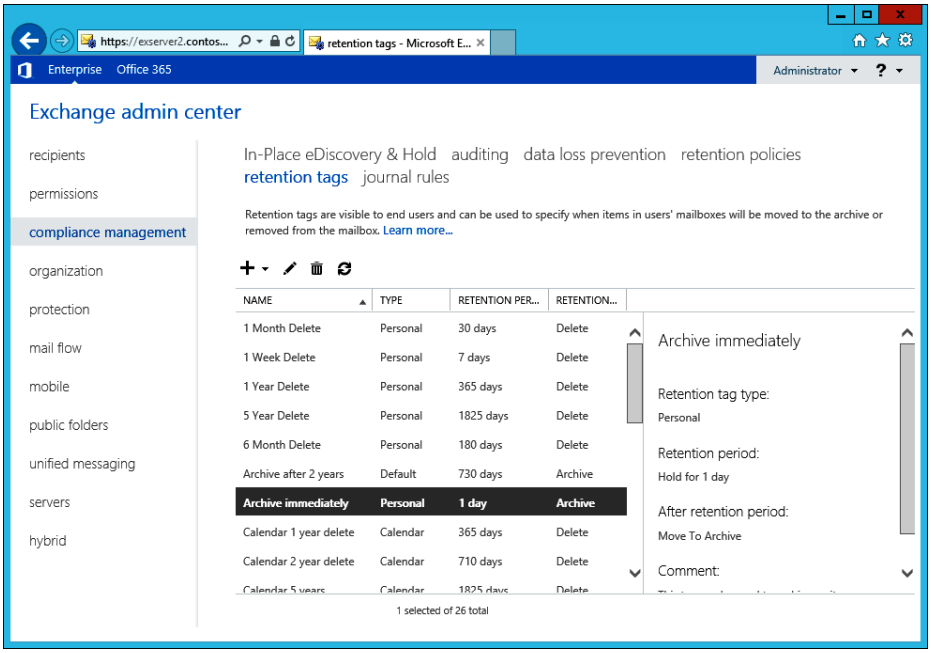


Figure 11-9 Viewing the set of retention policy tags defined for the organization

As an example of how to accomplish the same task with EMS, here is how to create the retention policy tag defined in the management retention policy for the Inbox folder.

```
New-RetentionPolicyTag -Name 'Inbox 30'  -RetentionAction DeleteAndAllowRecovery
-AgeLimitForRetention 30 -Type Inbox -Comment 'Inbox items are automatically deleted
after 30 days' -RetentionEnabled $True
```

You can check the properties of your new retention tag with the Get-RetentionPolicyTag cmdlet. For example:

```
Get-RetentionPolicyTag -Identity 'Inbox 30' | Format-List
```

You should see that the output from Get-RetentionPolicyTag confirms that the tag can cover any class of item (MessageClass = *, the default for Exchange 2010 and Exchange 2013), that it is for the Inbox folder (Type = Inbox), and that items that inherit this tag because they are stored in the Inbox folder will be moved to the Deleted Items folder after 30 days (indicated in the RetentionAction and AgeLimitForRetention properties). In fact, unlike managed folders, retention tags don't accommodate the notion of item segregation; you cannot build a retention tag that applies only to items of a certain class in a folder (such as applying the policy to items of class IPM.Note but ignoring those of class IPM.Contact).

Now review the EMS code that could be used to create some of the other tags that are required for the management retention policy. This time, a personal tag (Type = Personal) is needed to enable users to mark items to be kept in their mailbox for 10 years (3,650 days, give or take a couple of extra days for leap years), after which the items will be automatically moved into the Recoverable Items folder. Exchange gives the action and retention period defined in a personal tag priority if a user applies it to an item in a folder that's already under the control of a retention policy tag. If a user applies this tag to an item in the Inbox, Exchange will not delete it after 30 days as called for by the retention policy tag associated with the Inbox. Instead, Exchange will respect the action and retention period defined in the personal tag because the rule is that an explicit policy always trumps an implicit policy. In addition, remember that Exchange keeps a personal tag on an item even if the item moves to another folder that has an associated retention policy tag. You create the new personal tag with the following command:

```
New-RetentionPolicyTag -Name 'Retain for 10 years'' -RetentionAction
PermanentlyDelete -RetentionEnabled $True -AgeLimitForRetention 3650 -Type Personal
-Comment 'Item to be kept for ten years before it permanently deleted'
```

Setting the Type parameter to Personal is critical here because it makes the tag personal and explicit rather than implicit, like the tags applied to all items in a folder.

TROUBLESHOOTING

I created a retention tag with the wrong type. What do I do?

If you make a mistake and create a retention tag of the wrong type, you cannot change the type with the Set-RetentionPolicyTag cmdlet. Instead, you must delete the tag with the Remove-RetentionPolicyTag cmdlet (or through EAC) and then re-create it afterward with the correct type. The sooner you realize the mistake, the better because this will restrict the processing that MFA does to apply and then remove the tag from items in user mailboxes.

To complete the design, the policy needs to provide managers with two default tags. The first forces any items older than two years (730 days) to be moved into the archive; the second removes items to the Recoverable Items folder after they are five years (1,825 days) old. Remember, a default tag is used when no other tag has been applied to an item.

```
New-RetentionPolicyTag -Name 'Archive after 2 years' -RetentionAction MoveToArchive
-RetentionEnabled $True -AgeLimitForRetention 730 -Type All -Comment 'Items older
than two years are moved to the archive unless otherwise tagged'
New-RetentionPolicyTag -Name 'Delete after 5 years' -RetentionAction
DeleteAndAllowRecovery -RetentionEnabled $True -AgeLimitForRetention 1825 -Type All
-Comment 'Items older than five years are removed from the mailbox'
```

You'll have noticed that all the tags you created specify –RetentionEnabled $True. Therefore, the tag is active and should be processed by the MFA. To disable a tag, set the RetentionEnabled property to $False. The MFA ignores a tag in this state.

Most of the tags you have created so far use the DeleteAndAllowRecovery action, with the exception being the default archive tag. The full set of available actions is as follows:

- **DeleteAndAllowRecovery** Moves the item into the Recoverable Items folder so that the user can recover it if necessary.
- **PermanentlyDelete** Immediately deletes the item so that it cannot be seen using Recover Deleted Items. If the mailbox is on retention hold or subject to an in-place hold, the item is retained in the Recoverable Items folder and still available in eDiscovery searches.
- **MoveToArchive** Moves the item to a folder of the same name in an archive mailbox. This action is only possible if the mailbox has an archive. If not, the action is ignored. Moving to the archive is analogous to the Outlook AutoArchive option that moves items into a PST on a regular schedule to help keep a mailbox under quota. The big difference is that users can't vote whether they want to use the option because Exchange moves items into the personal archive automatically without asking for user opinion. Policies that move items into an archive mailbox are known as archive policies. Exchange ignores the archive tags if you create a retention policy that includes tags to move items into the archive and apply it to a mailbox that doesn't have a personal archive. If the mailbox is subsequently assigned a personal archive, the MFA applies the archive tags for the mailbox the next time it runs.

Tags that specify MoveToArchive are sometimes called move tags, whereas those that specify DeleteAndAllowRecovery and PermanentlyDelete are called delete tags. To check that all the required tags are in place that you need to build the retention policy, you can review the set of tags through EAC or execute the following EMS command:

```
Get-RetentionPolicyTag | Format-Table Name, Type, RetentionAction, RetentionEnabled,
AgeLimitForRetention -AutoSize
```

If you make a mistake in defining a retention policy tag, you can remove the tag with the Remove-RetentionPolicyTag cmdlet. If the tag has already been applied to mailbox items, the MFA cleans up by removing any reference to the removed tag from items as it processes mailboxes.

INSIDE OUT **An advantage of using New-RetentionPolicyTag**

EAC offers administrators the choice of three actions to assign to a retention tag. They can delete and allow recovery, permanently delete, or move to the archive. In fact, Exchange supports a fourth action, which is to mark the item as expired. An expired item simply means that Outlook or Outlook Web App displays an indication to the user that the item is past its best-by date, so it is less destructive in some respects than removing or archiving items. MFA eventually removes or archives expired items if default tags exist in the policy to mandate these actions, but in the meantime, it's up to the user to decide whether to keep the item (in which case she should apply a new retention policy to it) or delete it. Marking items as expired is often used to test the effect of a retention tag on items because it is less disruptive than a tag that deletes or archives an item. Here's an example of how to create a retention tag for the Inbox that expires items after 30 days:

```
New-RetentionPolicyTag -Name 'Inbox Expiry' -RetentionAction
MarkAsPastRetentionLimit -RetentionEnabled $True -AgeLimitForRetention 30 -Type
Inbox -Comment 'Anything in the Inbox more than 30 days is marked as expired'
```

Creating a retention policy

Now that you have created the necessary retention tags to help managers impose order on their mailboxes, you can create a new retention policy. Under the Compliance Management section of EAC, choose Retention Policies and click New (+). You have to name the retention policy and add the set of retention tags that enable the new policy to impose some level of retention compliance on the users to whom you assign the policy.

Retention policy names are entirely internal and are never revealed to users. The major step in building the policy is to scan the set of available tags and select the necessary ones (Figure 11-10). When the required tags are selected, click OK to return to the initial screen and then Save to create the new policy. EAC then checks to ensure that the set of tags is valid and that you haven't done something such as specifying two default retention tags or two folder tags for the same folder.

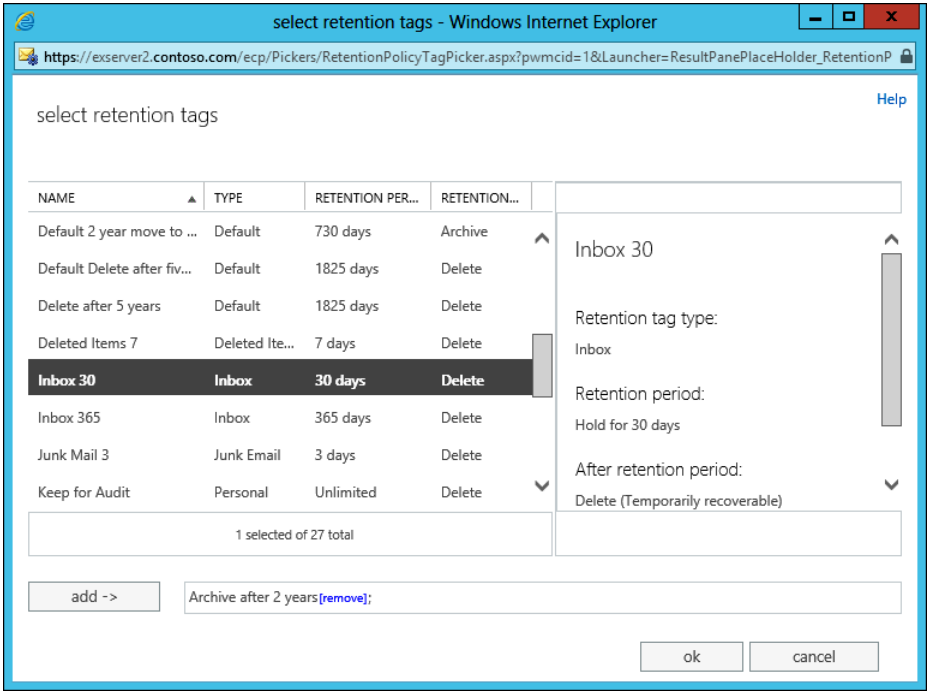


Figure 11-10 Adding retention tags to a retention policy

When EAC returns to Compliance Management, you should then see something like Figure 11-11 with the new policy listed in the set of retention policies defined for the organization and the set of retention tags used by the policy shown in the action pane.

You can also create retention policies through EMS by using the New-RetentionPolicy cmdlet. In this command, you create the policy and associate the 11 tags you want to use with the new policy. Don't worry about uppercasing the tag names as defined. EMS can locate tags as long as you spell their names correctly.

```
New-RetentionPolicy –Name 'Management Retention Policy' –RetentionPolicyTagLinks
'Archive after 2 years', 'Calendar 5 years', 'Delete after 5 years', 'Deleted Items
7', 'Inbox 30', 'Junk Mail 3', 'Keep for Audit', 'Retain for 10 years', 'RSS Feeds
3', 'Sent Items 30', 'Sync Issues 1'
```

You can examine details of the new retention policy with the Get-RetentionPolicy cmdlet:

```
Get-RetentionPolicy –Identity 'Management retention policy' | Format-List
```

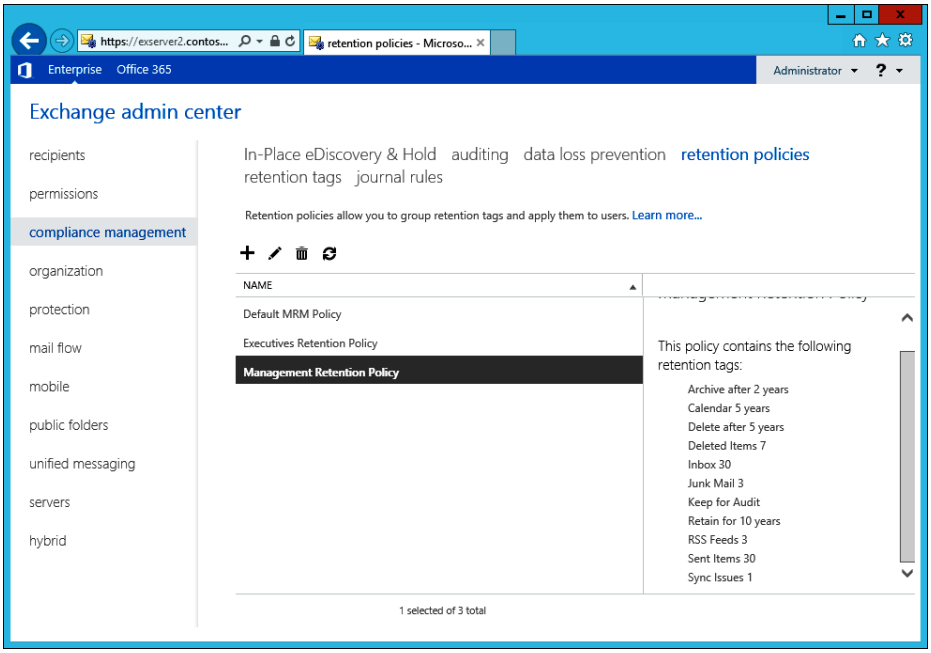


Figure 11-11 Reviewing the tags in a new retention policy

Some of the properties Get-RetentionPolicy reveals apply to Exchange Online only. For example, Exchange Online enables administrators to mark a retention policy as default (IsDefault = $True), meaning that Exchange applies this policy to users' mailboxes if another retention policy isn't assigned.

Provided there is good reason for each tag to exist in the policy, a 10-tag policy composed of some folder tags, a default tag, and some personal tags is a reasonably simple retention policy because other policies can incorporate a lot more tags to create a very exact retention environment for a user to operate within. You might have more tags than this if you decide to include a retention policy tag for every default folder. Using retention policy tags to clean out items that otherwise accumulate and are never cleared out in default folders such as Sync Issues, Junk E-Mail, and RSS Feeds is a good example of where you can gain real value from a well-designed retention policy.

INSIDE OUT Good reasons to limit the number of tags in a policy

Microsoft recommends that you have no more than 10 personal tags in a policy; otherwise, you might confuse users with too much choice. (Only personal tags usually show up in client user interfaces, hence the focus on these tags.) This is reasonable advice, but as with most advice, there will be times when you need to incorporate more tags in a policy to meet specialized business needs. A more sophisticated policy for a department might have separate retention tags for many of the default folders, a set of personal tags developed specifically to suit the retention needs of the department, and a default retention tag for everything else. User interface constraints are another good reason for limiting the number of tags in a policy. If you have five personal tags or fewer in a policy, there's a reasonable guarantee that Outlook Web App can display all the tags in its user interface. However, if you include 20 personal tags in a policy, you'll find that Outlook Web App cannot list all the tags, and some will simply drop off the end of the available list. This is especially true when you consider the relatively small display screen of some of the devices that Outlook Web App now supports.

Outlook 2013 takes a different approach from Outlook Web App by displaying up to 10 personal tags when a user selects an item, right-clicks, and selects Assign Policy from the shortcut menu. This list includes an entry for More Retention Policies that reveals the full set of personal tags defined in the policy. As you can see from the rightmost screen in Figure 11-12, the full set of retention tags for the policy assigned to a mailbox contains 14 tags, many of which are of dubious value. Using a separate dialog box keeps the Outlook user interface under control while still permitting users to discover all the tags available in a policy at the expense of requiring separate clicks and additional knowledge of where to find the tags. For all these reasons, it's just a bad idea to create a tag-filled policy.

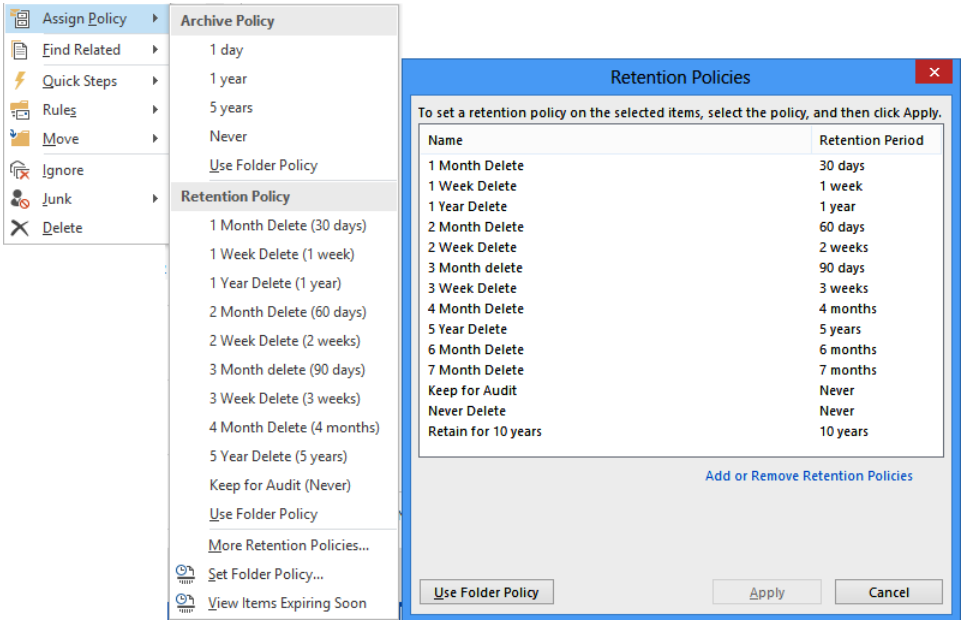


Figure 11-12 How Outlook 2013 displays the full set of personal tags to users

INSIDE OUT Calendar and Tasks

Although retention policies support tags for calendar and task items, no client currently exposes the necessary user interface to enable users to apply policies to these folders. In some respects, this is a good thing because it means that administrators are the only people who can set a retention policy for these important folders. Some might disagree because they want to apply a different retention policy, but that's the current situation.

Applying a retention policy to mailboxes

You can apply retention policies to mailboxes by:

- Selecting mailboxes individually in EAC and editing the mailbox properties to assign a retention policy. This option is acceptable when you have to deal with only a few mailboxes. You cannot set a retention policy on a mailbox when it is initially created with EAC; this has to be done afterward, possibly because the use of personal tags

in a retention policy requires an enterprise CAL. The standard CAL permits the use of default and folder tags in a retention policy.

- Selecting multiple mailboxes and applying a Bulk Edit. This exposes options available for all the selected mailboxes in the action pane. The option to enable an archive for all the selected mailboxes is available through the More Options link at the bottom of the action pane.
- Running the Set-Mailbox cmdlet to apply retention policies to a group of selected mailboxes. This is clearly a more efficient approach to take when you have to deal with more than a few mailboxes at one time.

When assigned to a mailbox, the policy becomes active the next time the MFA processes the mailbox. At this point, MFA writes the policy information into the mailbox. (See the "Behind the scenes with the MFA" section later in this chapter.) Until MFA processes the mailbox, the policy tags will not appear in the client user interface, so the lack of these tags is a good indication that the MFA has not yet gotten to a mailbox. You can force the issue by running the Start-ManagedFolderAssistant cmdlet as follows, passing the name of the mailbox you want MFA to process:

```
Start-ManagedFolderAssistant -Identity 'Tony Redmond'
```

If you're setting a policy for a group of users (see Figure 11-13), you'll probably do it in one operation by selecting the mailboxes with the Get-Mailbox cmdlet and piping the results to Set-Mailbox. For example:

```
Get-Mailbox -Filter {CustomAttribute7 -eq 'Management'} | Set-Mailbox
-RetentionPolicy 'Management retention policy' -RetentionComment 'Management
retention policy applies to this mailbox'
```

To discover the set of mailboxes that have retention policies in place, you can use a command like this:

```
Get-Mailbox -Filter {RetentionPolicy -ne $Null} | Format-Table Name, RetentionPolicy
-AutoSize
```

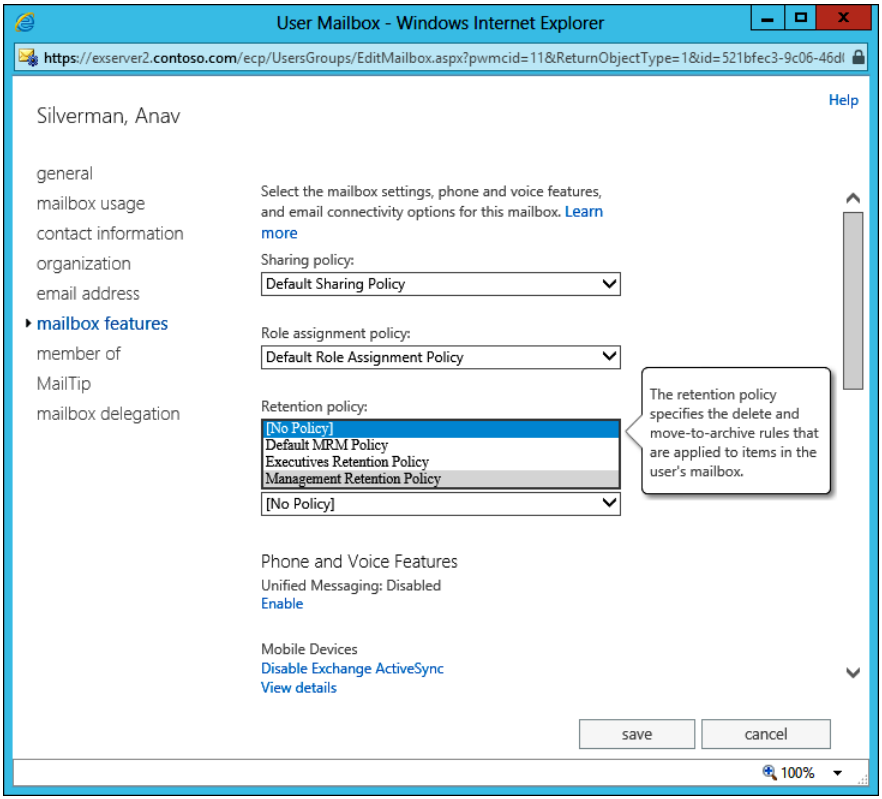


Figure 11-13 Assigning a retention policy to a mailbox

INSIDE OUT Only one retention policy—ever

A mailbox can only ever have one retention policy, so when you assign a retention policy to a mailbox, the action overwrites any policy that might already be in place. You can change retention policies multiple times on a mailbox, but this isn't a good idea unless you really need to switch policies because the effect of the different policies might confuse users; the MFA responds to different retention settings in the different policies. Setting a value for the RetentionURL parameter is not compulsory, but it is a useful way to communicate how a user might find additional details about the company's retention policy. This URL is visible only through Outlook 2010 and Outlook 2013 (Figure 11-14) and isn't displayed by earlier clients or Outlook Web App.

```
Set-Mailbox -Identity 'TRedmond' -RetentionPolicy 'Management retention policy'
-RetentionComment 'Management retention policy applies to this mailbox' -RetentionURL
'http://Intranet.contoso.com/RetentionPolicies.html'
```

By looking at the parameters that set retention information with the Set-Mailbox command, you can see that the RetentionComment property provides the text you can see beside Account Settings. The RetentionUrl property populates the URL for the More Information link, hopefully taking the user to a website on which she can find some additional information to explain why a retention policy is necessary and what it means to the user. It is good practice to set up such a website and populate it with some practical examples of what a retention policy means to a user. For instance, it could explain to users that items in some of the default folders are automatically cleared to the archive after a certain period. Taking such a step might just save some expensive help desk calls!

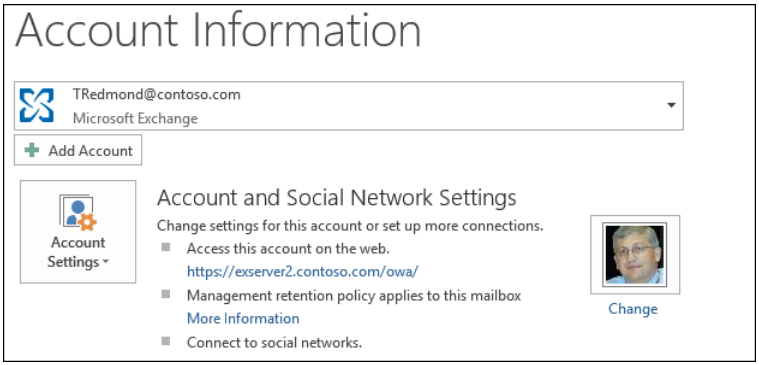


Figure 11-14 How Outlook shows retention policy information in its backstage area

It is common practice to set these properties when you place a mailbox on any type of hold (in-place or retention). These topics will be discussed in the following sections. For now, although you don't have to set these properties to impose an effective retention regime, they are helpful to communicate information to users about what's going on in their mailbox. Experience with many projects demonstrates that anything that assists in effective communications with users is likely to reduce help desk calls.

The value of $Null

When you want to remove a retention policy from a mailbox, you just set its policy to *$Null*. For completeness, it's a good idea to also set the other properties associated with retention policies to $Null. Here's the command:

```
Set-Mailbox -Identity 'JSmith' -RetentionPolicy $Null -RetentionComment $Null
-RetentionURL $Null
```

After you begin to deploy retention policies to mailboxes, the question arises of how to integrate the assignment of retention policies with any user provisioning process your company has in place. Unless you also enable an archive for a mailbox when it is created, Exchange won't assign a default retention policy automatically, so an explicit administrative action is usually required to allocate a retention policy to a mailbox. This action is not difficult to code with EMS, but it is something that needs to be considered as part of your deployment plan.

INSIDE OUT Managing retention policies and tags in a hybrid environment

If you run a hybrid environment in which some mailboxes are on-premises and some are on Office 365, you have to synchronize retention policies and tags from the on-premises organization to the cloud tenant domain to ensure that the same policy applies to all customers. You can create the retention policies and tags within Office 365 or you can use the scripts provided by Microsoft. Export-RetentionTags.ps1 exports details of retention policies and tags from an on-premises organization to a file in XML format, and Import-RetentionTags.ps1 imports the data from a previously exported file. Regretfully synchronization is not automatic and has to be done on a regular basis.

Modifying a retention policy

Policies can evolve over time by the addition or removal of tags. The easiest way to modify a retention policy is to edit it with EAC; the console will take care of all complexities involved in selecting whatever tags you want to add to or remove from the policy and then making sure that the tags are processed correctly. If you want, you can do the same work with EMS, but this is not recommended because great care must be taken to ensure that the retention tags are written correctly into the policy.

Customizing retention policies for specific mailboxes

You can tailor the retention policy for a specific user by assigning personal tags on a per-mailbox basis. You can do this only if a retention policy already applies to the user's mailbox. For example, assume that you want to assign a new personal tag to a user so he can mark an item to be moved into the archive after a year. You can do this as follows:

```
Set-RetentionPolicyTag –Mailbox JSmith –OptionalInMailbox '5 Year Delete'
```

Exchange adds the optional tag to the set of tags covered in the retention policy that already applies to the mailbox and makes the expanded set available the next time the user connects. Unfortunately, no cmdlet is available to report whether a mailbox has been assigned optional tags. If you examine a mailbox with Get-Mailbox, it tells you only whether a retention policy is assigned. Therefore, if you want to change the list of optional tags assigned to a mailbox, you have to write the complete list with Set-RetentionPolicyTag. For example, to add a tag to the one that has already been assigned, use this command:

```
Set-RetentionPolicyTag –Mailbox JSmith –OptionalInMailbox '5 Year Delete', '2 Year
Delete'
```

EMS doesn't validate that the tags you assign to a mailbox will be effective. For example, you can assign a new archive tag to a mailbox that doesn't have a personal archive. This is really a null operation because neither Outlook Web App nor Outlook displays archive tags if the mailbox doesn't have a personal archive.

To remove all optional retention tags from a mailbox, set the list to *$Null* as follows:

```
Set-RetentionPolicyTag –Mailbox JSmith –OptionalInMailbox $Null
```

INSIDE OUT Accessing personal tags through Outlook Web App

Exchange makes accessing personal tags easier by allowing users to see a list of available personal tags through the Outlook Web App Options list to decide which personal tags they would like to use. Figure 11-15 shows the option exposed through the Organize Email section of Outlook Web App Options. A retention policy must be in effect for a user's mailbox, and the MyRetentionPolicies setting of the user role assignment policy (RBAC) that applies to the mailbox must be allowed (Figure 11-16) before Outlook Web App reveals personal tags. If shown, the user sees the personal tags that she can already use because they are included in the retention policy (listed as Required) and the other personal tags that are defined for the organization that she can choose to use (listed as Optional). The user cannot remove any of the Required tags because their presence is mandated by the retention policy that is applied to the mailbox. A user can begin to apply personal retention tags to items immediately after adding the tags to her mailbox.

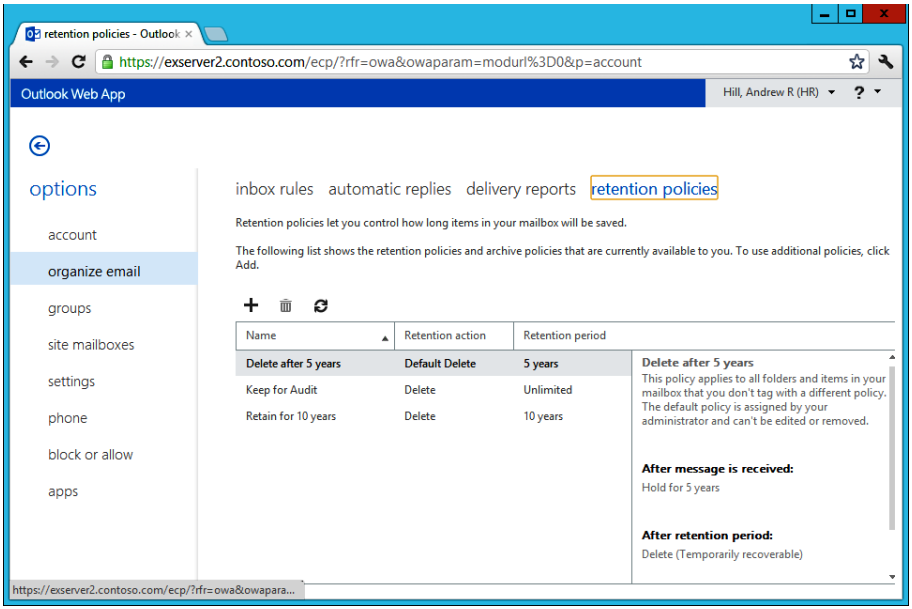


Figure 11-15 User access to personal retention tags through Outlook Web App Options

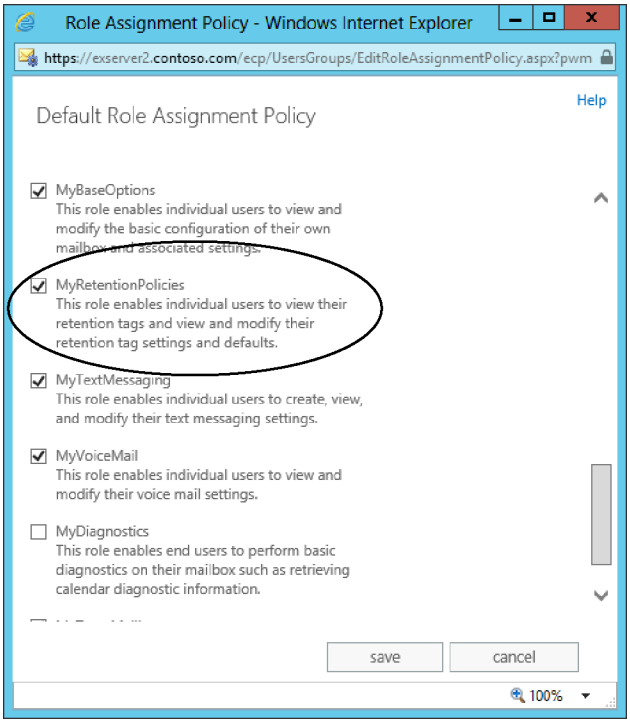


Figure 11-16 Amending a user role assignment policy to reveal personal retention tags

User interaction with retention policies

The first evidence users see that their mailbox has been assigned a retention policy is when retention information appears when they look at messages. This information is based on the tag stamped on an item by the MFA. Thirty days or so before an item's retention period expires, Outlook and Outlook Web App begin to inform users that they might want to take action to preserve the item; otherwise, the MFA will process it again and delete it or move it into the archive, depending on the action required by policy. These warnings are visible when a message is opened or shown in the message preview. Figure 11-17 shows how Outlook Web App advises that a message has five days before it expires as the result of the retention policy tag placed on the Inbox. The user now has the choice either to take action or to let the message expire, in which case the MFA will process whatever action is defined in the tag.

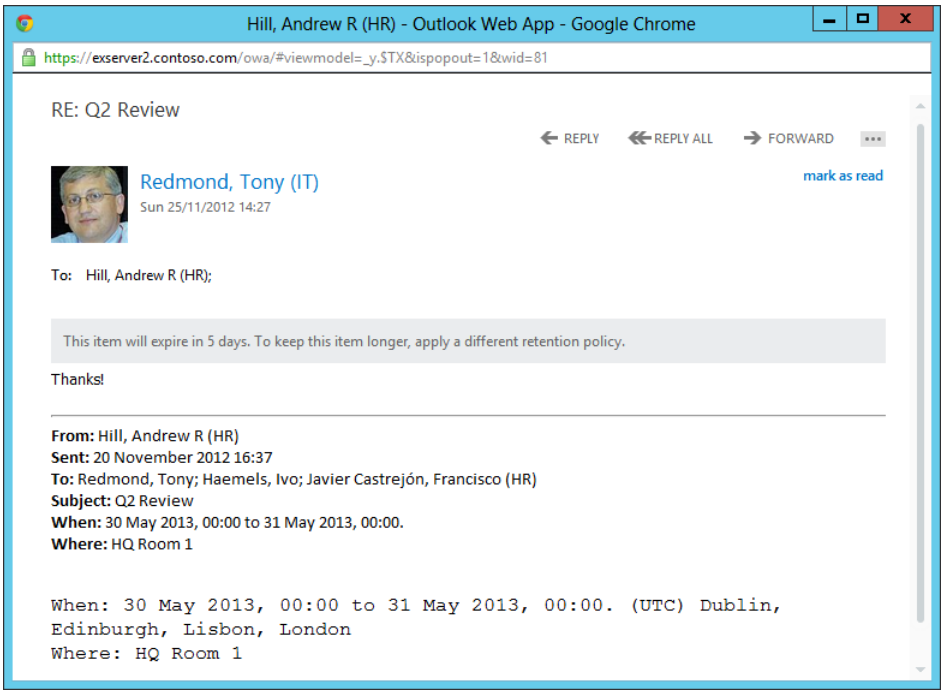


Figure 11-17 Outlook Web App warning that an item is approaching its expiry deadline

Users have two options. First, they can move the item to a different folder and so remove it from the influence of the retention policy tag that applies to Inbox items. After it is moved, the item is governed by the default policy tag defined in the retention policy that applies to the mailbox, if one exists, or by an explicit tag that is applied to the folder and therefore inherited by all items that are added to the folder. If neither of these conditions exists, the item is left untagged and is therefore not subject to processing by the MFA.

The second option is to apply a personal tag to the item. Users can choose from any of the personal tags defined in the retention policy applied to their mailbox by right-clicking an item and then selecting the personal tag to apply.

You won't see the user interface for retention policies unless a policy is applied to your mailbox and the MFA has processed the mailbox to apply the retention policy. As part of this process, the MFA creates a hidden folder-associated item (FAI) in the user mailbox that clients use to populate the retention tag picker. If the policy is subsequently updated with a new retention or archive tag, the new tag will not be visible to clients until after the MFA next processes the mailbox.

After a personal tag has been applied to an item, the item is no longer subject to the provisions of either the folder policy or the default policy because an explicit tag always takes precedence over a tag placed on a folder. The personal tag also remains with the item if it is moved to another folder or into the personal archive. If users want to impose a different retention policy on the item, they must replace the existing tag with a new personal tag.

The Managed Folder Assistant automatically applies the retention policy

Although most retention policies include a default policy tag to provide a catch-all retention action for items not tagged by other means, Outlook and Outlook Web App support the use of personal retention tags to set a specific retention policy on a folder. Exchange applies the policy defined in the personal retention tag to items held in the folder in much the same way it applies the retention policy tags placed on default folders such as the Inbox. In some respects, you can use this approach to create a roughly equivalent situation to the functionality that Exchange 2007 managed folders provides. However, you have to create the folders and apply the retention policies manually, whereas the MFA does the work to push out new folders to user mailboxes and apply the retention policy automatically for managed folders.

Setting a retention policy on a folder

Default folders such as the Inbox or Sent Items probably come under the control of a folder retention tag included in the retention policy that's applied to a mailbox. You cannot override the policy set on default folders, but you can apply a different policy to user-created folders by assigning a personal tag to the folder. All the items that are held in the folder will then inherit the tag placed on the folder unless they in turn are tagged with a different personal tag.

To set a new policy for a folder by applying a personal tag using Outlook 2013, select the folder and click the Assign Policy icon on the toolbar. Outlook then displays a list of the personal tags. Click one of the tags to select it. Outlook puts a check mark beside the chosen tag to indicate that it will apply this tag to the folder; the items in the folder will inherit the tag from the folder. Another way of accomplishing the task is to select the properties of a folder by right-clicking the folder in the Outlook folder list and then choosing to the Policy tab (Figure 11-18).

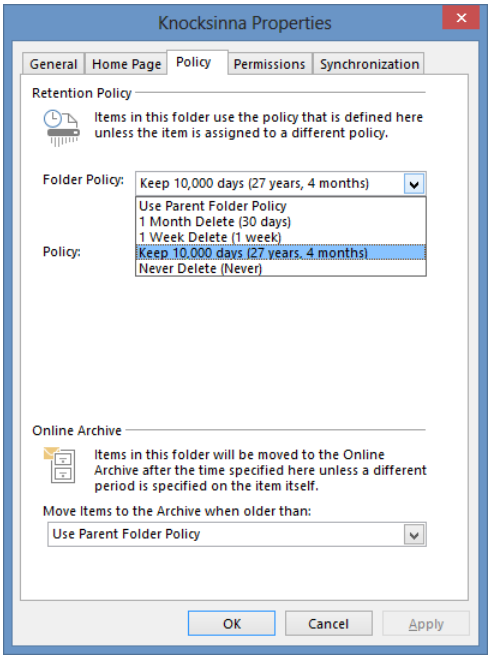


Figure 11-18 Applying a personal retention tag through folder properties

If a policy tag is in force for the folder, it will be shown as the current folder policy. If you attempt to apply a personal tag on a default folder that is controlled by a retention policy, you won't be able to select a new personal tag to apply. If the folder policy is listed as Use Parent Folder Policy, no specific personal tag has been applied to the folder, and the MFA will apply the default retention tag to the items in the folder. This is also the case for archive policies, shown in the Online Archive area at the bottom of the screen, where you can see that the folder uses the default archive tag.

You can select any personal tag available to the mailbox through its assigned retention policy and apply it to the folder. Outlook displays the available personal tags in a drop-down list (Figure 11-18). When selected, Outlook updates the retention data stored in the mailbox (which makes the new policy visible to the user). In the background, the MFA applies

the policy to the folder and the items contained in the folder the next time it processes the mailbox unless an explicit personal tag has already been applied to an item in the folder, in which case the existing tag is retained. To set a default policy on a folder with Outlook Web App, select the folder from the folder list under the Mailbox root, right-click to select Apply Policy, and then select the personal tag to apply to the folder.

Removing tags from policies

You remove a retention tag from a policy by selecting the policy in EAC and editing the list of tags, removing the one you no longer use. When you remove a tag like this, the MFA erases details of the tag from the policy information that is held in user mailboxes. This action makes the tag unavailable (invisible) to the user. If the tag is replaced by another tag, like when the tag for a default folder is replaced by another tag for the same folder, MFA applies the new tag to the folder the next time it processes the mailbox.

However, if a personal tag is removed from a policy, the MFA leaves the tag in place on whatever items to which the tag has been applied and continues to process the items according to the policy expressed with the now-removed tag. This is because the tag is still a known object within the Exchange configuration data, so the tag stays in effect until it is replaced in some way, such as by a different personal tag being applied. Despite the fact that the user can no longer apply this personal tag to other items, it will be shown to the user if he views an item to which the tag was applied when it was included in the policy.

If you delete a tag, it is removed from Active Directory. The MFA cannot process a nonexistent tag, so it has to search all mailboxes to discover items that were stamped with the now-deleted tag and restamp these items with whatever tag is now applicable, such as the default tag. Searching a Mailbox server to locate and then process potentially hundreds of thousands of items is obviously not good for server performance, and this will happen across every Mailbox server in the organization unless the deleted tag was used only for mailboxes on specific servers, which is very hard to predict and control. Therefore, you should not delete a tag without good reason. If you want to remove a tag, it is always better to just remove the tag from the policies when it is present rather than deleting it from the organization.

You can also disable a tag by editing its properties to set the retention period to Never. The MFA still considers the tag to be valid, but it ignores items that are stamped with the tag and never deletes or moves these items. In addition, although the tag is disabled, the MFA will not apply the default tag in the policy to these items, so they are essentially in limbo with respect to retention. Disabling a tag is a good way of stopping the MFA from processing items while keeping the items stamped so that they can be reactivated for retention purposes by just editing the tag to set its retention period to some number of days.

Changing a retention tag: An exception to the rule

Changing the retention period in a tag is similar to removing a tag. All items stamped with the retention period up to when you made the change continue to use that retention period; items that are stamped with the updated policy will have the new retention period. The exception to this rule is when you set the RetentionEnabled property of a tag to $False; this value instructs the MFA to leave the tags in place but ignore them when it processes items. For example:

```
Set-RetentionPolicyTag –Identity 'Keep Items Forever' –RetentionEnabled $False
```

This situation continues until the user explicitly assigns a replacement tag to an item or you remove the tag from Active Directory by using the Remove-RetentionPolicyTag cmdlet. When this happens, the MFA removes the deleted tag from any items on which it was used the next time it runs.

Removing a retention policy

The Remove-RetentionPolicy cmdlet removes a retention policy from the organization. For example:

```
Remove-RetentionPolicy –Identity 'Retention Policy - PR Department'
```

Removing a retention policy removes the policy from any mailboxes to which it is currently applied. If any mailboxes are associated with the policy, EMS prompts you to confirm its removal. If you proceed, Exchange removes the reference to the now-deleted policy from the mailboxes. Exchange can't decide what retention should replace the one that has just been removed, so no policy is applied. Locating the mailboxes to which a retention policy is applied is therefore a proactive step you should take before you remove the policy. You can scan mailboxes to discover where a retention policy is applied with a command such as this:

```
Get-Mailbox | Where {$_.RetentionPolicy –eq "Retention Policy - Audit Department"} |
Select Name
```

A similar set of commands can be run to locate mailboxes with a specific retention policy and assign a new retention policy to the mailboxes. For example:

```
Get-Mailbox | Where {$_.RetentionPolicy –eq "Retention Policy – Audit Department"} |
Set-Mailbox –RetentionPolicy 'New Retention Policy for Auditors'
```

Upgrading from managed folders

Managed folders are now a deprecated feature, and the code that supports these folders will be removed from Exchange in a future version. It's therefore important to upgrade managed folders to equivalent retention tags as soon as possible. You can do this by using

managed folders as templates to create new tags. For example, assume that you have a managed folder called Never Delete that acts as a repository for items that users never want to have removed from a mailbox because they are so important. You could argue that these items could be stored in an archive mailbox. However, archive mailboxes didn't exist in Exchange 2007, and it takes time for people to change their behavior. You can use a command such as the one shown here to create a new retention policy tag from the Never Delete managed folder:

```
New-RetentionPolicyTag –Name 'Mark item to never expire' –ManagedFolderToUpgrade
'Never Delete' –Comment 'Tag created from old Never Delete managed folder'
```

To complete the process, you must associate the new tag with a retention policy and assign it to a user, and then the user can apply the new tag to any item in his mailbox rather than just to the items placed in the managed folder.

How the Managed Folder Assistant implements retention policies

After you apply a retention policy to a mailbox, you can either wait for the next scheduled run of the MFA or start it manually so that the new policy is applied immediately. When the MFA runs, it performs the following tasks:

- It applies the tags specified in retention policies to the mailboxes covered by these policies and stamps the items in the various folders covered by the policies with the appropriate tag name and expiration date.
- If a policy defines a retention or expiry period for items, it stamps a Messaging Application Programming Interface (MAPI) property (ElcMoveDate) on the items, indicating the date and time from which the retention period will start. A future run of the MFA can then use this date and time to calculate when to delete an item or mark it as expired.
- It locates items in folders that are past their expiration date and takes whatever action is defined in the policy (delete, age out, move to another folder).

The MFA works on a work cycle basis: it assesses the expected workload in terms of the number of mailboxes it has to process and then spreads out its processing across the complete window. For example, if 600 mailboxes are to be processed over three hours, the MFA creates its own internal schedule to process 200 mailboxes per hour or roughly three mailboxes per minute. In addition, a checkpoint is defined for the work cycle, at which time the MFA looks for new mailboxes that should be added to its list for processing. The default values for the work cycle and checkpoint are both 1 day, so the MFA attempts to process every mailbox in its list daily and checks for new mailboxes daily. The MFA logs event 9017

when it begins to process mailboxes, but the more important event to look for is 9025, logged when the MFA has had to skip a mailbox for some reason, such as when the mailbox is being moved to another database.

Overall, the work cycle mechanism makes more effective use of server resources in an easy and relaxed manner throughout the day and doesn't create potential spikes in demand.

You might want to run the MFA immediately, perhaps to apply a policy to a group of users for the first time. Forcing immediate execution for a selected mailbox is useful when you start to apply policies to mailboxes and want to gauge the effect of the policy by examining the contents of a known mailbox. This might be easier than asking users what happened to items in their mailboxes (especially if you've made a mistake with the policy and just removed half the items from the mailbox). To force processing for a selected mailbox, specify its name with the –Identity parameter:

```
Start-ManagedFolderAssistant -Identity 'Akers, Kim'
```

To process a group of mailboxes, either provide a set of mailbox identifiers as input or use the Get-Mailbox cmdlet with a filter to retrieve a set of mailboxes and pipe it as input to Start-ManagedFolderAssistant. In the first example, two mailbox identifiers are provided as input. In the second, you process all the mailboxes in a database. In the third, you use a filter to find all the mailboxes from a particular office.

```
"Redmond, Tony", "Akers, Kim" | Start-ManagedFolderAssistant
Get-Mailbox -Database 'VIP Data' | Start-ManagedFolderAssistant
Get-Mailbox -Filter {Office -eq 'Dublin'} | Start-ManagedFolderAssistant
```

The time required for the MFA to complete its run depends on the number of mailboxes and the number of items to which it has to apply retention policies. A run on a small server that hosts a few hundred mailboxes will complete in a couple of minutes unless the mailboxes hold thousands of items. However, processing 7,000 mailboxes, each of which holds an average of 20,000 items, could take several hours, especially if the server is loaded with other tasks or the policies cause a heavy I/O load because many items are permanently removed or moved from primary to archive mailboxes. You should monitor the first runs of the MFA on a server to gauge the scope of the activity and how long a normal run takes to complete. Equipped with this information, you can quickly assess whether future runs are progressing as expected.

After the MFA has applied a new policy to a mailbox, the next time the user connects to the mailbox with a client that supports retention policies, she will see that retention tags are shown on items and the retention policy options are visible. Another important point that you should understand is that if you apply a retention policy that contains a default policy tag, the MFA stamps the default tag on every item in the mailbox. This action forces Outlook to download the complete contents of the mailbox the next time the client

connects and synchronizes with Exchange. Such a massive synchronization has the potential to flood a network and keep clients fully occupied for a long time. Including a default archive tag in a policy does not have the same effect because the MFA does not stamp every item with this tag.

Behind the scenes with the MFA

When the MFA first processes a mailbox, it creates a hidden FAI item of type IPM. Configuration.MRM in the Associated Contents table of the Inbox folder. The item stores MRM configuration in XML format in a PR_ROAMING_XMLSTREAM property. (PR = property; Roaming means that the information is available wherever the client connects, and XMLSTREAM indicates the type.) The MFA then updates the item any time a change occurs in the retention policy assigned to the mailbox, such as when a new personal tag is added to the policy. If a new retention policy is assigned, the MFA updates the item with details of that policy.

Exchange uses FAIs as a means to hold configuration and other data it needs to store in a mailbox but does not want to reveal to users when they run clients such as Outlook. The item the MFA creates holds details of the retention policy that has been assigned to the mailbox, including details of the retention tags the client needs to display in its user interface. If the item does not exist in the mailbox, a client remains unaware that a retention policy is in force. This is why the MFA has to process a mailbox before the client user interface is populated with details of the policy. In addition to the tags provisioned through the retention policy, the item also holds information about any personal tags the user has selected for his own use through Outlook Web App Options (see Figure 11-15). As noted earlier, you cannot see this information with a normal client, but you can with the MFCMAPI utility by opening a mailbox that has been assigned a retention policy, opening the Inbox folder, and then opening the associated contents table before finally identifying the MRM item in the set held in the table. Figure 11-19 shows the details of the MRM configuration item as exposed by MFCMAPI, and you can clearly see details of the retention tags specified in the policy in the text box.

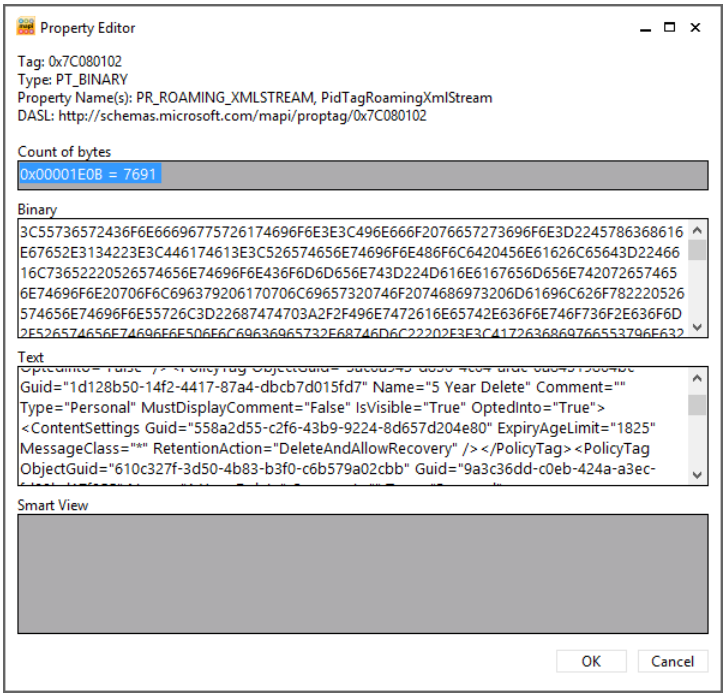


Figure 11-19 Viewing the MRM configuration item with MFCMAPI

The MFA uses a number of MAPI properties for mailbox folders and items during its processing. These properties are:

- **PR_START_DATE_ETC** For a folder, this property holds the GUID of the retention tag that governs a folder. Every tag is assigned a GUID (stored logically in the GUID property and visible with EMS). This GUID is stamped into this property so the MFA knows which retention tag, action, and period applies to the folder. For an item, the property holds a composite value containing the default retention period plus the start time for the retention period. By adding the retention period to the start time, the MFA determines the expiry date.

- **PR_RETENTION_PERIOD** This is the number of days items should be retained in a folder. When a personal tag is applied to an item, this property is set. However, if the property does not exist, the item (or subfolder) inherits the retention period from its parent folder.

- **PR_RETENTION_DATE** This is the calculated date when an item's retention period expires. Clients display this information to users. When clients work in online mode (such as in Outlook Web App), Exchange takes care of calculating this value, otherwise clients that work in offline mode perform the calculation. otherwise.

- **PR_RETENTION_FLAGS** For a folder, this flag indicates whether the retention tag is inherited from the parent folder. If the user sets an explicit tag on a folder, the value is nonzero.
- **PR_POLICY_TAG** This exists only for items and contains a binary encoded value pointing to the policy tag that governs the item.
- **PR_ARCHIVE_TAG** This exists only for items and points to the archive tag that governs an item.
- **PR_ARCHIVE_PERIOD** This exists only when an item has been stamped with an explicit archive tag to contain the number of days in the archive retention period.
- **PR_ARCHIVE_DATE** This contains the date when an item will be archived.

You can view these properties through MFCMAPI. Figure 11-20 shows the value of the PR_RETENTION_DATE property for an item.

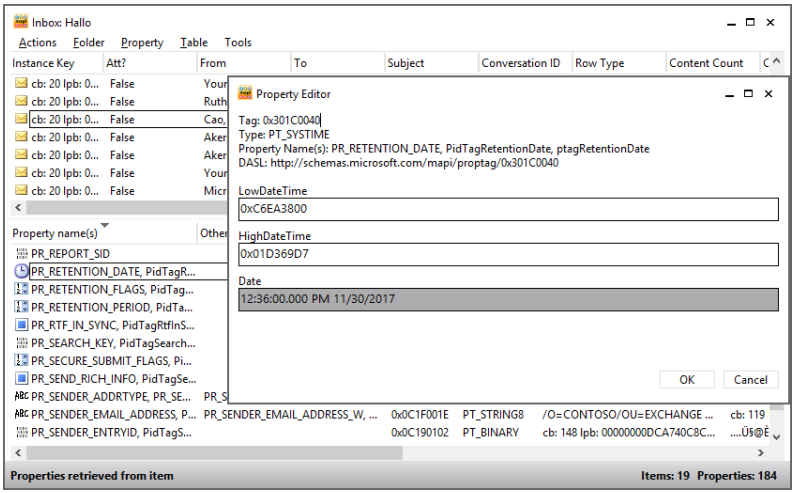


Figure 11-20 Viewing the PR_RETENTION_DATE property for an item

Retention date calculation

Date calculation is a very important part of the work the MFA does when it processes items. The MFA has to understand the date that should be used to calculate the age of the item (when the item first appears in the mailbox) and either the date when the item will expire or the date when the MFA has to take the action the retention policy requires (deleting the item or moving it into the archive).

Given the nature of email, most items enter a mailbox when they are delivered in the Inbox or are sent and stored in Sent Items. Items often stay in these folders for much of their lifetime and, if they do stay, will probably come under the control of either a folder tag (if defined for the folder) or a default tag (that applies to untagged items in the mailbox). The exception, of course, occurs when a user explicitly applies a personal tag to an item in the Inbox or Sent Items.

When the MFA runs, it examines items in the mailbox and determines what processing is required. Assume that an Inbox item exists that was delivered on 1 April 2013 and that a folder tag that requires items to be deleted and allow recovery after 30 days is applied to the Inbox. MFA stamps this item with a start date of 1 April 2013 by updating the PR_START_DATE_ETC property. It then calculates the expiration date by adding 30 days to the start date; 1 May 2013 is the result. This date is then stamped on the item by updating the PR_RETENTION_DATE property. On or after 1 May 2013, the MFA returns to the item and discovers that its retention period has expired. (If you view the item by using Outlook before the MFA returns, you'll see the item marked as Expired.) When the MFA processes an expired item, it takes the retention action defined in the tag. In this case, it moves the item into the Recoverable Items folder, where it will remain until the retention period for the mailbox expires, at which time the MFA will remove the item permanently from the database.

This is the simplest kind of processing the MFA is called on to perform, but it happens for a large percentage of items because many users leave messages in the Inbox and Sent Items folders. People who let items accumulate in these folders are often called pilers because they create piles of messages and then rely on client search facilities to locate specific items when required.

Broadly speaking, the other kind of user behavior is represented by the filers, or people who move items from the Inbox and Sent Items into more appropriate folders in which the items form collections that represent important categories of work (or play) as viewed by the user. Or indeed the user attempts to keep the Inbox and other folders under some form of control by deleting unwanted items on a regular basis.

Assume that your user receives a message, reads it, and then comes back to the message a week or so later when its content is no longer required, perhaps because he has acted on a request that was contained in the message. Because the item has been in the mailbox for more than a day, the MFA should have already processed it and stamped its properties with a start and an end date for retention purposes. The user now deletes the message as part of his cleanup effort, and the client moves the item into the Deleted Items folder.

You now have a different retention context because the item has moved out of the sphere of control exerted by the folder tag placed on the Inbox. Retention policies usually include a folder tag to control the Deleted Items folder and, if so, the MFA now must apply the

instructions contained in that tag. Otherwise, the MFA will apply the instructions contained in the default tag.

Assume that the folder tag assigned to Deleted Items requires items to be removed from this folder (using Delete And Allow Recovery) after seven days. The MFA examines the item and discovers that its start date is 1 April 2013 and that it should now be removed on 8 April 2013. If the user moved the item into Deleted Items before 8 April 2013, the item is kept until 8 April 2013 and then removed. However, if the MFA processes the items on 8 April 2013 or after, it knows that the expiration date for the item is already past and therefore immediately executes the retention action and moves the item into the Recoverable Items folder, where it will remain until the deleted items retention period specified for the database elapses. The default deleted items retention period for newly created databases on an Exchange 2013 server is 60 days. However, this period is calculated based on the item's deletion date rather than its creation date, so it will be held for another 60 days to allow the user to recover the item during this period. (For example, if an item is deleted on 2 April 2013, the MFA will purge it on 1 June 2013.) Some companies use an extended deleted items retention period of up to a year to ensure that users can recover deleted items themselves without involving administrators, a useful practice that costs a little more disk space for databases while releasing administrators from the tedium of having to restore databases to recover a now-important item deleted some months in the past.

Calendar and task items pose some particular difficulties for the MFA when it comes to calculating their expiration date, which is why Microsoft took some time to find the best way for the MFA to proceed and implemented only the processing from Exchange 2010 SP2 RU4. Calendar items are either nonrecurring or recurring and will be either in the Calendar folder or in Deleted Items. It is possible that calendar items might turn up in other folders, but this happens very rarely in practice; if it does, the MFA deals with these items as if they are in the Calendar folder. The following rules are observed:

- Nonrecurring items in the Calendar folder expire according to the end date of the appointment or meeting. For example, if you create a calendar entry for a trip that lasts from 1 June 2013 to 10 June 2013 and the folder tag for the Calendar folder specifies a retention period of two years, the item will expire on 10 June 2015.
- Recurring items in the Calendar folder expire according to the end date of their last occurrence. For example, if a series of meetings is scheduled from 1 May 2013 to 1 September 2013, the item expires on 1 September 2015. If a recurring event has no end date, it will never expire.
- Calendar items in the Deleted Items folder expire based on their message-received date. If this property is not populated (for instance, for an item the user created in his own calendar and never sent to anyone else), the MFA bases its decision on the message-creation date. If neither of these dates is populated, the MFA ignores the item.

Task items receive similar care when the MFA calculates their expiration. The following rules are used:

- Nonrecurring (simple one-off) tasks stored in any folder other than Deleted Items expire based on the message-received date. If this property does not exist, the MFA uses the message-creation date.
- Recurring tasks expire based on the date of their last occurrence. A recurring task with no end date will never expire.
- A regenerating task (a special form of recurring task that generates a specified time after the preceding instance of the task is completed; see Figure 11-21) never expires. Only the most dedicated of users who understand how to make maximum use of tasks create these items!
- Tasks that have been placed in the Deleted Items folder expire based on their message-received date. If this property does not exist, the MFA uses the message-creation date. If neither date is available, the task never expires.

As explained in the previous section, you can use MFCMAPI to validate the dates the MFA uses to calculate item expiration.

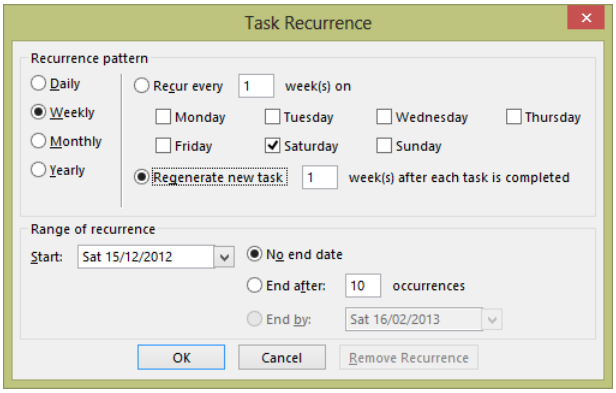


Figure 11-21 Specifying a regenerating task

Preserving information

An important part of being able to achieve a compliance strategy is the ability to preserve information. Retention policies help somewhat because they can automatically move information into archive mailboxes. However, they also remove information through tags that require items to be moved into the Recoverable Items folder or permanently deleted. Items

held in the Recoverable Items folder will eventually be removed from the database and permanently deleted.

Some method is therefore required to limit the work of the MFA as it attempts to keep user mailboxes under control. The mechanism available in Exchange is called *holding*; any MFA processing that might delete an item is restricted, and (sometimes more importantly) users are also prevented from interfering with items.

Exchange 2010 introduced two kinds of holds. A retention hold has nothing to do with the preservation of information except insofar as it is used when an administrator wants to make sure that items are not removed from a mailbox for a temporary period, usually a number of weeks. The classic example of when a retention hold is used is when a user is on vacation for an extended time, and you don't want the user to come back to find his mailbox emptied because the MFA has removed many items in his absence. Usually, people get a chance to rescue items (if they remember) after the MFA has cleaned up their mailbox by using Recover Deleted Items, but if someone is on vacation and probably not thinking about email, he won't be interested in checking his mailbox. Furthermore, if the vacation lasts longer than the deleted items retention period set on the mailbox database (the default is 60 days), items will be permanently deleted while the user is still away. A retention hold solves the problem by instructing the MFA to ignore any delete processing for a specified period. In this case, you might set the retention hold for 65 days to allow the user a few days following his vacation to check the mailbox. Another common example of when a hold might be used is described in *http://technet.microsoft.com/en-us/library/dd297955.aspx*. It occurs when a company is starting to use retention policies and wants to take a cautious approach. By implementing a retention hold on the mailboxes covered by retention policies, it ensures that any items the MFA deletes are retained and can be recovered. When users have become accustomed to the type of automatic mailbox housekeeping enabled through retention policies, the retention holds can be released from mailboxes, and the retained items will be removed.

The second kind of hold used in Exchange 2010 is a litigation hold (sometimes called a legal hold).

You can make the case that a legal hold is similar to a retention hold in some respects in that the MFA is prevented from removing items from a mailbox. However, the concept behind a legal hold is very different because the focus is entirely on the need to preserve information for what could be an extended period (months or even years). Instead of simply making sure that the MFA doesn't remove something important while a user is on vacation, a legal hold ensures that any deletes or edits that occur within the mailbox are preserved while the hold is in effect. This is done to ensure that all possible information that might be required to satisfy a legal discovery action is preserved and remains discoverable, even if a user—by error or deliberately—attempts to remove that information.

When a legal hold is in force, the Store tracks any attempt that the user might make to edit or otherwise alter items in the mailbox and will capture versions silently in such a way that the edits are retained, indexed, and remain available to investigators if an eDiscovery search is performed. These edits are retained in a special location within the Recoverable Items folder. Unless they are informed, users are unaware that items are being preserved.

Placing a mailbox on retention hold can be regarded as a commonplace activity. Wise administrators take advice and guidance from the company's legal department before they place a mailbox on litigation hold to ensure that the action complies with any legal requirements that are in force and does not compromise any document retention policies that are in effect. There's no point in enabling a feature that collects unneeded or unwanted data.

Exchange 2013 supports both retention hold and litigation hold and introduces a more powerful form called an in-place hold, the details of which are discussed in the "In-place holds" section later in this chapter. The older forms of hold are there for backward compatibility with Exchange 2010 servers and because they do provide a specific kind of hold that might be useful in some circumstances. However, the general advice is that you should use in-place hold whenever possible because that is where the focus and development effort is now concentrated.

Putting a mailbox on retention hold

You cannot set retention hold through EAC. To put a mailbox on retention hold, you need to run the Set-Mailbox cmdlet through EMS. For example:

```
Set-Mailbox -Identity 'Tony Redmond' -RetentionHoldEnabled $True
-StartDateForRetentionHold "10/10/2013" -EndDateForRetentionHold "11/11/2013"
-RetentionComment "Mailbox placed on retention hold by A.N. Administrator following
instruction by Legal Department"
```

This command puts the nominated mailbox on retention hold from 10 October 2013 through 11 November 2013. If you want to be more specific, you can include hours and minutes along with the date and provide a value such as 10/10/2013 09:00 to start the hold at 9 A.M. on 10 October 2013. If you don't provide start and end date values for the retention hold, the mailbox is placed on an indefinite hold starting immediately. The retention comment is intended for two purposes: so the administrator can enter a note about why the mailbox was placed on retention hold and to provide the mailbox owner with a notification that the hold is in effect. The text entered for the comment is shown to the user in the backstage area of Outlook 2010 and Outlook 2013, so you should keep this in mind when you compose the comment. The text becomes available to the user after the next time the MFA processes the mailbox.

Another point to remember when you put a mailbox on retention hold is that extra quota is consumed to hold items that would otherwise be removed by the MFA. Given the size of mailbox quotas allocated today, this might not be an issue in your deployment, but it's

wise to check whether a user is close to her quota before you enable the hold and to adjust quotas if necessary.

To remove the retention hold, you run Set-Mailbox again to set the hold parameter to $False. For example:

```
Set-Mailbox -Identity 'Tony Redmond' -RetentionHoldEnabled $False -RetentionComment
$Null
```

Putting a mailbox on litigation hold

When you place a mailbox on litigation hold, Exchange stops removing items from the database when their deleted items retention period expires, and any attempts by the user to delete or change items are retained in the Recoverable Items folder. Items are retained indefinitely until the litigation hold is released, subject to the recoverable items quota not being exceeded. Because items are retained, they remain available to be indexed and can be retrieved by searches. (See the "Searching mailbox contents" section later in this chapter.)

You set litigation hold on a mailbox with EAC by editing the mailbox properties, where the option is available in the Mailbox Features section (Figure 11-22). When you save the new setting, EAC warns you that litigation hold has been replaced by in-place hold. When saved, the new settings become active as soon as the notice that litigation hold has been enabled on the mailbox has replicated throughout the organization.

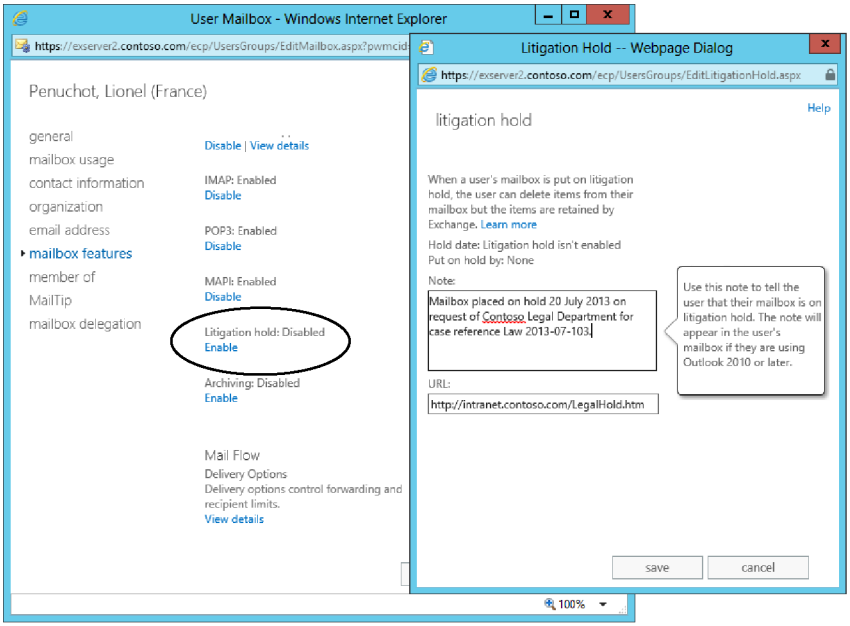


Figure 11-22 Putting a mailbox on litigation hold

Alternatively, you can put a mailbox on litigation hold using the Set-Mailbox cmdlet. For example:

```
Set-Mailbox –Identity 'Ruth, Andy (VP Sales)' –LitigationHoldEnabled $True
-RetentionComment 'Mailbox placed on litigation hold on 16 May 2013' -RetentionURL
'http://intranet.contoso.com/LegalHold.html' –LitigationHoldDate '12/25/2012 09:00'
–LitigationHoldOwner 'Legal Department'
```

EMS issues the same warning and advises that you should use an in-place hold if possible. After running the command, you can examine the updated settings with:

```
Get-Mailbox –Identity 'Ruth, Andy (VP Sales)' | Format-List Retention*, Litigation*
```

The RetentionComment and RetentionURL properties are used to populate the Account Settings section of the Outlook backstage area and inform users that their mailbox has been placed on hold. The –LitigationHoldDate and –LitigationHoldOwner parameters hold the date and time when the hold was enforced and the account that enforced the hold. Exchange completes these details automatically with the current date and time and the primary email address of the user who runs the command to place a mailbox on litigation hold using EAC or EMS.

Litigation hold: What about the user?

Exchange doesn't automatically inform users that their mailbox has been placed on litigation hold and, unless they visit the Outlook backstage area and notice the retention comment, they will be unaware that their mailbox is in a hold status. Indeed, if they don't use Outlook 2010 or Outlook 2013, users might never be aware of this fact. For this reason, you might want to incorporate a step in the hold process to inform whomever authorizes the litigation hold that he is responsible for sending users an email notification to inform them of why the hold is being placed on their mailbox and provide some information about what being on litigation hold means for a mailbox.

Releasing the mailbox from litigation hold is done by reversing the process:

```
Set-Mailbox –Identity 'Akers, Kim' –LitigationHoldEnabled $False –RetentionComment
$Null
```

When a mailbox that is on retention or litigation hold is moved from an Exchange 2010 server to Exchange 2013, the hold remains in place. The same happens when a mailbox moves in the reverse direction. Setting any hold on a mailbox—retention or litigation—could take up to 60 minutes to become effective because the hold is respected after Exchange refreshes the cache it uses to hold Active Directory account information. The exact delay depends on your Active Directory infrastructure and how quickly updated mailbox settings are replicated. Two influences are in play. First, Active Directory must

replicate the updated settings to all global catalog servers before you can be assured that the hold applies across the forest. Second, the Store caches Active Directory data about mailbox properties for performance reasons and therefore will not know that a hold setting has changed for the mailbox until the next time the Store refreshes its cache. The updated hold setting is fetched from Active Directory and becomes effective the next time the Store refreshes its cache. The complete cycle of Active Directory replication and Store cache refreshes could take up to an hour. Therefore, it is a good idea to implement holds, if possible, at a time when users are not actively using their mailboxes.

Searching mailbox content

The content index catalogs that are maintained by Search Foundation for mailbox databases are critical to the ability of Exchange to perform searches. If the catalogs are unhealthy or not fully populated, search results will be unpredictable or incomplete. Administrators don't have to do anything to configure or manage Search Foundation because it is installed automatically on every Mailbox server.

INSIDE OUT **Two versions of content indexes**

Exchange 2010 uses the MSSearch component to build its content indexes for mailbox databases. Exchange 2013 uses Search Foundation. Because the two versions of Exchange use different search technology, you cannot execute an eDiscovery search that encompasses mailboxes on Exchange 2010 and Exchange 2013 servers. Instead, you will have to execute one search for Exchange 2010 and another for Exchange 2013 (using separate discovery search mailboxes) and then combine the search results. For this reason, the recommendation is to move all the mailboxes that are likely to be searched to Exchange 2013 as soon as possible.

Exchange uses the same content indexes for searches by clients, including Outlook Web App and Outlook. However, Outlook uses the Exchange content indexes only when it is configured to work in online mode. Most Outlook clients are now configured in cached Exchange mode, so they use local search indexes created with Windows Desktop Search to conduct searches even when connected to a server.

Items are typically indexed within a few seconds of their creation on a server. The contents of attachments are also indexed, with the caveat that some attachments might fall into the unsearchable category. (See the "What Exchange can search" section later in this chapter.) Indexing throttles back automatically in periods when Mailbox servers experience heavy load. Again, administrators don't have to take any action for this to happen.

Exchange can search across items stored in mailboxes, archive mailboxes, and the Recoverable Items structure (and site mailboxes if this feature is deployed). It cannot search through deleted mailboxes, even if the mailbox content is still in the database because it hasn't exceeded the deleted mailbox retention period. The searching process typically follows these steps:

1. Determine the material that needs to be found and identify the criteria that can be used for a search, such as keywords.
2. Create a mailbox search using the initial criteria.
3. Perform an estimate (a search without retrieval) to assess the effectiveness of the criteria.
4. Refine the criteria to generate search results more accurately; iterate until satisfied.
5. Retrieve information from user mailboxes by using the final search criteria and store it in a discovery mailbox.
6. Have the retrieved information reviewed by investigators.

Underneath the hood, Exchange uses the New-MailboxSearch cmdlet to create new searches and its companions, the Get-MailboxSearch, Start-MailboxSearch, and Stop-MailboxSearch cmdlets, to set search criteria, start or resume a previously defined search, and stop a search. Information is retrieved from the content indexes that Search Foundation maintains. If Exchange captures information from Lync 2013 conversations such as the details of phone conversations (telephone numbers of participants and so on), text for instant messaging conversations, or files that are shared between participants, this information is discoverable through mailbox searches. The same is true if SharePoint 2013 and Exchange 2013 are connected to provide the ability to conduct a single search across mailboxes and SharePoint sites from the SharePoint eDiscovery portal.

INSIDE OUT eDiscovery search policy for exiting employees

If your company commonly needs to conduct eDiscovery searches, it's wise to have a policy of disabling mailboxes for a month or so after someone leaves the company rather than deleting mailboxes immediately. This is especially true for mailboxes that belong to company officers or other executives, which you might need to keep for up to a year after an employee leaves the company. Office 365 attempts to address the problem of how to retain deleted mailboxes for discovery purposes with the Inactive Mailboxes feature. This feature might well surface in a future update for the on-premises version of Exchange.

Search and destroy

As mentioned earlier, EAC executes the New-MailboxSearch command when it needs to search mailboxes. Exchange also supports the Search-Mailbox command. Although EAC doesn't use this cmdlet, you can run it in EMS to perform a search. The big difference between the two cmdlets is that Search-Mailbox can also delete content it finds, which is a useful facility when dubious or unwanted material has arrived on a server and been delivered to several mailboxes. If this happens, you can run Search-Mailbox to locate the unwanted items and remove them in one operation. For example, the following command scans all the mailboxes on a server, looking for items received during a particular period that contain the term "Great Bargain." Any matching items are copied to a discovery mailbox, where they are stored in a folder under the Search Results root. The items are then deleted from the source mailboxes.

```
Get-Mailbox -Server ExServer2 | Search-Mailbox -SearchQuery "Received:
> $('01/01/2013 00:00:00') AND Received: < $('01/31/2013 23:59:59')
AND Great Bargain" -LogLevel Full -TargetMailbox 'Legal Discovery Mailbox'
-TargetFolder 'Search Results' -DeleteContent
```

The Search-Mailbox command can wreak havoc on mailboxes if it is not controlled. Before any attempt is made to remove mailbox content, it is wise to run the command in a log-only mode so that the results are reported and copied to a discovery mailbox without any deletions. When you are satisfied that the search query locates the right content in the right mailboxes, you can run the command again, this time including the DeleteContent parameter, and the content will be removed. Depending whether a hold exists on the target mailboxes, some of the information might be retained. However, this should not be an issue because the offending information is no longer available to users.

In-place holds

The legal hold mechanism available in Exchange 2010 retains mailbox data that might be required for litigation. However, a legal hold operates on an all-in basis that has a side effect of holding everything in the mailbox, including information that is not necessary for legal review. For instance, if you place a mailbox on litigation hold because the mailbox's owner is involved in a patent action about a certain invention, the information the lawyers require is most likely anything to do with the technology covered by the patent and any interaction between the inventor and co-inventors or advisors as the invention evolved from idea to practical implementation. Messages to the inventor's aunt asking her to come to tea on a rainy Saturday in March 2013 are unlikely to add much to the knowledge of the lawyers who seek to defend or attack the patent, but because Exchange retains everything in a mailbox when it is on litigation hold, the possibility exists that the item might be provided to the lawyers for review, something that drives up legal costs without adding anything of value to the discovery process.

The mailbox search mechanism provided in Exchange 2010 is simple to understand and execute. It is capable of uncovering vast amounts of information gathered from user mailboxes on an on-demand basis. However, like many first versions of solutions, these mailbox searches are a blunt instrument.

INSIDE OUT Permissions Required

An account needs to be a member of the RBAC Discovery Management role group to create an in-place hold. Two important roles are present in this role group. The Mailbox Search role enables a user to create a new in-place hold and specify the query that Exchange will use to find matching content. The Litigation Hold role is required to copy matching items from user mailboxes so that investigators can review the data. It is possible to divide the two roles and create a situation in which you have a set of authorized users who are allowed to create in-place holds and another who are allowed to use the in-place holds that were previously created to retrieve and review data from user mailboxes. This creates a separation in function that often exists between those who oversee the search process and those who have to separate the wheat from the chaff in the discovered material.

A more evolved and comprehensive approach is taken in Exchange 2013, with the goal being to satisfy the following requirements:

- Preserve items on hold immutably. That is, neither users nor a computer process can take action to alter the information stored in items that are on hold. To meet this requirement, steps must be taken to ensure that users cannot delete items from mailboxes that are on hold and that the MFA cannot remove items either.
- Provide a method to target items that need to be held rather than having to hold everything in a mailbox. Sometimes it is necessary to hold everything, but more often, investigators are interested in specific information.
- Preserve items indefinitely or for precise periods.
- Allow mailboxes to be governed by different hold conditions arising from multiple investigations.
- Make holds transparent to users; users should not have to alter the way they work just because it is necessary to hold information in their mailbox.
- Make sure that any information that is held is indexed and remains discoverable.

Exchange 2010 satisfies many of these requirements and preserves data immutably. The biggest advance in Exchange 2013 is the introduction of the in-place hold mechanism to enable a more granular form of information preservation and more efficient searches. The idea is that instead of forcing Exchange to retain everything in a mailbox, you can formulate a precise query to identify the information in which the lawyers are really interested, instruct Exchange to hold that information, and ignore everything else. In-place holds make sure that information is retained in mailboxes until it is required and that any deletions users perform are captured so that the information can always be retrieved. In effect, Exchange 2013 supports three distinct types of mailbox searches:

- **Query-based in-place holds** These searches identify a set of criteria that Exchange uses to retain information in the mailbox. Items that meet the hold criteria remain in their normal folders unless an attempt is made to delete them, in which case Exchange retains the items in a special subfolder of the Recoverable Items folder. Microsoft presents the close integration of in-place holds with mailbox searches as a considerable advantage for Exchange 2013. See the "How in-place holds work" section later in this chapter for more information about how Exchange uses holds to retain information.

- **In-place hold without criteria** These searches require Exchange to retain information but don't set specific criteria. This forces Exchange to retain everything in a user mailbox.

- **Regular** These searches do not require Exchange to hold any information in mailboxes. They simply search whatever the mailboxes currently contain, using the criteria provided by the person who creates the search. This is the kind of search Exchange 2010 supports.

Up to five in-place holds can be in effect on a mailbox to handle situations when it is impossible to gather all required material based on one query or when an individual comes under the aegis of multiple legal actions. You can access details of the holds that apply to a mailbox through EAC. The details pane shows the number of holds that apply for the selected mailbox (Figure 11-23), and you can click View Details to have EAC show you the names of the holds. Whenever multiple holds are in effect for a mailbox, Exchange combines the queries by using an OR operator to locate matching items. You can apply more than five in-place holds to a mailbox, but when this happens, Exchange applies a complete hold to the mailbox to reduce the complexity of the combined search terms specified by the various holds. In effect, when you ask Exchange to resolve search term OR search term OR search term OR search term OR search term OR search term OR search term, the resulting set of items would probably be the entire mailbox anyway. The same is true if you put a mailbox in litigation hold alongside in-place holds—all the information in the mailbox is kept.

The information EAC shows about in-place holds comes from the mailbox properties. You can retrieve the same information by using EMS as follows:

```
Get-Mailbox -Identity 'Tony Redmond' | Format-List DisplayName, InPlaceHolds
```

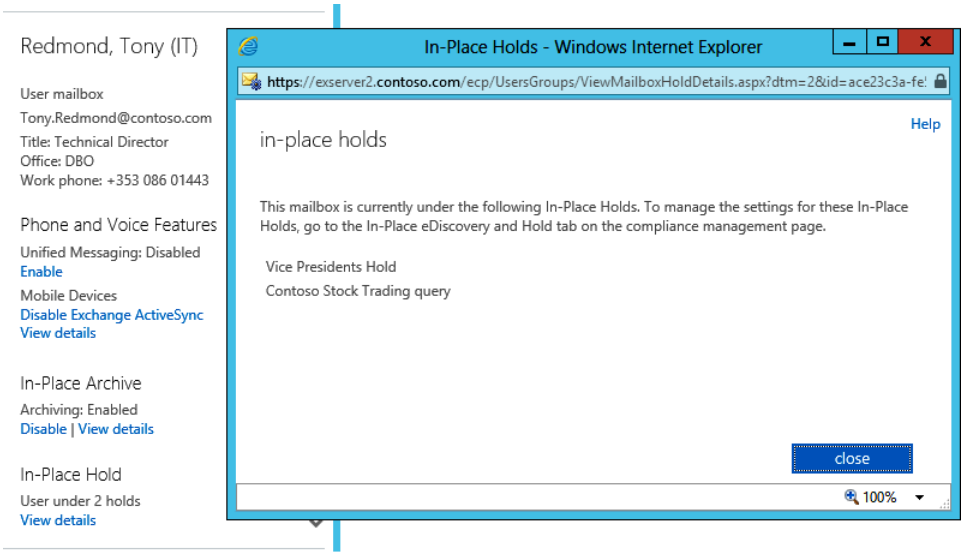


Figure 11-23 Viewing hold details for a user

The information in Figure 11-23 indicates that two holds are in place, each of which is identified by a GUID that ties to the full information about the hold maintained in the Exchange configuration data in Active Directory.

When query-based searches are used, all existing data that meets the hold criteria, plus any new data created during the lifetime of the hold that meets the criteria, are retained. Like retention holds, in-place holds can be time-based and have start and end dates that control how long information is kept. Date-controlled holds are useful because they ensure that data is kept for precise periods and will be deleted thereafter. Using your patent example, you could create a date-controlled hold combined with a query to ensure that any items relating to patent applications are retained. It is also possible to create a date-controlled hold that retains items for a specific number of days based on their creation date. Unlike retention holds, mailboxes are unlikely to run into quota exhaustion because only certain items are retained rather than everything.

It is also possible to emulate the way litigation holds work in Exchange 2010 by creating an in-place hold that has no query parameters or start and end dates to force Exchange to hold all data in the target mailboxes indefinitely or until the in-place hold is removed.

The Exchange 2010 Exchange Control Panel (ECP) offers a Discovery Management section from which administrators can create and execute multimailbox searches. The equivalent place in EAC is the In-Place eDiscovery And Hold option of the Compliance Management section (Figure 11-24), which is where you formulate the queries that underpin in-place holds and execute search options to estimate, preview, and retrieve information specified by queries from user mailboxes.

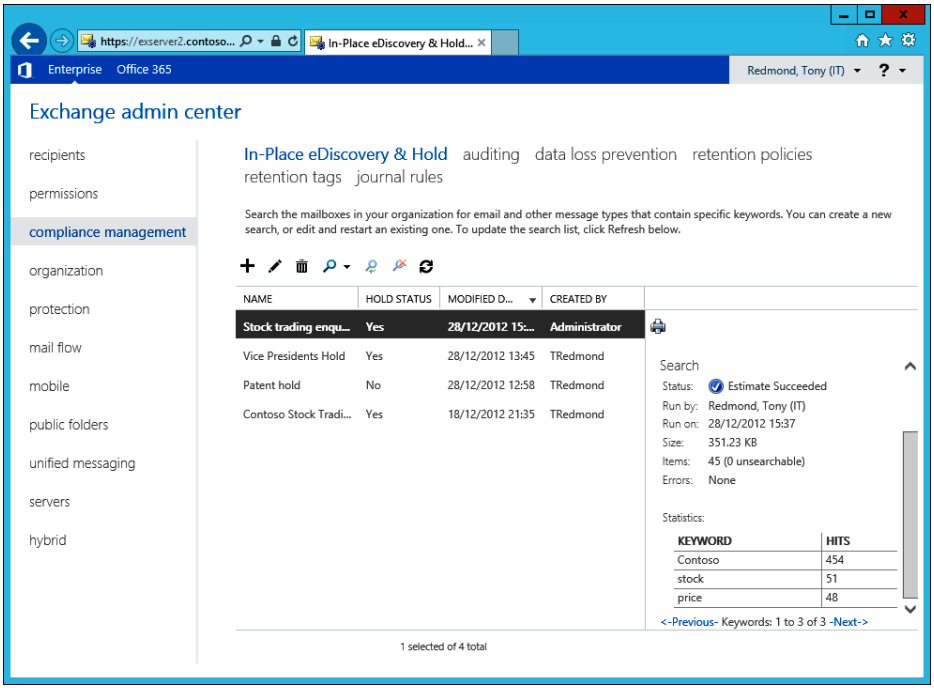


Figure 11-24 The In-Place eDiscovery And Hold section of EAC

Not all the searches shown in Figure 11-24 are based on in-place holds; some searches remain similar to the simpler query-based type Exchange 2010 uses. For instance, if you run the following command in EMS, you can see which searches use in-place holds:

```
Get-MailboxSearch | Format-Table Name, InPlaceHoldEnabled
```

In addition to the InPlaceHoldEnabled property being set to $True if a search uses an in-place hold, the InPlaceHoldPeriod property will be set to either Unlimited (indicating that items should be retained indefinitely) or a certain number of days to govern exactly how long items are retained.

In-place holds work for mailboxes on Exchange 2013 servers only, so if you have some mailboxes on Exchange 2007 or Exchange 2010 servers and need to conduct a search to satisfy a discovery action, you will have to:

- Move the mailboxes to an Exchange 2013 server.
- Arrange to conduct searches on the different server platforms and then collate the results. This is easy enough to do using multimailbox searches for Exchange 2010 mailboxes; you will need a third-party product to do the same for Exchange 2007 servers.

From the preceding, it's obvious that the best idea is to move any mailbox that might be involved in a discovery action to Exchange 2010 or Exchange 2013 (preferably) as quickly as possible. Remember that deleted items are not moved when mailboxes migrate from Exchange 2007 servers because this version of Exchange does not use the enhanced Recoverable Items structure found in Exchange 2010 and Exchange 2013.

It is not a good idea to move mailboxes from Exchange 2013 to earlier versions of Exchange if they contain information that might be of interest to searches. As discussed earlier, retention and litigation holds remain in place only if the mailbox moves to an Exchange 2010 server. In-place holds are a new feature of Exchange 2013. Mailboxes that move to Exchange 2010 or Exchange 2007 disappear from searches that use this feature.

Creating a new search

Assuming that you know what information needs to be retained and keywords or other identifying phrases that can be used to locate matching data in user mailboxes, you can create a new search. From the In-Place eDiscovery And Hold section of EAC (shown in Figure 11-24), click New (+) to begin defining the search over four screens:

1. Give the search a name; provide some information about why the search is necessary and what it hopes to find (Figure 11-25). Although Exchange won't care what name you give (such as the "Great XYZ search for all things"), it's better to use a name that makes sense to anyone involved in the search, administrators and legal staff alike. Ideally, the descriptive text entered for a search should let anyone who accesses the search understand its purpose. It's a good idea to include information such as who initiated and authorized the search, the expected duration of the search period, and other pertinent details.

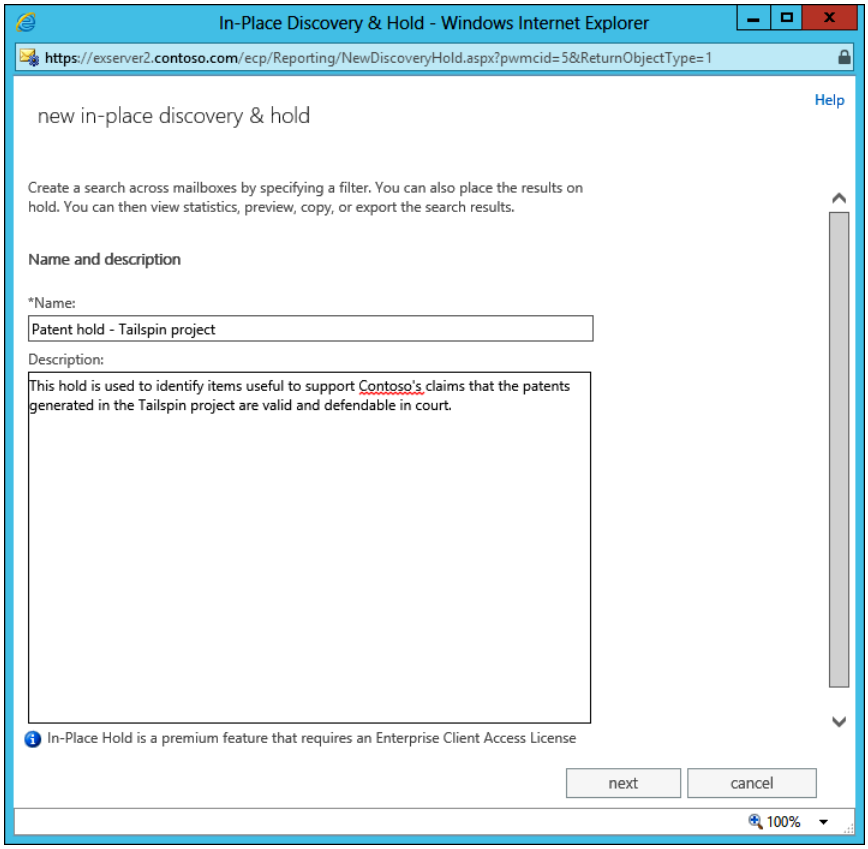


Figure 11-25 Naming a new search

2. Define the mailboxes the search covers. You can select All Mailboxes or specify a list of mailboxes, including groups. It's always better to restrict the number of mailboxes as much as possible to help Exchange execute the search and retrieve search results. Usually, the number of mailboxes covered by a search is dictated by the legal team to meet the demands of a discovery action, and the preference of system administrators to restrict the number of mailboxes might be ignored. As shown on the screen on the left in Figure 11-26, adding hundreds or thousands of mailboxes through this screen would be time consuming, so it's a good idea to use groups whenever possible. There are some points to consider when using groups that are discussed in the "Using groups with searches" section later in this chapter. If you select All Mailboxes for a search, you cannot specify that an in-place hold should occur because this action could cause a huge amount of information to be retained in a large Exchange organization.

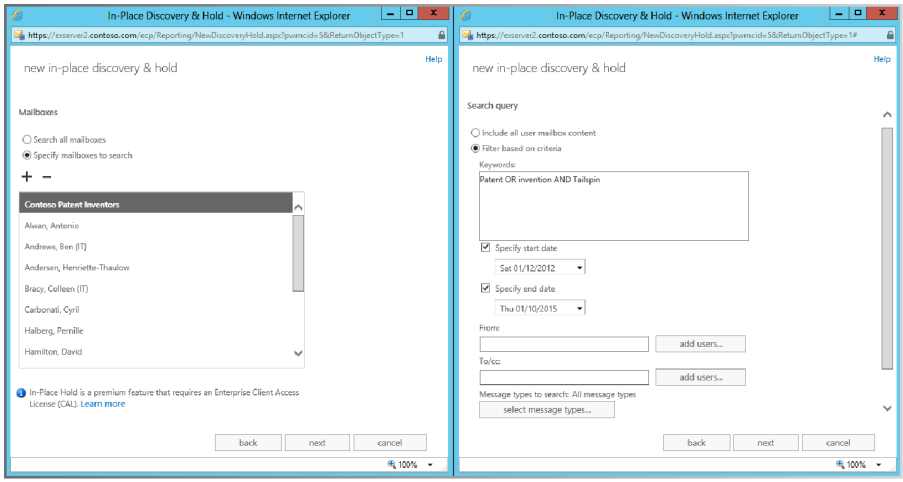


Figure 11-26 Specifying source mailboxes and the query for a search

3. Specify the search query that locates matching items in user mailboxes. In the screen on the right in Figure 11-26, you can elect to search for all content or filter based on a query ranging from very simple (as in this case) to quite complex, depending on the needs of the search. See the "Search syntaxes" section later in this chapter for more information on the keyword query language (KQL) search syntax that Exchange 2013 uses. In addition to the basic query, you can refine it by specifying:

   a. Start date and end date.

   b. The email addresses of users who sent messages.

   c. The addresses of people who were TO or CC recipients for messages.

   d. The exact types of items stored in mailboxes that are of interest to the search. For instance, you might only want to search for contents held in documents stored in mailboxes, such as attachments sent around for messages.

4. The last stage is to decide whether to include an in-place hold for the search. This requires an enterprise CAL for every mailbox covered by the hold. Everything else to do with the search up to this point is covered by the standard CAL. When you set an in-place hold (Figure 11-27), Exchange monitors items in user mailboxes covered by the search for items that match the query and takes steps to ensure that no matching items are removed from the mailbox. This can be done indefinitely or for a specified period. In this case, you want items to be retained for approximately six years, or 2,192 days.

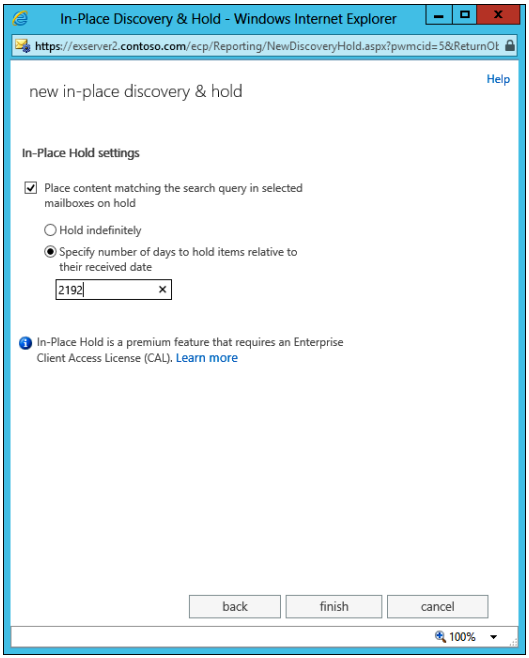


Figure 11-27 Deciding whether items should be placed on hold

When you've completed all the screens, click Finish to have EAC save the new search. During the save process, EAC validates that the query you entered on the third screen is valid. It's easy to enter search terms by using incorrect syntax until you become accustomed to KQL syntax; when you use incorrect syntax, you see the error shown in Figure 11-28. If this happens, close the error screen and go back to the screen where you entered the query and adjust its syntax. Eventually, perhaps after some additional trial and error, EAC can save the search and perform an initial estimate of the items that can be uncovered when the search is executed.

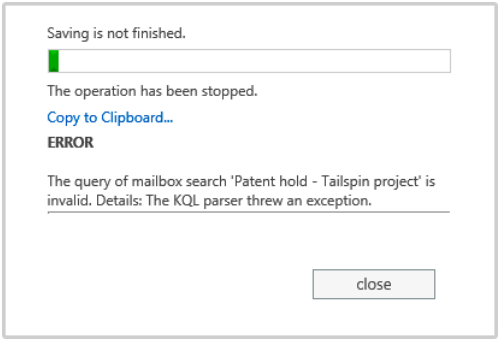


Figure 11-28 Detecting a problem with search syntax

Refining a search

After you enter all the search criteria and click Save, Exchange stores the criteria as search metadata in the Discovery folder (Figure 11-29) of the search arbitration mailbox (*SystemMailbox{e0dc1c29-89c3-4034-b678-e6c29d823ed9}*). Exchange then generates an initial estimate of results based on the query specified in the search. The idea is that you immediately see how effective the search criteria are in terms of identifying information across the set of selected mailboxes. A good search locates the right information and only that information. Conversely, a bad search casts its net far too broadly and finds information that is not required for discovery. Remember that someone eventually must go through all the items recovered by a search and that this process grows increasingly expensive as the number of items to be reviewed grows. In addition, although computers accept heavy workloads, setting a task that will retrieve tens of thousands of items from Mailbox servers across the organization and result in several gigabytes of data transferred to the target discovery mailbox is not something that you really want to do without thinking. It is much better to have a focused search that delivers exactly the right information in the right quantity, which is the goal that search estimates and reviews help you achieve.

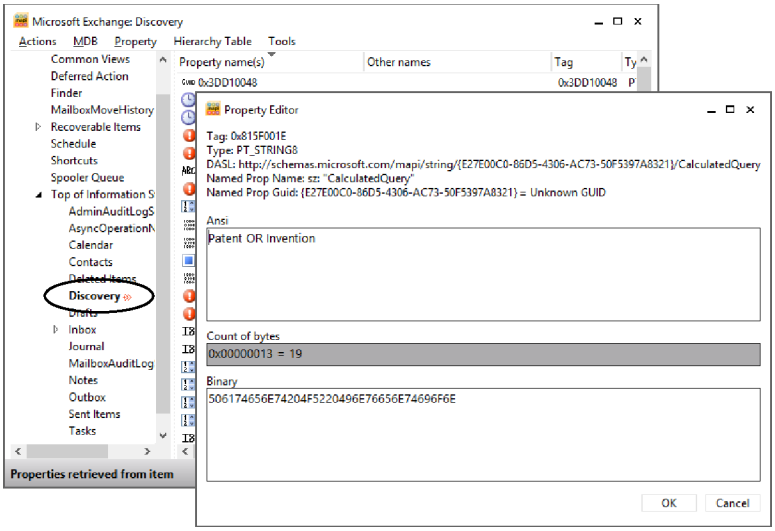


Figure 11-29 Where search metadata is held

To return to the current search, the metadata that describe the search parameters to Exchange can be fetched and examined by running the Get-MailboxSearch cmdlet. For example:

```
Get-MailboxSearch -Identity 'Patent Hold - Tailspin Project' | Format-List
```

Among the interesting properties that can be found in the output are:

- **SourceMailboxes and Source** These properties list the source mailboxes for the search.
- **TargetMailbox** Eventually, when you have refined the search and are ready to copy items, this property holds the name of the discovery mailbox in which Exchange will store the items retrieved from user mailboxes.
- **SearchQuery** This is the query used for the search in KQL syntax.
- **Senders and Recipients** If the search criteria include checking for items sent by or received by specific users, their email addresses are listed in these properties.
- **MessageTypes** If blank, this property means that all types of items held in the source mailboxes should be searched. Otherwise, it contains the exact types to be searched.
- **SearchDumpster** For a search to be complete, it should search the Recoverable Items folder to check whether any deleted items match the search criteria, so this flag is usually set to $True. You can avoid searching the Recoverable Items folder by setting the flag to $False, but this has to be done through EMS because searches initiated from EAC always include Recoverable Items.
- **IncludeUnsearchableItems** This is usually set to $False, meaning that the initial search performed to generate an estimate ignores any items Search Foundation is unable to index such as S/MIME protected messages.
- **IncludeKeywordStatistics** This is usually set to $True, meaning that Exchange should return keyword statistics for the search. The TotalKeywords property contains the number of keywords by the search.
- **ExcludeDuplicateMessages** This is usually set to $False when generating an estimate to understand exactly how many items might be found. When you are ready to retrieve items, it's possible to set this flag to $True so that Exchange de-duplicates found items.
- **Status** After a search is created and a first estimate is made, the value should be Estimate Succeeded, indicating that Exchange ran the search successfully and generated an estimate of the items in the source mailboxes that meet the search criteria. You also see a numeric value for the total number of source mailboxes (NumberMailboxesToSearch) and another numeric value for the total number of items found (ResultNumberEstimate). The exact number won't be known until you retrieve items and store them in the search mailbox, but the estimate is usually accurate.

In addition, you see a value returned as an estimate of the size of the found items (ResultSizeEstimate).

- **PreviewResultsLink** This property contains a URL to enable the investigator to view the search results by presenting an interface similar to Outlook Web App to the items in the source mailboxes. Reviewing items in this manner is an excellent way for an investigator to determine whether the search is successful or needs to be refined.
- **InPlaceHoldEnabled** If set to $True, an in-place hold is present. The InPlaceHoldIdentity property provides a link to the hold that is written into the source mailboxes to provide a connection to the search.

Equipped with knowledge about the initial estimate for a search, you can think about how the search might be refined. Perhaps the criteria expressed in the query are not quite good enough and need another keyword, or perhaps the association between the keywords needs to be tweaked in some way. Maybe you missed some mailboxes and need to add them to the source list. And what about unsearchable items? Are they likely to be a problem? (See the "What Exchange can search" section later in this chapter.)

In most cases, an investigator refines a search by trying different combinations of keywords until he is satisfied that the search will uncover a reasonable (or expected) volume of information. After each change is made to the search parameters, you can use Estimate Search Results (Figure 11-30) to assess how effective the search is. When an estimate is requested, Exchange queues the search for processing. Depending on the size of the organization and the scope of the search, the new estimate might be available in a matter of minutes or take a little longer. Eventually, the search estimate completes, and its metadata is updated so EAC can display new information about keyword statistics (visible in the details pane shown in the bottom right of Figure 11-30).

When the investigator is satisfied that the search query is generating results in the general area of the desired set, he can use human intelligence to validate the effectiveness of the query by selecting Preview Search Results. From the discussion about search properties, you know that the metadata stores a URL that generates an interface similar to Outlook Web App to display the items the search identified. The URL will look something like this:

https://exserver2.contoso.com/owa/default.aspx?cmd=contents&module=discovery&discoveryid=Patent%20hold%20-%20Tailspin%20project.

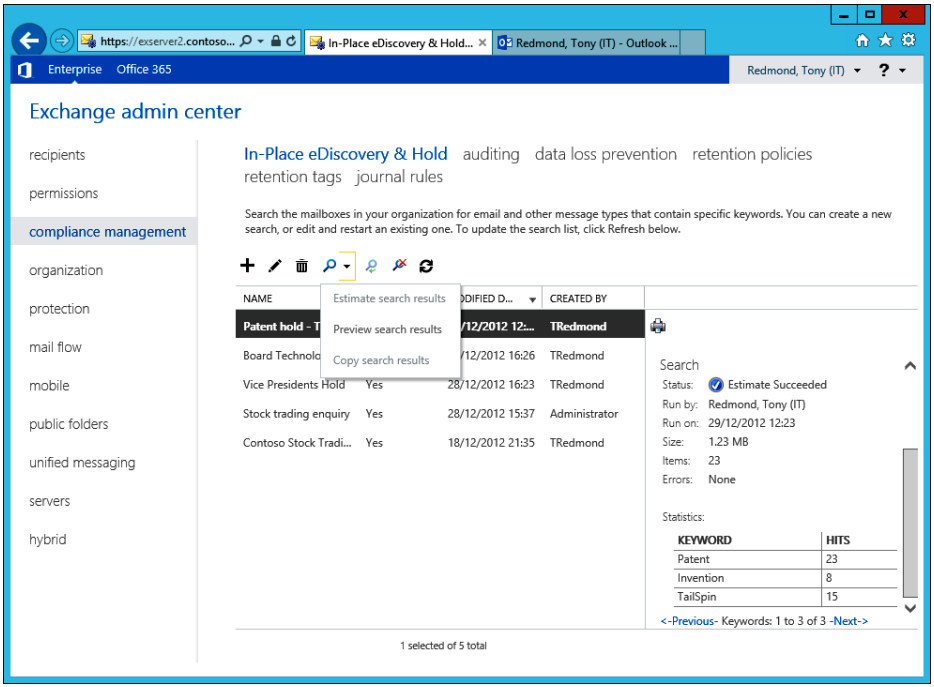


Figure 11-30 Options available in EAC to continue searching

Clicking Preview Search Estimates displays a new tab in the browser, shown in Figure 11-31. The intention behind previewing is simply enabling an investigator to see the actual content of items identified by the search rather than allowing her full Outlook Web App functionality. You cannot, for instance, find an item of interest and email it to another user when previewing search results. Cut and paste is also disabled in the viewing pane, so you cannot use that route to save information and include it in a message, although nothing stops you from taking a screen shot. An investigator should be able to tell from the items turned up by a search whether any further refinement is necessary. Either too much spurious information is being uncovered or too little information of interest has been found. In either situation, the preview should give an extremely good hint about what needs to be done to improve the search to a point at which the investigator can begin extracting items from user mailboxes and copying them to a discovery mailbox.

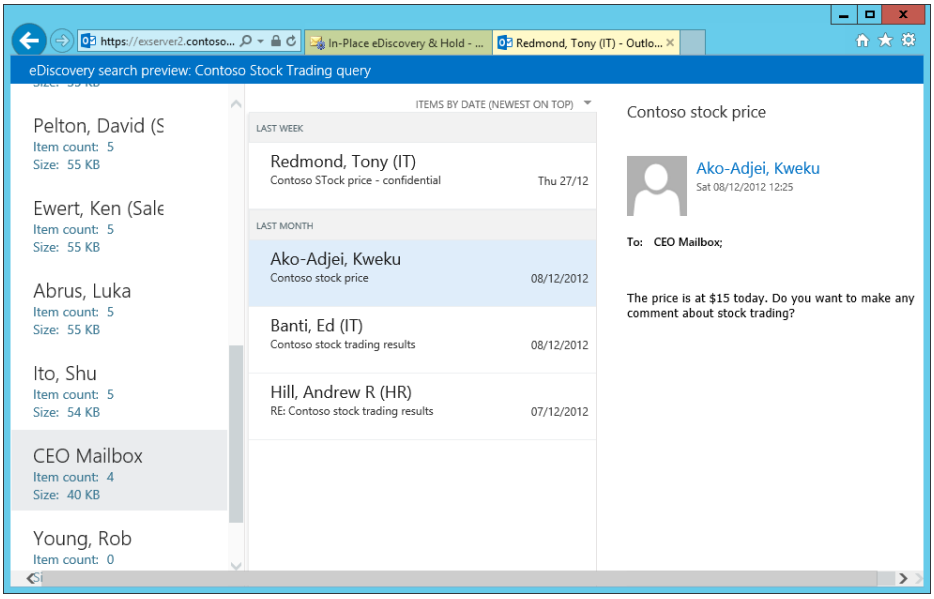


Figure 11-31 Previewing search results

All searches that EAC launches automatically examine deleted items held in the Recoverable Items folder. (The SearchDumpster flag is $True.) If you want, you can exclude deleted items from the search by updating the flag to $False. However, Recoverable Items is included to ensure that any items of interest are captured even if a user has attempted to remove all traces of their existence. If an item is found in Recoverable Items, it is shown in the Recoverable Items folder within the user's primary mailbox or archive when the items are reviewed in the discovery mailbox.

INSIDE OUT **No need for start or end dates**

You don't need to pass a start or end date for an eDiscovery search. Although most searches are date-based, you don't need to use any dates; Exchange simply assumes that you want to search from the start of time right up to the current date and time. It's as simple as that!

Retrieving discovered content

When investigators confirm that a search is correctly focused on the items that are necessary, they can proceed to the final stage in the process: copying items from user mailboxes

into a discovery mailbox where the items can be reviewed and dealt with as necessary. When you select Copy Search Results, EAC displays the screen shown in Figure 11-32.

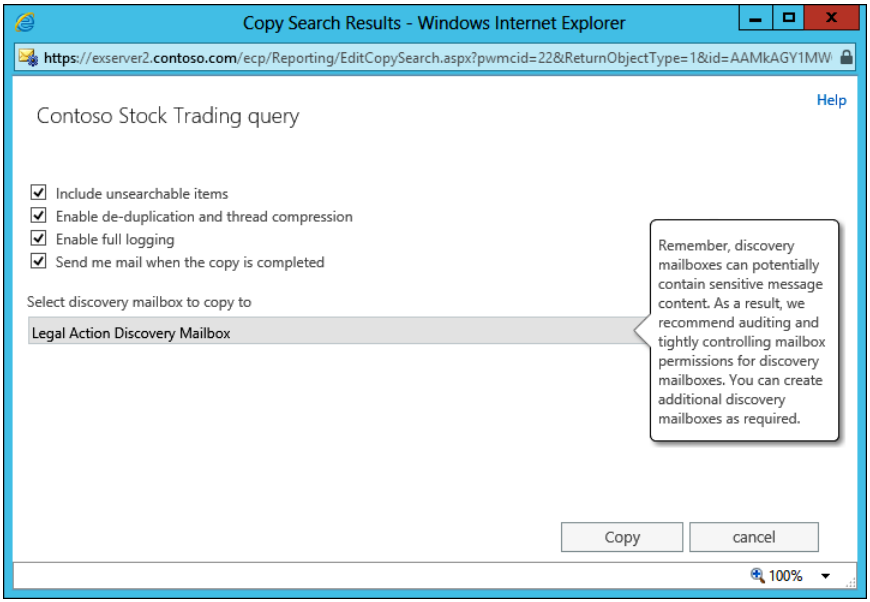


Figure 11-32 Setting parameters to copy items found by a search

The options available are:

- **Include Unsearchable Items** Although a query used with an in-place hold cannot be guaranteed to find matching content in unsearchable items, search and copy can locate matching unsearchable items based on item properties such as Subject. Therefore, you can include unsearchable items when instructing search to copy items from user mailboxes. To hold everything in a mailbox, including any unsearchable items, apply a litigation hold or an in-place hold that doesn't have an associated query.

- **Enable De-duplication** It is the nature of email systems to have identical items in user mailboxes. For example, a message you send to 14 recipients starts by being discoverable in 15 mailboxes. Gradually, the number of discoverable copies reduces as recipients delete their copy, but a search inevitably turns up multiple copies of items. De-duplication causes the search to retrieve a single copy (the first copy encountered); you depend on the recipient list for the item to understand who actually received it. Apart from the system overhead incurred to copy and store every instance of a message found in the searched mailboxes, providing extra copies of messages drives up the cost of responding to legal discovery actions if the lawyers or other individuals who review the search results are paid on a per-item

basis. De-duplication might or might not be acceptable to your legal team. If de-duplication is not used, a search copies every copy of matching items it finds, no matter the number of mailboxes in which the items are located. Exchange 2013 does not support thread compression at this point.

- **Enable Full Logging** The default is to use basic logging when copying items. Full logging generates comprehensive details of what's been done to fetch items from user mailboxes.
- **Send Me Email When The Copy Is Completed** Copying items located by a search can take an awfully long time (defined as being longer than a coffee break), especially if gigabytes of information have to be retrieved from remote Mailbox servers. Asking Exchange to send you an update to let you know when everything is done is simple good sense and can enable you to do more valuable work in the interim.

Note

In terms of de-duplication, you should remember that because Exchange does not expand the membership of a group into a message header, seeing that a message was delivered to a group doesn't tell you whether someone received a copy because there's no way of proving what the membership of the group was at the point when the transport service expanded the group membership and delivered the message. For more information on journaling, see Chapter 2, "The Exchange transport system," in *Exchange Server 2013 Inside Out: Connectivity, Clients, and UM* by Paul Robichaux (Microsoft Press, 2013).

When you click Copy, EAC updates the search metadata and launches the search and copy operation. Items are copied in the background and, eventually, you receive an email similar to that shown in Figure 11-33 informing you that the search and copy operation has completed.

After notification that the search has finished copying items, you can return to EAC. At this point, the properties of the search shown in the details pane should reveal the number of items the search found. It also has a link you can click to access the discovery mailbox storing the copied items (Figure 11-34). The link invokes a new Outlook Web App session connected to the discovery mailbox, assuming the user who ran the search also possesses full access permission for the discovery mailbox.

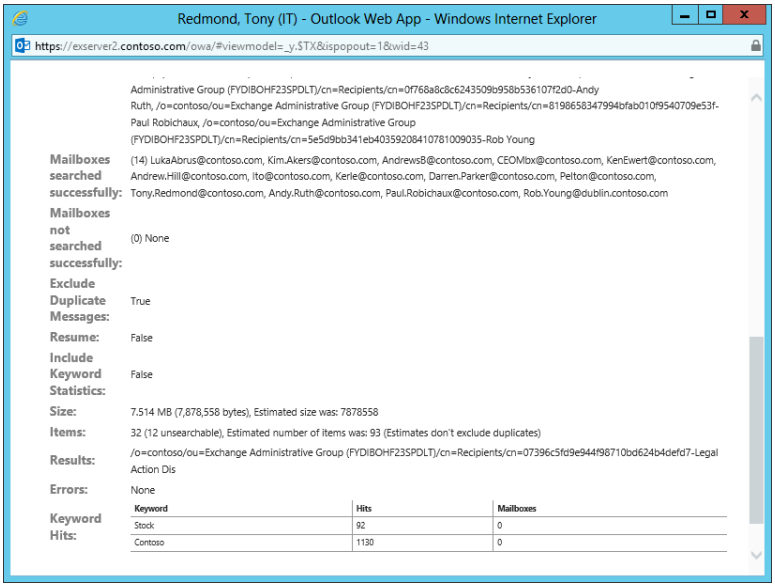


Figure 11-33 Email stating that the search and copy operation completed

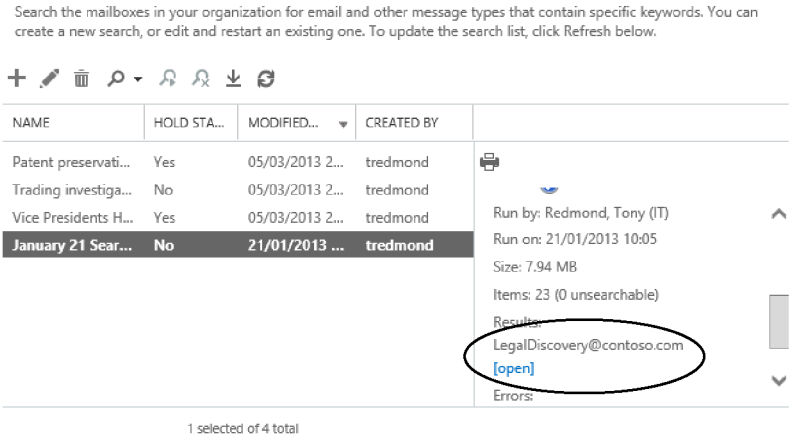


Figure 11-34 Link to access discovery mailbox

INSIDE OUT **Dealing with hybrid deployments**

Searches conducted in hybrid deployments are quite likely to include some on-premises and some Office 365 mailboxes in the scope of their query. In this situation, searches are conducted from EAC running in the on-premises organization and using an on-premises discovery mailbox to store the copied results. Any messages found in Office 365 mailboxes are copied across the Internet, using the connections linking the on-premises organization with the Office 365 tenant.

Examining search results

By default, an Exchange organization has a single discovery mailbox. You have to select a discovery mailbox to use when you copy items for a search. The default discovery mailbox serves well for most purposes. However, as described in Chapter 5, you can create additional discovery mailboxes to hold search results. You might want to segregate the items belonging to one search clearly from those belonging to another, spread the load required to copy items as they are fetched from databases around the organization, or isolate items retrieved from mailboxes belonging to users in a certain country to comply with privacy regulations.

The results of the search, including copies of all items that match the search criteria, are placed in the selected discovery mailbox. If you copied items for the same search to the discovery mailbox previously, Exchange removes the items for the old search before it copies items for the new search.

Within the discovery mailbox, Exchange organizes search items differently, depending on the kind of search you perform. For searches that retrieve all items (de-duplication not performed), items are organized into a set of folders under a root folder called the name you gave to the search. For example, if you call the search "Illegal stock trading investigation," Exchange creates a root folder of this name in the discovery mailbox and then creates a child folder underneath for each mailbox in which a matching item was found (Figure 11-35). The date and time of the search (the date and time of the server rather than the client workstation that starts the search) is appended to the mailbox name to clearly identify different searches that have occurred and provide a solid time line for when evidence is gathered for an investigation. If you open the folder for a mailbox, you see all the folders from which items have been copied in both the primary mailbox and the archive (if the mailbox has one). You can then click the items to review their content and decide whether they are of real interest to your investigation. Incriminating evidence can be retained and any useless information discarded.

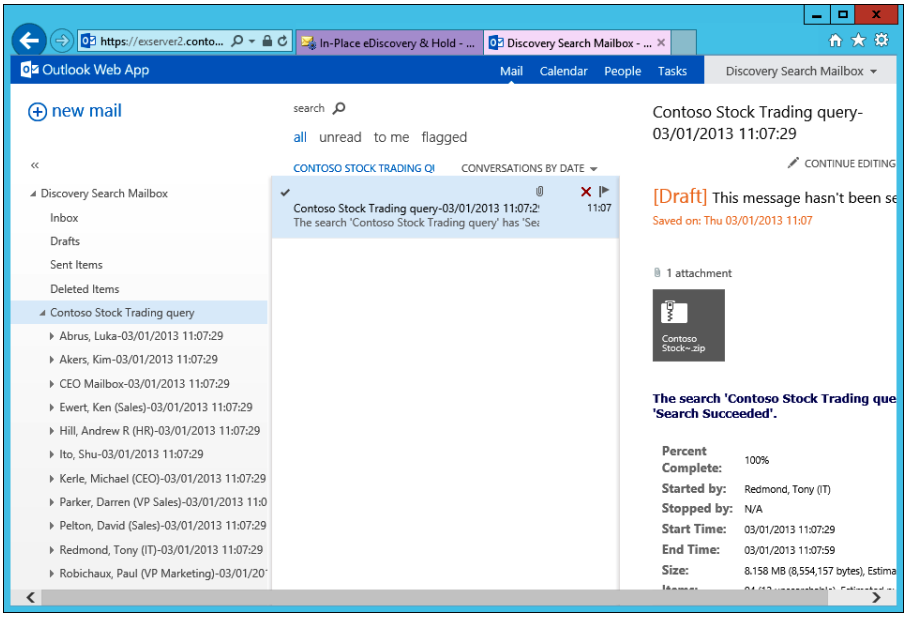


Figure 11-35 Search items placed in a discovery mailbox

In Figure 11-35, you can also see a message in the Inbox of the discovery mailbox that contains details of the search performed and an attachment containing the log file for the search. The log file contains information about all the items the search found, including the folder in which the items are located in the source mailboxes.

Items retrieved for de-duplicated searches are stored in a different form. As you can see from Figure 11-36, instead of a folder structure composed of a separate folder for each mailbox that is searched, Exchange creates a single folder named after the search together with the date and time the items were copied, and a single copy of each item that's found by the search is stored in the folder. The message identifier, which is a unique value established when items are first created, is used as the basis of de-duplication. As noted earlier, investigators have to verify that they can discover who received an item located by the search by opening the item and examining its properties. Far fewer items are copied for a de-duplicated search, so it's a good idea to use this kind of search as the starting point for an investigation and move to a full copy only if absolutely required.

Note the presence of the Unsearchable subfolder. This contains all the items Exchange considers unsearchable for some reason (perhaps because of an attachment Search Foundation could not index). The items have been located by the search because some elements such as the message properties have been indexed. An investigator must open and examine each of the items in the Unsearchable folder to determine whether it meets the search criteria.

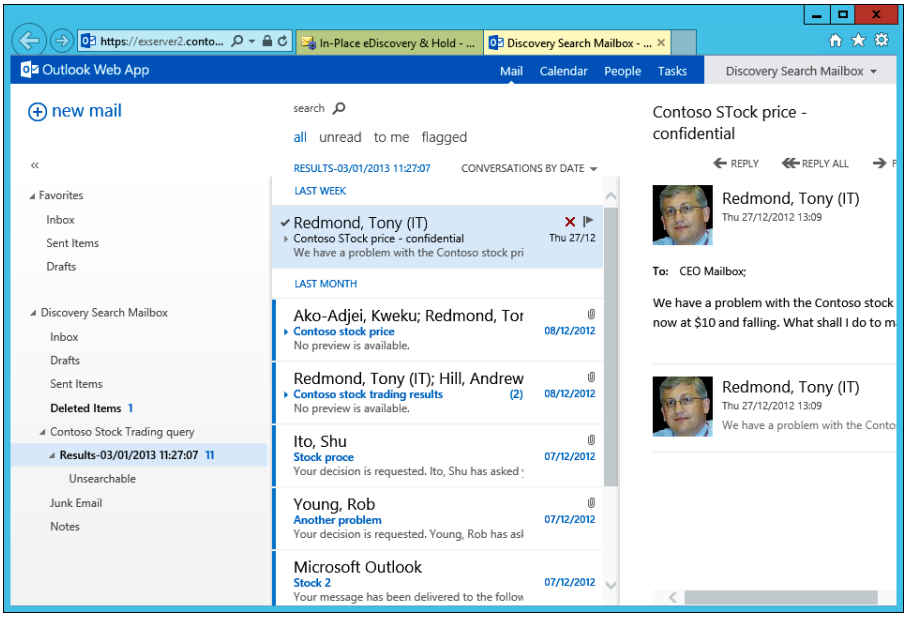


Figure 11-36 Items retrieved for a de-duplicated search

Outlook Web App in Exchange 2010 supports an annotation option an investigator can use to mark an item for follow-up. This option is not available in Exchange 2013.

Controlling access to discovery mailboxes

The users who perform eDiscovery searches are not necessarily those who can access the results of the searches that are placed in discovery mailboxes. You need to assign full access permission to the discovery mailbox to a user before he can open it to access the search results. By default, members of the Discovery Management role group should be able to access the default discovery mailbox, but you have to grant full access explicitly to any other discovery mailboxes you create for use in mailbox searches.

A clear separation therefore exists between the following:

- Membership of the Discovery Management role group, which is required to be able to create and execute mailbox searches.
- Full access to the discovery mailbox used for a mailbox search, which is required to open the discovery mailbox and review the items copied there by the mailbox search.

The separation between the two requirements enables a division of responsibilities between those who are responsible for responding to requests for information (often the IT department) and those who will review the retrieved information forensically to look for evidence or other information that is important to an investigation (often the legal department). You might therefore create discovery mailboxes to hold information retrieved for different types of searches so that you can restrict access to those mailboxes to ensure that confidential material is always treated in a correct and legally defensible manner. Some discovery mailboxes might be used for straightforward legal discovery actions and be under the control of the legal department, whereas others might be used for the pursuit of internal complaints against an employee for offenses such as sexual harassment and be restricted to selected members of the HR department.

CAUTION!

Access to content held in discovery mailboxes should be carefully controlled so that only the people who need to review and work with the data have access. You must also be sure that the users do not interfere with the search results in an unauthorized manner. For example, it would not be a good situation if someone attempted to cover up illegal activities by appearing to conduct a search for suspicious items and then deleted a selected group of the discovered items to remove evidence. To address this situation, you can enable auditing for discovery mailboxes to force Exchange to capture information about the actions these users take when they work with items. More information about how to set up mailbox auditing is available in the "Auditing mailbox access" section later in this chapter.

Exporting discovered content

After the members of the investigation team have settled on the content that is relevant to a discovery action, they might have to export the information so that it can be provided to a third party such as external legal advisors. Exporting data from the discovery mailbox to a PST is the usual approach in these cases. This can be done in two ways:

- Select the search from EAC and click the Export icon.
- Run the New-MailboxExportRequest command.

The EAC Export To PST option exports the complete contents of a mailbox without applying a filter. You might not want to export everything that has been uncovered by a search from the discovery mailbox, so EMS is often the better solution because you can build a mailbox export request based on whatever filter is required and use it to export the data.

For example, this command uses a date filter to export items from the default discovery mailbox to a PST:

```
New-MailboxExportRequest -ContentFilter {(Received -ge "01/01/2012") -and (Received
-lt "01/01/2013") -or (Sent -ge "01/01/2012") -and (Sent -lt "01/01/2013")} -Mailbox
"DiscoverySearchMailbox {D919BA05-46A6-415f-80AD-7E09334BB852}" -Name ExportContoso
-FilePath \\ExServer2\PST\ExportContosoAction.pst -Baditemlimit 10
```

For more information about using the New-MailboxExportRequest cmdlet, see the "Exporting mailbox data" section in Chapter 10.

Resource throttling for searches

If you add more than 50 mailboxes to a search, you encounter a slight problem in that Exchange cannot display keyword statistics for the search simply because the search is likely to impose a large overhead on servers to gather the keywords when information from more than 50 mailboxes is reviewed. The error message EAC displays (Figure 11-37) is a little misleading because you might assume that Exchange is limited in terms of the number of mailboxes that search supports. You can still perform an estimate of a search, review search results, retrieve information, and store it in a search mailbox, even if you lack one of the very useful tools available in Exchange to assess search effectiveness.

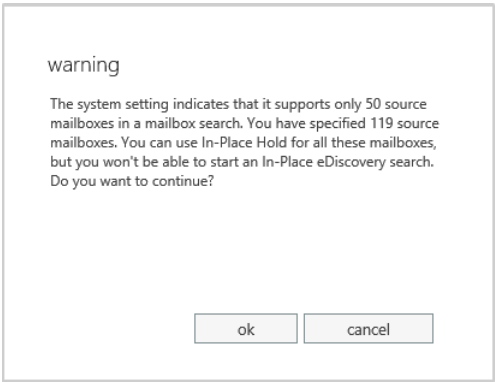


Figure 11-37 Error displayed when more than 50 mailboxes are added to a search

The resources any Exchange process can consume are constrained by throttling policies. You can amend the policy to increase the number of resources available to searches. First, to see the categories of search resources the default throttling policy governs, run the Get-ThrottlingPolicy command:

```
Get-ThrottlingPolicy | Format-List *Discovery*
```

The following list describes the meanings of the settings that govern searching:

DiscoveryMaxConcurrency The maximum number of searches a single user can perform concurrently

DiscoveryMaxMailboxes The maximum number of mailboxes that can be searched for estimate purposes (for which you can see statistics for keywords)

DiscoveryMaxKeywords The maximum number of keywords that can be specified in the query used for an eDiscovery search

DiscoveryMaxMailboxResultsOnly The maximum number of mailboxes from which a search can copy results to a discovery mailbox

DiscoveryPreviewSearchResultsPageSize The maximum number of results that can be shown on a single page in a search preview

DiscoveryMaxKeywordsPerPage The maximum number of keywords that EAC can display on a page when viewing search statistics

The issue that you run into with mailboxes is controlled by the DiscoveryMaxMailboxes setting. By increasing this value from 50 to 120 or more (the current search covers 119 mailboxes), you won't see the error condition afterward. However, the default global throttling policy provided with Exchange 2013, which applies to all users unless a specific throttling policy has been assigned to them, is read-only. Therefore, you must create a new throttling policy and set increased values in it and then assign that policy to the users who will run eDiscovery searches from EAC or EMS. Any value that is not set will be inherited from the global throttling policy.

To create a new throttling policy, run the New-ThrottlingPolicy command and set the scope for the new policy to Regular so it can be applied to user mailboxes:

```
New-ThrottlingPolicy –Name 'Discovery Throttling Policy' –ThrottlingPolicyScope
Regular
```

Next, update whatever settings you need to increase:

```
Set-ThrottlingPolicy –Identity 'Discovery Throttling Policy' –DiscoveryMaxMailboxes
150 –DiscoveryMaxKeyWords 1000
```

Finally, assign the new policy to the users who need to run eDiscovery searches. Typically, these are the members of the Discovery Management role group.

```
Set-Mailbox –Identity 'Tony Redmond' –ThrottlingPolicy 'Discovery Throttling Policy'
```

Increasing the values for resources assigned through throttling policies should be done only when absolutely required. You do not want to order an eDiscovery search that absorbs so many system resources on a server that it cannot adequately respond to user requests.

INSIDE OUT **Memory limitations and mailbox searches**

eDiscovery searches make considerable resource demands on Mailbox servers to search for and retrieve items, so Exchange 2010 limited the number of mailboxes that can be searched in a single operation to 25,000 unless you increase this number by updating the HKLM\SOFTWARE\Microsoft\ExchangeServer\v14\Discover *DWORD* value in the system registry. The value should be set to the number of mailboxes you want to search. Make sure that the value is set to the same number on every Mailbox server in the organization. Exchange 2013 takes a different approach; it uses throttling policies instead of registry entries, which eases the problem of making sure that every Mailbox server is configured in the same way. The value for the maximum number of mailboxes for a search is reduced to 5,000 in the default throttling policy. This reflects the experience gained with Exchange 2010, when the majority of searches scanned fewer than 5,000 mailboxes. You can increase this value, but do so only when you're sure that such a large search is absolutely necessary.

How in-place holds work

In-place holds work on the basis that messages are evaluated by the Email Lifecycle Assistant (ELC), a component of the Managed Folder Assistant (MFA), against hold criteria when they are considered for deletion from a database. If an item satisfies hold criteria, ELC moves it into the DiscoveryHolds subfolder in the Recoverable Items folder. Users cannot see items held in this folder by using normal clients such as Outlook. It is possible to view the items by using a raw protocol client such as MFCMAPI or the EWS Editor. However, although you can export a copy of an item, you can't remove items from the DiscoveryHolds folder by using these utilities.

As noted earlier, an administrator can assign up to five hold queries to a mailbox. If a sixth hold query is added, the entire mailbox is placed on hold because it is more efficient to retain everything rather than attempting to resolve multiple queries.

The life cycle of items subject to a query hold is as follows:

- Normal processing occurs for the item. It is delivered or created, dealt with by the user, and eventually deleted and moved into the Deleted Items folder. The user might delete the item, or the MFA might move it to comply with a retention policy.
- Eventually, the Deleted Items folder is emptied, and the items held there are moved into the Recoverable Items folder. Because the item is still available to the user and accessible by an eDiscovery search, there is no need to consider it against any query

hold criteria that might be set on the mailbox. As normal, the user can restore an item from the Recoverable Items folder and move it back into a regular mailbox folder.

- The MFA runs on a work cycle basis with the goal of processing every mailbox on a server at least once a week. As part of its regular processing, the MFA assesses items in the Recoverable Items folder to determine whether their retention period has expired. Because an in-place hold has been assigned to the mailbox that requires items to be retained for another period (or indefinitely), the MFA moves expired items that would otherwise be permanently removed from the database into the DiscoveryHolds folder. The same processing occurs if a user deletes an item from the Recoverable Items folder.

- ELC evaluates the items in the DiscoveryHolds folder to determine whether they are needed for the in-place hold. If the item satisfies the criteria, the item is retained. If not, it is purged from the database as normal.

- ELC continues to evaluate items in the DiscoveryHolds folder every time the MFA processes the mailbox. Items are retained for as long as they satisfy any of the in-place holds in existence. When a query-based hold is removed, the items that were retained are purged from the database unless they are subject to another in-place hold.

Figure 11-38 shows the contents of the DiscoveryHolds folder for a mailbox that is subject to an in-place hold. You can see that the folder is a subfolder of Recoverable Items and the items in the folder.

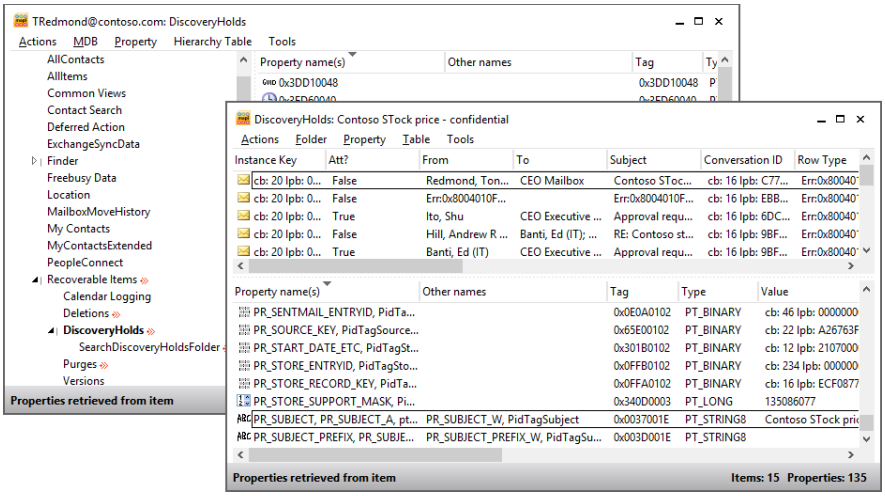


Figure 11-38 Viewing the contents of the DiscoveryHolds folder with MFCMAPI

After the last query-based holds are removed from a mailbox, the items held in the DiscoveryHolds folder are removed.

Using groups with searches

It might seem convenient to base the target set for a search on a distribution group or even a dynamic group, and, indeed, EAC enables you to enter the name of a group when configuring the mailboxes for a hold as part of the list covered by a search. However, when you save the details of the hold, Exchange detects that a group is involved and flags a warning similar to that shown in Figure 11-39. A hold is an individual setting for a mailbox. Although Active Directory provides backlinks to enable Exchange to associate the membership of a normal group with its underlying mailboxes, Exchange does not use these backlinks to point to mailboxes for search purposes. In any case, backlinks are not available for dynamic groups, whose membership is usually only expanded when needed by the transport system to determine the best routing arrangement for a message. If you want to apply a hold to a group, the only way Exchange can do this is to expand the membership of the group into the individual mailboxes and apply the hold in that manner. Any external entries that are part of the group (for example, a mail-enabled user belonging to another organization) are ignored. The rationale here is that you implement the hold against the membership of a group at a specific point in time when those users are considered pertinent to an investigation. Users who subsequently join the group might have nothing whatsoever to do with the issue at hand.

Expanding a group's membership makes it possible for Exchange to apply the hold. To do this, the set of mailboxes addressed by the dynamic group is populated into the search properties, and a link is created to the search in the mailbox properties. The mailbox names are added to the SourceMailboxes property of the search metadata, and the GUIDs that point to each mailbox are added to the Sources property.

However, it is critical to remember that the hold is only applied to the mailboxes that constitute the group's membership at the point in time when you save the search. Exchange does not include a mechanism to detect when a mailbox is added to or removed from a group so that it can add or remove the hold. This responsibility therefore remains totally in the hands of the administrator or other person who oversees the hold. Ideally, she should check the hold on a regular basis to see whether it is effective and doing what it is supposed to do, including validating that the correct set of mailboxes are under the control of the hold.

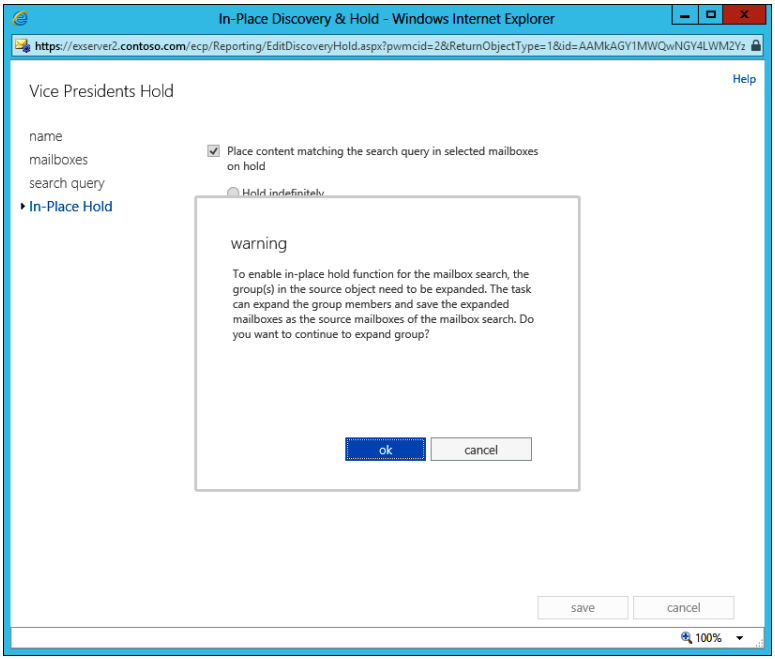


Figure 11-39 Warning after a group is added to a hold

Although groups have limitations when you create a search, they also have their advantages. From the earlier description of creating a new search, you know that:

- You cannot incorporate a hold for mailboxes covered by a search if the All Mailboxes option is selected.
- The EAC user interface (UI) is acceptable when adding 10 or 20 mailboxes to a search; it is less satisfactory if hundreds or more mailboxes have to be added. Remember that some throttling settings that govern the use of system resources during searches might have to be adjusted if you want to conduct searches for large numbers of mailboxes.

You can work around the first restriction if you create a group containing all the mailboxes in an organization and add it to a search. You can work around the second by creating a group containing all the target mailboxes first and then adding the group to the search. In both cases, the relevant group can be easily created by using simple Windows PowerShell commands: Get-Mailbox to retrieve the mailbox information, New-DistributionGroup to create the group, and Add-DistributionGroupMember to populate the group.

Another way of solving the problem is to use EAC to create a new dynamic distribution group and select Users With Mailboxes as the basis on which to create the group. You can then use this group to add mailboxes to the search. Alternatively, you can create a dynamic group that includes all user mailboxes with a command like this:

```
New-DynamicDistributionGroup -Name 'All Contoso Mailboxes' -RecipientFilter
"((RecipientTypeDetails -eq 'UserMailbox'))" -RecipientContainer 'Contoso.com'
-Alias 'ContosoMbx'
```

In either case, it is wise to validate the set of mailboxes after EAC expands them and adds them to the search to ensure that you're not going to search mailboxes that are unnecessary and unwanted. Adding many mailboxes to a search using a group will probably also run into throttling problems, so you must adjust the throttling policy for users that run searches if you plan to add more than 50 mailboxes to searches using groups.

Removing a search

To remove a search, select it in EAC and click the wastebasket icon or run the Remove-MailboxSearch cmdlet in EMS. However, before you remove a search that includes an in-place hold, you have to edit its properties to remove any of the mailboxes that are covered by the search and then save the updated search. This enables Exchange to remove the links to the search that are held in mailbox properties. If you don't remove the mailboxes, you'll see an error similar to that shown in Figure 11-40.

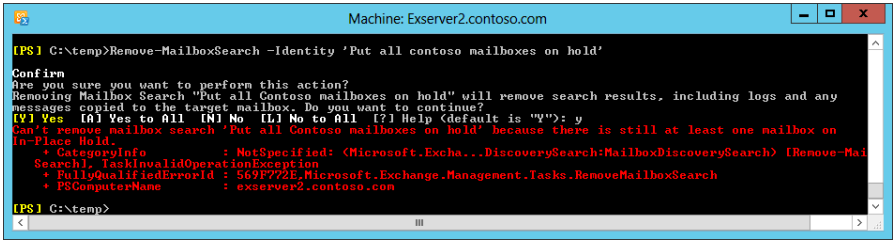


Figure 11-40 Error when removing a mailbox search

Removing a lot of mailboxes from a search through EAC can take some time. It's much easier to run the following commands to remove the mailboxes and then delete the search with EMS:

```
Set-MailboxSearch -Identity 'Put All Contoso Mailboxes on Hold' -SourceMailboxes
$Null

Remove-MailboxSearch -Identity 'Put All Contoso Mailboxes on Hold' -Confirm:$False
```

Removing a search does not delete any content previously recovered by executing the search that might remain in a discovery mailbox. This data has to be removed separately,

preferably under the direction of the legal department in a way that respects user confidentiality.

Executing searches with EMS

EAC is a very convenient interface with which to create and initiate searches, but you can do the same through EMS by using a set of cmdlets that are exposed only if you are a member of the Discovery Management role group. These cmdlets are as follows:

- **New-MailboxSearch** Creates and initiates a new mailbox search.
- **Get-MailboxSearch** Retrieves details of a mailbox search.
- **Set-MailboxSearch** Changes the search criteria for a search that has already been created.
- **Start-MailboxSearch** Restarts a mailbox search.
- **Remove-MailboxSearch** Removes a mailbox search. This action also removes from the discovery mailbox all the items a search finds.
- **Search-Mailbox** An older cmdlet that is now mainly used to search and remove content from user mailboxes. (See the "Search and destroy" section earlier in this chapter.) EAC does not use it.

For example, a new search to look for information about potential illegal stock trading by company officers could be initiated with this command:

```
New-MailboxSearch -Name "Stock Trading Discovery 2" -SourceMailboxes 'Company
Officers'  -TargetMailbox 'DiscoveryMailbox@contoso.com' -StartDate '10/01/2010'
-EndDate '11/30/2010' -SearchQuery "XXE Stock tip" -StatusMailRecipients
'LegalSearch@contoso.com' –SearchDumpster –DoNotIncludeArchive –EstimateOnly
-IncludeUnsearchableItems –ExcludeDuplicateMessages:$False -LogLevel Full
```

Table 11-4 lists the most important parameters you are likely to use with the New-MailboxSearch cmdlet and their meaning.

TABLE 11-4 Important parameters for the New-MailboxSearch cmdlet

Parameter	Meaning
Name	A unique identifier for the search that should be something meaningful such as "Illegal stock trading review."
SourceMailboxes	Specifies the mailboxes Exchange will search. If you have more than a few mailboxes to search, it is more convenient (and probably more accurate) to create a distribution group to identify the mailboxes to include in the search. If you don't specify the –SourceMailboxes parameter, Exchange searches all mailboxes.

Parameter	Meaning
TargetMailbox	Specifies the SMTP email address of the discovery mailbox in which you want to store the search results. The default discovery mailbox has a rather long and complicated email address, so it's easier to assign a new and shorter secondary email address to the mailbox to make it easier to type. In fact, this mailbox doesn't have to be a discovery mailbox; Exchange is happy to place search results in any mailbox you select.
SearchQuery	A KQL-format query that Exchange executes to locate items in the target mailboxes. In the example, Exchange matches any of the words in the search query. This search query is a very simple one, and some trial and error is probably required to arrive at the best query. If you omit the search query, Exchange finds every item in every mailbox you include in the search and stores copies of all those items in the discovery mailbox. This kind of search can inundate a server with work.
StatusMailRecipients	Tells Exchange the recipients who should be notified by email after the search is complete. No message is sent if you don't provide a value for this parameter. You can provide one or more recipient SMTP addresses to receive notifications, separating each address with a comma. It's often more convenient to use a distribution group for this purpose.
SearchDumpster	Forces Exchange to include the contents of the Recoverable Items folder in the search. All searches executed through EAC include this parameter.
DoNotIncludeArchive	Instructs Exchange to ignore items stored in any personal archives that are assigned to mailboxes.
EstimateOnly	Tells Exchange that it is to run a search estimate only rather than copy items that match the search criteria to the discovery mailbox.
ExcludeDuplicateMessages	Tells Exchange how to deal with duplicate items it encounters in mailboxes. Set the parameter to $True to force Exchange to de-duplicate (copy only a single instance of an item) or $False to copy every copy of an item it finds.
LogLevel	Dictates the level of logging Exchange performs for the search. Valid options are Suppress, Basic (default), and Full. If Basic or Full is chosen, Exchange creates a search report in the root folder for the search in the discovery mailbox.

INSIDE OUT **Running concurrent searches**

You can run concurrent searches as long as each search has a different name. The searches proceed a little more slowly because of contention when writing found items into the discovery mailbox. If you need to run concurrent searches on an ongoing basis, it would be a good idea to spread the load by creating several discovery mailboxes and locating them in different databases.

What Exchange can search

All item types stored in an Exchange mailbox database are discoverable, including voice messages, drafts, attached documents of various formats, and IM conversations (if stored in mailboxes). Exchange 2013 searches depend on Search Foundation to build and maintain context indexes extracted from mailbox databases. Although Search Foundation has no difficulty indexing the complete body text of messages because they are plaintext, rich text format, or HTML, some issues might be encountered with attachments, which can be in any format. Before Search Foundation can include the actual content for an attachment in its indexes rather than simply its metadata (such as the file name or author name), it must be able to extract the content. Search Foundation includes a large number of filters to enable it to deal with the most common file formats, including Microsoft Word, HTML, Microsoft PowerPoint PPTX files, and Adobe PDF. An additional set of IFilters, including those for Microsoft Excel, OneNote, older versions of PowerPoint, and Open Document files, is provided to Search Foundation when Exchange 2013 is installed on a Mailbox server (other IFilters are available from third-party vendors if you need to be able to index a specific format). Between Search Foundation and Exchange, a very large set of file formats can be indexed. To see the full set of searchable file formats on a server, you can run this command:

```
Get-SearchDocumentFormat | Format-Table Name, Extension, FormatHandler -AutoSize
```

The Set-SearchDocumentFormat command is also available to change the way Exchange processes particular formats. For example, you can disable indexing of particular types of files by running this command, a step that requires some forethought because of its potential impact on indexing and subsequent discovery operations.

Even though Search Foundation possesses out-of-the-box capabilities by which it can index the bulk of items encountered in an Exchange environment, it is possible for your company to create content in a format that Search Foundation does not know about. In this case, you must install an IFilter that supports the specific format on all Mailbox servers.

Search Foundation then detects and uses the IFilter to include the items in that format in its indexes. If you do not install an IFilter, Search Foundation indexes the metadata for the items to allow searches to proceed, but these items will be deemed unsearchable and returned as such when you execute a search. Apart from application-specific files, other items Exchange deems unsearchable include items encrypted with Secure Multipurpose Internet Mail Extensions (S/MIME). However, messages protected with Active Directory Rights Management Services (AD RMS) remain searchable for discovery purposes.

When you decide to copy search results to a discovery mailbox, you can include unsearchable items. Normally, it's a good idea to do this because the person assigned to review the search results might be able to discover what these unsearchable items contain, perhaps by examining the context of where the item was discovered. (An item found in a folder called Videos is likely to contain video content, for instance.)

You can see a list of unsearchable items with the Get-FailedContentIndexDocuments cmdlet. When you run the cmdlet, you can pass it the name of a server to see all items on a server or just a mailbox database to see the unsearchable items in the content index for that database. For example, this command lists various issues that were encountered in a specific database:

```
Get-FailedContentIndexDocuments -MailboxDatabase DB2
```

```
DocID   Database  Mailbox        Subject   Description
-----   --------  -------        -------   -----------
77      DB2       SystemMai...             The document parser encountered a processing error.
78      DB2       SystemMai...             The document parser encountered a processing error.
1287    DB2       Rob Young                The document parser encountered a processing error.
```

You can see that a number of items in Rob Young's mailbox have had an issue. Exchange assigns each of the items an identifier (DocId), but there's no way to extract details for a specific item. Instead, you have to run the cmdlet again, this time using the mailbox parameter to restrict the output to just details for Rob Young's mailbox. To see additional information, pipe the results to the Format-List cmdlet and then redirect the output to a text file you can then interrogate at your leisure to see what you can discover. The command might look like this:

```
Get-FailedContentIndexDocuments -Mailbox 'Rob Young' | Format-List > C:\temp\Docs.txt
```

You can then search through the output text file to see whether anything captured there provides an indication of why an item is unsearchable. For example, you can see in this extract that the parser used to extract the content from an item was unable to complete for some reason and that the item is a Word document.

```
DocID            : 1667
Database         : DB2
MailboxGuid      : 4e09fc34-e61a-4eea-87b8-d19b214a92ab
Mailbox          : Rob Young
SmtpAddress      : Rob.Young@dublin.contoso.com
Subject          : RE: 2003/2010 coexistence
ErrorCode        : 7
Description      : The document parser encountered a processing error.
AdditionalInfo   :  309003 Document 'exchange://localhost/
Attachment/298057e9-43de-417b-a740-7ab58b6e48bb/eb22b1a1-b1c9-4972-a163-
ba508f018d6b/919123003011.1/Transitioning Client Access to Exchange 2013.docx'
was partially processed. The parser was not able to parse the entire document.
IsPartialIndexed : False
FailedTime       : 01/4/2013 11:27:52
```

Should you be worried if many unsearchable items exist for your database? It depends. First, it depends on the percentage of unsearchable items. If 0.0002 percent of items are unsearchable, it's probably acceptable because any search has a very high chance of discovering information that's required. Second, it depends on the items that are failing to be indexed. If they are all of the same type and a filter is available, you can install that filter to solve the problem. However, if the items are of a type for which a filter is not available or that is known to be unsearchable (such as S/MIME encrypted items), you might have to live with the situation.

INSIDE OUT Controlling the search black box

In most respects, Search Foundation is a black box to Exchange, No user interface is available to control how Exchange interacts with Search Foundation. However, you can affect how Search Foundation operates by making some changes to its parameters held in the system registry (which therefore only apply to an individual server). These settings are found at HKLM\Software\Microsoft\ExchangeServer\V15\Search \SystemParameters and the most important are:

- **MaxAttachmentCount** Default is 10; controls how many attachments to an item will be indexed.
- **MaxAttachmentDepth** Default is 2; controls how many nested levels within a document structure will be indexed.
- **ProcessImages** Default is 0 (zero); controls whether images are indexed.

Normally, a relatively small number of items turn out to be unsearchable. In addition, remember that item metadata (sender, recipients, subject, and so on) and message bodies are always indexed and searchable, so if a small percentage of attachments can't be searched, it probably won't be of great concern in a legal search. After all, if people are doing something they shouldn't, they are likely to leave some trace of their activity in a searchable property that can be discovered. After this happens, the next step is often for investigators to take a complete copy of the suspect's mailbox to conduct a detailed search to discover what it contains, and any lurking unsearchable items can be reviewed at that time.

Important

An in-place hold depends on the ability of Exchange to understand when an item might satisfy the criteria stated for the hold. Unsearchable items might not expose sufficient information to Exchange for it to assess whether these items should be retained and so create the potential for required items to be removed from mailboxes. If large numbers of unsearchable items are created because of an application you use or other reason, it's best not to use a query-based hold. Instead, you can create a hold on everything in mailboxes to make absolutely sure that everything that might be required to satisfy a search is available and can be reviewed manually if necessary. This will add a little overhead to the way searches are performed, but it's the best way to ensure that nothing slips through.

Search syntaxes

Exchange 2010 uses AQS (Advanced Search Syntax) to construct its multimailbox searches. Exchange 2013 takes a different approach and uses KQL (keyword query language). Why the change?

AQS is shared with other Windows search components such as Windows Desktop Search, which Outlook clients use. In fact, Exchange 2010 supports only a subset of the full AQS capabilities. However, KQL is shared with other Office 2013 applications, the most important of which is SharePoint 2013 because the two applications can form a single discovery domain across the email stored in Exchange and the documents held in SharePoint.

Giving Exchange and SharePoint a common search syntax makes great sense and is the driving force behind making the change to search syntax in Exchange 2013. Another advantage is gained in that KQL can perform proximity searches. When you want to search for items that mention the words "Azur project" and have the word "bribe" somewhere close to those words, AQS can certainly find anything that includes "Azur project" AND "bribe," but it can't find "Azur project" with "bribe" within 30 words. (In KQL syntax, the

word "bribe" is NEAR [*n*=30] the other phrase.) This capability could be very useful in searches that start by being somewhat imprecise because you're not quite sure about what you're looking for. It's true that searches like this might produce more results than you can deal with on a practical basis, but they could provide a hint about how searches might be refined to home in on the critical items. KQL also supports wildcard searches, meaning that you could use a term such as *toso or cont*, both of which will force the search to find items relating to "Contoso."

For more information on KQL syntax, see *http://msdn.microsoft.com/en-us/library/ee558911.aspx/*.

KQL syntax is powerful. It will be interesting to see how it is used to frame search queries as Exchange 2013 is deployed. Even better, the Exchange community can learn KQL tips and techniques to improve searches from those who work with SharePoint and vice versa.

The value of the Recoverable Items structure

Given the discussion so far about retention and the importance of immutability in achieving compliance, it's possible for users to attempt, probably inadvertently, to remove data required to meet regulations or some other control by deleting items from their mailboxes. This is why Exchange contains the Recoverable Items structure to hold deleted items and other information that needs to be retained. Items are moved into the structure when a user deletes them, when the MFA executes the Delete And Allow Recovery action for items when their retention period expires, or when Exchange needs to retain an item as the result of an in-place hold. Previous versions of the Recoverable Items structure were known as the "dumpster," a term that is used in many documents and other discussions about Exchange.

The Recoverable Items structure is present for every mailbox and saves administrators from the need to restore mailbox databases to recover items that users have deleted in error. Of course, not every user can justify the expense of going through a full database recovery, and even the most important users probably can't justify the expense for every item they delete in error, but mailbox restores for item recovery were quite a common practice before Microsoft first added the ability to retain items and allow users to recover those items in Exchange 2000. The elimination of these restore operations is of huge value to administrators, and that's why the Recoverable Items folder is one of the high-value, low-cost features in Exchange.

The function of the Recoverable Items structure

Before considering how the Recoverable Items structure operates in Exchange 2013, you should review some basics. By default, the Recoverable Items structure (a root folder and a set of subfolders) holds deleted items for a retention period. Items are moved into Recoverable Items after a user selects an item from any folder in the mailbox and deletes

it using the Shift+Delete combination, a function that bypasses the normal approach by which items are deleted. The items then move into the Deleted Items folder, where they remain until the folder is emptied, and subsequently move into Recoverable Items.

The default deleted items retention period is 14 days. The deleted items retention period set for a mailbox is inherited from the mailbox database in which the mailbox is first created and can be changed afterward to meet whatever requirements exist. For example, you might decide that the mailboxes executives use should have a deleted items retention period of 120 days, although all other mailboxes have a retention period of 60 days.

You can run the following command to examine the properties that govern deleted items retention on a database. The equivalent properties for an individual mailbox are visible by running Get-Mailbox:

```
Get-MailboxDatabase -Identity 'DB1' | Format-List *delete* -AutoSize
```

Items in the Recoverable Items folder are regarded as being soft deleted in that users have deleted them from their original folder and then emptied the Deleted Items folder. Because they are soft deleted, the items can still be recovered by being moved back into a folder and so remain in the database. In versions prior to Exchange 2010, soft-deleted items are kept in the Deleted Items folder but are hidden from the user's view. From Exchange 2010 onward, when users empty their Deleted Items folder, Exchange moves the items into a new subfolder under Recoverable Items called Deletions. The items in this folder are what Outlook and Outlook Web App show when a user selects Recover Deleted Items. Non-MAPI clients that use protocols such as POP3 and IMAP4 do not include the user interface or the basic support in the protocols to enable recovery from the Recoverable Items folder, and that's why you don't find the feature in these clients.

When you select Recover Deleted Items, the client displays a list of all the items held in the Recoverable Items folder. For example, Figure 11-41 shows a list of deleted items from my mailbox as displayed through Outlook Web App. The list includes items deleted from all folders.

Recoverable items expire when the mailbox's deleted items retention period expires, at which time the items are hard deleted or permanently removed from the database. The MFA is responsible for removing items from the database, so an item is finally removed from the database the next time the MFA runs after its retention period expires. Deleted items that are still recoverable don't count against the user's normal mailbox quota, so an extended retention period won't make any difference to users, except that they can recover items at any time up until the retention period passes.

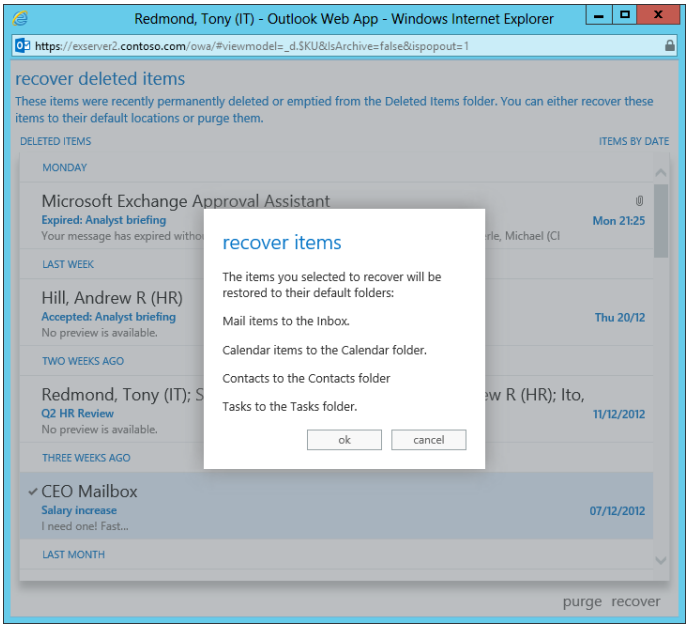


Figure 11-41 Recovering deleted items from the Recoverable Items folder

Experience says: Use longer deleted items retention periods

By contrast, when removed from the database, items can be made available again only by an administrator after considerable effort to restore the database that contains the mailbox from a backup and export the recovered items to a PST to provide to the user. Therefore, experienced Exchange administrators prefer to use longer rather than shorter deleted items retention periods. The default 14-day period is a good starting point, but some feel that an even longer period is better because it makes absolutely sure that an administrator never has to go searching for a backup to recover an item for a forgetful user. After all, if someone can't remember that she made a mistake in deleting an item within 120 days (and who can remember deleting an item that long ago?), maybe the item isn't important enough to warrant administrator intervention.

Improvements in Exchange 2013

From a user perspective, the experience of working with the Recoverable Items folder is similar to previous versions. Behind the scenes, the introduction of new features in both Exchange 2010 and Exchange 2013 to meet legal and regulatory requirements has had a massive influence on how items are stored and retained in Recoverable Items. If mailboxes

are under any form of hold, their owners shouldn't be able to affect deleted items because these items could form part of the legal record, and if a user could remove or alter these items, it would compromise the ability of Exchange to provide an immutable repository. If you've been accustomed to the way that Recoverable Items worked in Exchange 2007 or earlier versions, you'll now see that:

- Items in Recoverable Items are included when you move a mailbox from one database to another rather than being purged as in versions prior to Exchange 2010. This prevents the loss of any item that should be retained as a consequence of normal rebalancing of server load.
- Items in Recoverable Items are indexed so that they can be discovered by searches.
- The Store maintains a separate quota for Recoverable Items.
- Versions are maintained of changes made to items in the mailbox.
- Users cannot purge data by using Recover Deleted Items to view the contents of Recoverable Items, selecting one or more items, and then deleting them (by using the Purge option shown previously, in Figure 11-41).
- Data in Recoverable Items is maintained on a per-mailbox rather than a per-folder basis. In other words, you cannot tell from which folder an item was originally deleted.

The Exchange 2007 version of Recoverable Items uses a hidden view maintained on a per-folder basis. As items are deleted, the Store sets a MAPI flag (ptagDeletedOnFlag) and starts the retention time countdown. The deleted items stay in their original folder but are hidden from clients. They cannot be indexed or searched because Outlook and other clients don't see these items. This mechanism works, but it can't accommodate the requirements for indexing to allow eDiscovery searches. The Recoverable Items folder is located in the non-IPM subtree of user mailboxes. Because it's designed to hold data required for internal purposes, no client interface ever exposes this subtree, so the folder remains invisible to users. In Exchange 2010, the Recoverable Items folder contains three subfolders, Deletions, Versions, and Purges, that hold items at different points in their journey toward eventual removal from the Store. Using folders instead of views enables indexing and searching and makes sure that data stored in Recoverable Items are moved along with the rest of the mailbox. The Audits subfolder is also present if the mailbox is enabled for auditing. (See the "Auditing mailbox access" section later in this chapter.) Audit entries age out after 90 days by default, so this folder can contain many items if a high degree of audit settings is enabled on the mailbox.

Exchange 2013 adds the DiscoveryHolds subfolder previously described in the "How in-place holds work" section earlier in this chapter. Figure 11-42 demonstrates how

MFCMAPI can be used to interrogate the Recoverable Items folder to reveal some of its secrets. On the left side, you can see the subfolders under Recoverable Items. On the right, you can see that the Deletions subfolder holds items waiting for the deleted items retention period to expire. These are the items users see when they use a client's Recover Deleted Items option.

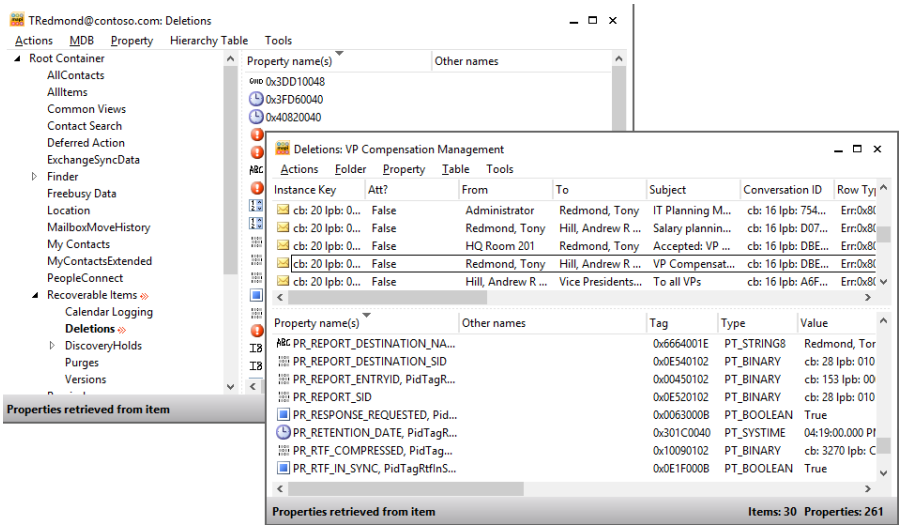


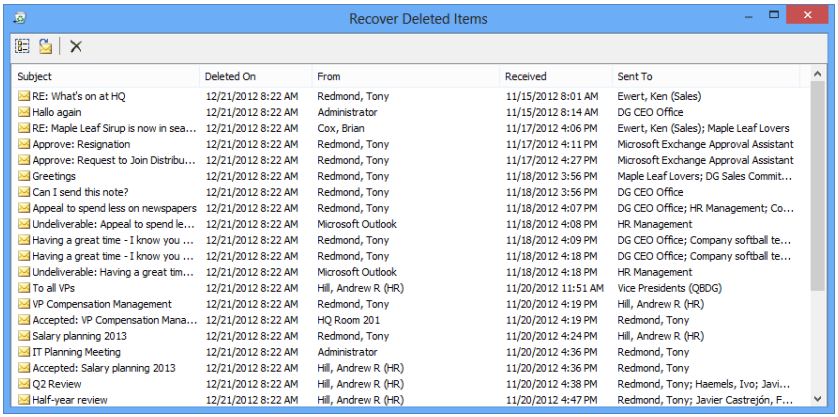
Figure 11-42 MFCMAPI exposing the Recoverable Items structure

In essence, the Deletions folder replaces the MAPI flag and hidden view Exchange 2007 used. When a user deletes an item as normal (soft delete) by emptying the Deleted Items folder, the Store moves the item into the Recoverable Items\Deletions folder. Recover Deleted Items accesses this folder even when accessed by older Outlook clients because the remote procedure call (RPC) Client Access Layer interprets client requests that were previously satisfied using the hidden view through items retrieved from the Deletions folder. If the user has a personal archive, a separate set of Recoverable Items folders is maintained in the archive to handle the deletions that occur for archived items.

The Store does not preserve the folder context for deleted items; you don't see the folder from which an item was deleted when you browse through items in Recoverable Items. This isn't usually an issue unless you have a very large number of deleted items in the mailbox and must peruse a long list to find the right item to recover, or a user deletes a complete, large folder by accident and is faced with the need to find and recover all the items from the deleted folder from among the mass of other items held in Recoverable Items.

Some observers have commented that this situation happens often enough to keep them busy restoring deleted folders from backup copies, in turn meaning that the thought of

ever going to a no-backup regime for Exchange is impossible until the Store captures folder information and holds it along with the other properties of deleted items. For now, as Figure 11-43 shows, the default sort order used in Recoverable Items is the date and time an item is hard deleted (removed from the Deleted Items folder). Sometimes it is easier to find items if you click the appropriate column heading to sort by message subject or author. Microsoft is considering how best to improve matters in future versions of Exchange, perhaps by supporting the preservation of the folder structure for deleted items, which could enable you to find items based on the folder from which they were originally soft deleted.



Subject	Deleted On	From	Received	Sent To
RE: What's on at HQ	12/21/2012 8:22 AM	Redmond, Tony	11/15/2012 8:01 AM	Ewert, Ken (Sales)
Hallo again	12/21/2012 8:22 AM	Administrator	11/15/2012 8:14 AM	DG CEO Office
RE: Maple Leaf Sirup is now in sea...	12/21/2012 8:22 AM	Cox, Brian	11/17/2012 4:06 PM	Ewert, Ken (Sales); Maple Leaf Lovers
Approve: Resignation	12/21/2012 8:22 AM	Redmond, Tony	11/17/2012 4:11 PM	Microsoft Exchange Approval Assistant
Approve: Request to Join Distribu...	12/21/2012 8:22 AM	Redmond, Tony	11/17/2012 4:27 PM	Microsoft Exchange Approval Assistant
Greetings	12/21/2012 8:22 AM	Redmond, Tony	11/18/2012 3:56 PM	Maple Leaf Lovers; DG Sales Commit...
Can I send this note?	12/21/2012 8:22 AM	Redmond, Tony	11/18/2012 3:56 PM	DG CEO Office
Appeal to spend less on newspapers	12/21/2012 8:22 AM	Redmond, Tony	11/18/2012 4:07 PM	DG CEO Office; HR Management; Co...
Undeliverable: Appeal to spend le...	12/21/2012 8:22 AM	Microsoft Outlook	11/18/2012 4:08 PM	HR Management
Having a great time - I know you ...	12/21/2012 8:22 AM	Redmond, Tony	11/18/2012 4:09 PM	DG CEO Office; Company softball te...
Having a great time - I know you ...	12/21/2012 8:22 AM	Redmond, Tony	11/18/2012 4:18 PM	DG CEO Office; Company softball te...
Undeliverable: Having a great tim...	12/21/2012 8:22 AM	Microsoft Outlook	11/18/2012 4:18 PM	HR Management
To all VPs	12/21/2012 8:22 AM	Hill, Andrew R (HR)	11/20/2012 11:51 AM	Vice Presidents (QBDG)
VP Compensation Management	12/21/2012 8:22 AM	Redmond, Tony	11/20/2012 4:19 PM	Hill, Andrew R (HR)
Accepted: VP Compensation Mana...	12/21/2012 8:22 AM	HQ Room 201	11/20/2012 4:19 PM	Redmond, Tony
Salary planning 2013	12/21/2012 8:22 AM	Redmond, Tony	11/20/2012 4:24 PM	Hill, Andrew R (HR)
IT Planning Meeting	12/21/2012 8:22 AM	Administrator	11/20/2012 4:36 PM	Redmond, Tony
Accepted: Salary planning 2013	12/21/2012 8:22 AM	Hill, Andrew R (HR)	11/20/2012 4:36 PM	Redmond, Tony
Q2 Review	12/21/2012 8:22 AM	Hill, Andrew R (HR)	11/20/2012 4:38 PM	Redmond, Tony; Haemels, Ivo; Javi...
Half-year review	12/21/2012 8:22 AM	Hill, Andrew R (HR)	11/20/2012 4:47 PM	Redmond, Tony; Javier Castrejón, F...

Figure 11-43 How Outlook 2013 presents deleted items for recovery

Note

The Store does not retain information if you delete a mailbox; it handles deleted items from active mailboxes only, so if you delete a mailbox, its contents are no longer visible to Exchange for functions such as eDiscovery searches. You can't suspend the final removal of a mailbox and retain it indefinitely or for a specified period, either, because the Store removes a deleted mailbox permanently from its database when the deleted mailbox retention period expires. If you need to retain information in the mailboxes of people who leave the organization so that the items in the mailboxes can be found by eDiscovery searches, you can disable the mailbox and keep it for as long as the items might be required. You can also import any PSTs you want to retain into the mailbox to make them available to searches.

Single-item recovery

Exchange 2010 and Exchange 2013 both include the ability to enable single-item recovery (SIR) for mailboxes. This feature provides a method of short-term content preservation and closes a gap that could force an administrator to recover an item from a backup copy when:

- A user runs Recover Deleted Items, selects an item, and purges it. If SIR is not enabled and a hold is not in place for the mailbox, Exchange performs a hard delete, and the item is removed from the database.
- The MFA processes the Recoverable Items folder and removes any items there because of a retention policy.

When SIR is enabled for a mailbox, Exchange does not perform a hard delete. Instead, items that would normally be removed from the database are retained in the Purges subfolder of the Recoverable Items folder until the deleted items retention period for the mailbox elapses, at which time they are removed. Keeping the items in the Purges folder hides them from the user. However, if necessary, as when the user says, "Oops, shouldn't have done that," an administrator can recover the item without using a backup copy. Items held in the Purges folder remain indexed and are discoverable. SIR is enabled for a mailbox by running the Set-Mailbox command. For example:

```
Set-Mailbox –Identity 'Andy Ruth' –SingleItemRecoveryEnabled $True
```

When the data from executives or other users who need to use SIR are gathered in a single database, you can enable all the mailboxes by piping a list of mailboxes into the Set-Mailbox cmdlet. In this example, you also set the period for the Store to maintain the deleted items to 28 days. (The default is 14.)

```
Get-Mailbox -Database 'DB1' | Set-Mailbox -SingleItemRecoveryEnabled $True
–RetainDeletedItemsFor 28
```

In addition to tracking deletes, when SIR is enabled for a mailbox, Exchange tracks any attempt to update items and takes a copy of items (called a copy on write) to preserve versions in case they are required for discovery purposes. The copies of the altered items are held in the Versions subfolder of Recoverable Items. In essence, copy on write prevents users from covering their tracks by editing information held in their mailbox. Copy on write is also enabled when a mailbox is subject to a litigation or in-place hold (in which case, the copy of the item is kept in the DiscoveryHolds subfolder). All the versions are indexed and discoverable.

Technically speaking, any action that changes an item generates a new version through a copy-on-write operation. Changes to subjects, message bodies, attachments, sender and recipient details, and date information are all actions that generate a new version.

Table 11-5 lists the actions that cause Exchange to retain a new version of an item in the Versions subfolder.

TABLE 11-5 Actions that cause item versions to be generated

Item	Actions that cause versions to be retained
Messages and posts	Updates to: ● Subject ● Item body ● Attachments ● Sender or recipient data ● Send or received dates
Other item types	Changes to any property visible to a client except: ● The folder in which the item is stored ● Item read status ● Retention tag status

Draft items are the only exception to deleted items processing. A draft item is one that has the unsent bit set in the MAPI message flags. If this bit is set, the Store does not capture updates in the versions folder. There are two reasons for this. First, a draft item typically goes through multiple revisions that might be captured by a client's autosave process before it is eventually sent. Second, a draft item is not really interesting for discovery until it is sent and becomes a full-fledged communication to another person. After all, it's not a problem if a user thinks about doing something wrong, such as making an illegal recommendation to someone else to engage in insider trading, and captures the thought in a draft message. The thought only becomes a problem and consequently of interest for discovery purposes if the user sends the message to another person.

Items captured through SIR remain in the Purges and Versions folders until their normal retention period expires. When this happens, the MFA removes the items as normal. If you need to keep items for an indefinite period, you should enable litigation hold for the mailbox to instruct Exchange to keep everything until the hold is lifted.

Table 11-6 summarizes the data retention states an Exchange mailbox can be in from the default position, in which the mailbox essentially operates much like Exchange 2007, through the enabling of SIR. This provokes the use of the Purges and Versions folders, until litigation hold or the application of an in-place hold freezes the mailbox and prevents any data from being removed until the hold is released.

TABLE 11-6 Data retention states for Exchange 2013 mailboxes

Mailbox status	Deleted items kept in Recoverable Items folder	Versions and purges kept in Recoverable Items folder	User can delete items from the Recoverable Items folder	When message management removes items from the Recoverable Items folder
Default	Yes	No	Yes	After deletion item retention period expires (120 days for calendar items)
Single-item recovery enabled	Yes	Yes	No	After deletion item retention period expires (120 days for calendar items)
Litigation hold enabled for mailbox	Yes	Yes	No	Items retained until litigation hold is released
In-place hold enabled for mailbox	Yes	Yes	No (for items matching the in-place hold criteria)	Items retained until hold expires (could be indefinite)

To recover an item preserved by SIR, you can run the Search-Mailbox cmdlet in EMS or a normal eDiscovery search as described earlier in this chapter to locate the item. An extremely precise filter might be necessary to locate the desired item. When located, the item can be exported from the discovery mailbox to a PST. Another option is to run the New-MailboxExportRequest command, which can export items to a mailbox complete with everything held in the Recoverable Items subfolders.

Initial measurement of the effect of SIR on mailbox sizes indicates a growth of 3 to 5 percent for a 14-day retention period. It's unusual to enable every mailbox in a database for SIR unless you have dedicated databases for VIPs or other users affected by legal proceedings, so the overall effect on a mailbox database is usually less.

Knowing what's in Recoverable Items

A user can view the items in the Recoverable Items folder at any time by using Recover Deleted Items in Outlook or Outlook Web App. Administrators can't access items in the Recoverable Items folder for a mailbox unless they open that mailbox, but they can get a sense of how much information is held in the various subfolders by using the

Get-MailboxFolderStatistics cmdlet and pointing it at the Recoverable Items folder with the –FolderScope parameter. For example:

```
Get-MailboxFolderStatistics –Identity TRedmond –FolderScope RecoverableItems | Select
Identity, ItemsInFolder, FolderSize, FolderType
```

As you know from the description in the previous section, the folders shown in the output are used as follows:

- The Recoverable Items (root) folder holds stripped versions of calendar items.
- The Calendar Logging folder stores logs generated when changes are made to a calendar item. By default, up to 6 GB of such logs can be held, with the intention of making the logs available to the Calendar Repair Assistant to help it determine the history of a calendar item if any inconsistency arises. You can update this amount by changing the CalendarLoggingQuota property for a mailbox. Calendar logging is controlled by the CalendarVersionStoreDisabled property on a mailbox. By default, the property is set to $False, so calendar logging proceeds.
- The Deletions folder stores any soft-deleted items.
- The Purges folder stores any hard-deleted items.
- The Versions folder stores any previous versions of deleted items that have been edited.
- The DiscoveryHolds folder stores any deleted items that meet the requirements of whatever in-place holds exist for the folder.

Calendar items are held in Recoverable Items for 120 days. Stripped versions of calendar items have no attachments. Exchange creates these copies from calendar items that are purged or updated in Recoverable Items to use as logs to track these changes. The stripped items are stored in the Recoverable Items folder. Full copies of items that are changed or purged are also stored in the Versions or Purges folders, respectively.

Managing quotas for Recoverable Items

Although deleted items held in Recoverable Items do not count against a user's normal mailbox quota, it makes sense for Exchange to exert some form of control over the folder; otherwise, the potential exists for it to swell to an uncontrollable size. Two properties are used for this purpose, as follows:

```
Get-Mailbox –Identity 'Tony Redmond' | Format-Table Recover*
```

The default quotas should be sufficient to allow for the retention of all necessary material in the majority of cases. In some situations, you might need to increase the assigned quotas, such as when you have to deal with scenarios such as:

- Mailboxes that receive extremely heavy traffic.
- Mailboxes that handle large numbers of attachments.
- Mailboxes whose content must be retained for extended periods (five years or more).
- Mailboxes that are affected by bugs. For example, in 2012, problems in the Apple iOS mail app caused calendar versions to accumulate in the Recoverable Items folder of mailboxes that used iOS devices to connect to Exchange through ActiveSync. Hopefully, this situation will not occur again, but it does illustrate how the Recoverable Items quota might be used in an unexpected way.

When a mailbox exceeds its recoverable items warning limit, Exchange logs an event in the Application Event Log of the Mailbox server that currently hosts the active copy of the mailbox. Normally, 10 GB of quota remains from the warning until the final limit is reached, so this should provide sufficient time to detect and rectify the problem.

If a mailbox reaches its recoverable items quota, users cannot empty the Deleted Items folder because this would cause the quota to be exceeded. For the same reason, they cannot permanently delete items from other folders by using the Shift+Delete combination. Because items remain discoverable in their current folders, these symptoms are problematic but not critical, but losing the ability to preserve copy on write for altered items is a big issue in terms of system immutability. It is therefore critical to conduct a regular scan of high-profile mailboxes that are subject to in-place holds to check that they are remaining under quota.

A simple one-line EMS command to retrieve information about mailboxes that are currently subject to in-place holds provides an excellent starting point for such a scan. Here's the code.

```
Get-Mailbox -ResultSize Unlimited | Where-Object {$_.InPlaceHolds -ne $Null} |
Get-MailboxFolderStatistics -FolderScope RecoverableItems | Format-Table Identity,
ItemsInFolder, FolderSize > C:\Temp\InPlaceFolders.txt
```

Your imagination (and some trial and error) can make this code a much better and more nicely formatted report. Those who simply want to see the data can review the text file output to discover any mailboxes that might run into quota problems soon.

Auditing administrator actions

As even a brief reading of this book reveals, an administrator can change many settings and create many new objects that influence the way Exchange operates. Up to Exchange 2010, no facility was available to track who did what and when at an administrative level. The addition of Windows PowerShell and its ability to affect many objects with relatively simple commands reinforced the need to log what happens within an Exchange organization.

Exchange 2013 includes the ability to audit actions taken by administrators in EAC and EMS. This is intended to enable organizations to maintain records of who did what and when to execute the cmdlets that manage Exchange. Apart from providing definitive proof about which account added a mailbox, changed properties on a connector, set up a new domain, or any of the myriad day-to-day operations that occur in an Exchange organization, maintaining an audit log can help satisfy regulatory requirements by demonstrating that strict controls are imposed on the work administrators do with Exchange.

Some administrators will view this development as yet another example of government surveillance as they struggle to keep the email system running. Others will consider increased oversight part of modern life much in the same way we all seem to be under video surveillance wherever we go. Since Exchange 2010 SP1, administrator auditing is enabled by default. The Set-AdminAuditLogConfig cmdlet controls how administration logging functions across the organization.

```
Set-AdminAuditLogConfig -AdminAuditLogEnabled $True
```

When logging is enabled, administrators see no indication that their actions are being captured in the audit log unless they search the audit log.

Enabling administrative logging instructs the Admin Audit Log agent, one of the standard sets of cmdlet extension agents shipped with Exchange 2013, to begin capturing details of administrative events. The admin audit agent runs on all Exchange 2013 servers to monitor the running of all cmdlets and record details of the cmdlets you configure to be audited. As described in Chapter 3, "The Exchange Management Shell," the execution of all business logic in Exchange 2013 flows through cmdlets so the agent can monitor all administrative operations.

INSIDE OUT **Disabling administrative auditing**

To disable administrative auditing, you run the Set-AdminLogConfig cmdlet and set the AdminAuditLogEnabled parameter to $False. Remember that this setting has to be replicated across the organization before it is effective on all servers, so it might take an hour or so before you can be sure that the setting is effective.

A number of other configuration settings are used to control the finer details of administrator audit logging. You can view the current audit configuration settings for the organization with the Get-AdminAuditLogConfig cmdlet. For example:

```
Get-AdminAuditLogConfig | Format-List
```

Auditing is performed on a cmdlet basis and can be further refined to select specific parameters to audit. By default, the use of every cmdlet and every parameter is audited, so the preceding configuration has values of {*} (asterisk). In fact, Exchange ignores auditing for cmdlets beginning with Get, Search, and Test to prevent cluttering up the audit log with entries for cmdlets that just retrieve or read information. The purpose is to audit operations that create or manipulate objects. Enabling the audit of every cmdlet is a bad idea because it generates a huge mass of audit entries, including entries for actions that are probably not important, such as a user updating her out-of-office (OOF) setting.

To focus on a specific set of cmdlets, define the cmdlets in a list passed in the AdminAuditLogCmdlets parameter for the Set-AdminAuditLogConfig cmdlet. For example, this command tells Exchange to audit any use of the New-Mailbox and New-DistributionGroup cmdlets and any cmdlet that has Transport in its name:

```
Set-AdminAuditLogConfig –AdminAuditLogCmdlets 'New-Mailbox, New-DistributionGroup,
*Transport'
```

Exchange now captures details about any creation of new mailboxes and distribution groups and any action taken with the cmdlets that manage the transport service. Assume that you just want to capture certain details about new mailboxes. This command captures details of the name, display name, and custom attributes for new mailboxes:

```
Set-AdminAuditLogConfig -AdminAuditLogParameters Name, DisplayName, *Custom'
```

You must arrive at a balance that captures the required auditing data but doesn't make it difficult to find an instance when necessary. It is likely that some trial and error will be required to settle on the right list of cmdlets and parameters to audit.

The audit mailbox

Exchange uses an AdminAuditLogs folder in the special arbitration mailbox with the display name Microsoft Exchange as the repository for administrator audit reports. This is a reasonably secure location because most administrators won't know about the existence of the folder. By default, audit reports are held for 90 days, and the MFA removes audit logs after their retention period expires. If you want to change the retention period, you can update it with the Set-AdminAuditLogConfig cmdlet. This command sets the audit log retention period to 182 days (approximately six months):

```
Set-AdminAuditLogConfig –AdminAuditLogAgeLimit 182.00:00:00
```

The audit configuration applies to administrator activity on an organization-wide basis so that all the reports generated across the entire organization go to the one mailbox. It is all too easy to fill this mailbox with audit reports if you use settings that enable auditing of an extensive set of cmdlets and keep the reports for extended periods.

How administrator auditing happens

Auditing is performed by the Admin Audit Log agent, which evaluates cmdlets as they are run against the audit configuration to decide whether the use of the cmdlet needs to be logged. If so, the agent creates an item containing details of the cmdlet in the Inbox of the audit mailbox. Table 11-7 lists the data that are captured in the audit reports.

TABLE 11-7 Data captured in audit reports

Field	Description
CmdletName	The name of the cmdlet that was executed
ObjectModified	The object the cmdlet was used to access (for example, a mailbox)
CmdletParameters	The parameters and values specified for the cmdlet
ModifiedProperties	The properties that were modified by the cmdlet
Caller	The user account that ran the cmdlet
Succeeded	True or False, to indicate whether the cmdlet succeeded
Error	Details of any error message that was generated
RunDate	Date and time when the cmdlet was executed in Coordinated Universal Time (UTC) format

The audit agent creates separate reports for each object if you execute an action that is performed against several objects. For example, if you use Get-Mailbox to fetch a list of mailboxes from a database and then use Set-Mailbox to set a new storage quota for each mailbox, the audit agent creates a separate report for each mailbox as it is updated.

You can also write your own entries into the audit log. For example, if you want to document a script being run or note a particular administrative operation you performed to solve a problem, you can capture it with the Write-AdminAuditLog cmdlet as shown here. You can insert up to 500 characters of text in the comment parameter, which is captured in the CmdletParameters property of the log entry:

```
Write-AdminAuditLog –Comment 'Server put into maintenance mode to prepare for the
deployment of Exchange 2013RTM CU3'
```

Only one audit mailbox is used for the organization. This can pose some difficulties in widely distributed organizations where actions performed in one part of the network might have difficulty being registered in the arbitration mailbox. Even in highly centralized environments, where a small set of administrators perform all actions for the organization, it is

possible to see errors caused by the unavailability of the database that hosts the arbitration mailbox, in which case Exchange cannot capture audit entries until the database becomes available again. While the database containing the audit mailbox is unavailable, Exchange writes event 5000 from the MSExchange Management Application source into the application event on the Mailbox server on which an attempt was made to run a command when it is unable to log an audit entry. To gain a full picture of what occurred for audit purposes, you might have to scan the Application Event Logs of all Mailbox servers.

Two methods exist to access administrator log entries:

- A set of generic audit reports cover the most common needs for an organization (as determined by the Exchange developers) is available through EAC (Figure 11-44). The reports cover both mailbox audit data and administrator audit data. The reports have not changed since they were first introduced in Exchange 2010.

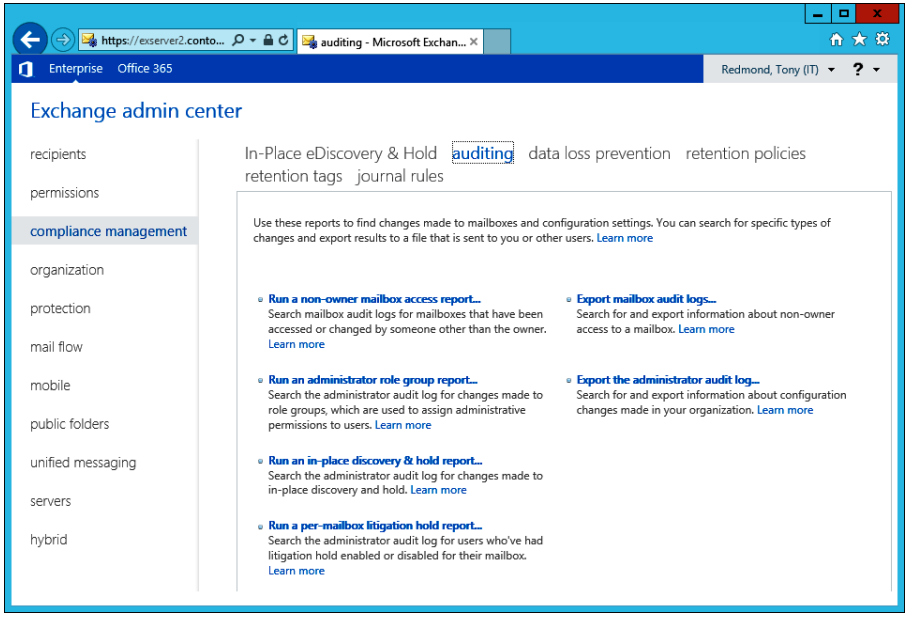


Figure 11-44 Audit reports available in EAC

- The Search-AdminAuditLog cmdlet is provided to enable administrators to create their own analysis of the administrator audit logs.

- The New-AdminAuditLogSearch cmdlet can be used to search administration logs in the background. The result is an email containing an XML attachment with the discovered audit entries that is delivered to specified recipients. Exchange limits the size of the XML attachment to 10 MB to ensure that it can be transmitted by most

messaging systems. This limitation might be a problem when acquiring audit information in large organizations.

The reports provided through EAC are reasonably sparse and don't cover all the information an auditor would conceivably want to extract from Exchange. For example, the litigation hold report is created by examining the audit log to pick out instances when administrators have enabled or disabled litigation hold for mailboxes and doesn't indicate why these actions were taken (because this information is not captured in audit logs). Often, you have to put the basic audit log information into context with information from other sources to arrive at a full picture of what happened, when, and why.

Interrogating the audit reports with the Search-AdminAuditLog cmdlet is straightforward. Here are a few examples that illustrate the possibilities that exist to discover what administrators are doing within the organization:

- Search for actions performed by one or more users. Each of the users you want to look for is identified by alias, email address, display name, or distinguished name. Separate the names with commas:

  ```
  Search-AdminAuditLog –UserIds Administrator, 'Tony.Redmond@contoso.com' |
  Format-Table Rundate, Caller, CmdletName
  ```

- Search for operations that have been performed against specific objects. These objects include mailboxes, databases, and connectors. For example, to locate audit entries for commands run against the DB1 and DB2 databases, use the command that follows. Note the use of CmdletParameters to output information about the parameters used when the commands were executed against the objects, which is often useful to understand what was done. The example output indicates what a mailbox move request was and shows that the database was the target for the move.

  ```
  Search-AdminAuditLog –ObjectIds "DB1", "DB2" | Format-Table Rundate, Caller,
  CmdletName, CmdletParameters
  ```

- Search for specific actions. In this example, you want to know who has recently mounted or dismounted mailbox databases. To locate the audit records, you specify the cmdlets that are used for these purposes. The ObjectModified property returned in each audit log tells you the name of the database that was operated on.

  ```
  Search-AdminAuditLog –Cmdlets Dismount-Database, Mount-Database | Format-Table
  RunDate, Caller, CmdletName, ObjectModified -AutoSize
  ```

- Search for audit records within a particular date range. In this case, you want to find out who has been creating new mailboxes over a specified period. You include the Succeeded property in the output because it is possible that some attempts to run the New-Mailbox cmdlet were unsuccessful, which is, in fact, what the results show.

```
Search-AdminAuditLog -StartDate "04/01/2013 00:00" -EndDate "04/15/2013 23:59"
-Cmdlets New-Mailbox | Format-Table RunDate, Caller, ObjectModified, Succeeded
```

- Analyze the cmdlets that administrators are running. These data are just for interest's sake, but they do reveal what the most commonly used cmdlets are.

```
Search-AdminAuditLog | Sort CmdletName | Group CmdletName | Format-Table Count,
Name -AutoSize
```

```
Count   Name
------- -------
     26 Add-DistributionGroupMember
      2 Add-MailboxPermission
      1 Dismount-Database
      1 Enable-Mailbox
      2 Mount-Database
      1 Move-DatabasePath
      1 Move-OfflineAddressBook
      4 New-ActiveSyncDeviceAccessRule
      1 New-ActiveSyncMailboxPolicy
      9 New-DistributionGroup
```

One small quirk of the Search-AdminAuditLog cmdlet is that it doesn't return the comments inserted into the audit log with the Write-AdminAuditLog cmdlet. Recall that the comments store the administrator-specified information they want to write into the audit log, so this is data you will want to be able to access. The data is held in the cmdlet parameters property of the audit entry, but if you include this property in the output set, all you will see is the string value, "comment." The data is in the audit log but must be extracted by directing the output of the Search-AdminAuditLog cmdlet into an array and then looking at the appropriate record in the array. For example, these commands create an array and then examine the cmdlet parameters data in the first record in the array:

```
$AuditArray = Search-AdminAuditLog -StartDate '11/1/2013 00:00' -EndDate '11/1/2013
23:59'
$AuditArray[1].cmdletparameters
```

Capturing audit data for administrator actions does not replace the need for operational discipline in the careful recording of changes made to server and organization configuration, including the following:

- Hotfixes and cumulative updates tested and applied, including any updates for add-on products
- Service packs for Windows and Exchange tested and applied
- Major network updates (for example, the introduction of a new Domain Name System [DNS] server)

- Installation of new Windows and Exchange servers
- Installation of any software on an Exchange server
- Changes to transport configuration such as the addition of a new connector or a change to transport settings

In this context, audit reports are helpful to record administrator actions, but they are not a complete solution. For instance, any changes made by Exchange objects through Active Directory tools such as ADSIEdit or Active Directory Users and Computers are not captured as audit entries.

Auditing mailbox access

Users gain access to mailboxes in different ways:

- Users log on and use their mailboxes as normal. Logically, Exchange calls this owner access.
- An administrator can delegate full access to a mailbox to another user. See Chapter 6, "More about the Exchange Administration Center," for more information on this topic. This is called delegate access. Users can also grant access through Outlook to individual folders in their mailbox such as the calendar or the Inbox.
- Administrators can grant themselves access to a user's mailbox and then log on to it. This is equivalent to delegating access because the same type of permissions are used to open and work with mailbox data.
- Administrators can run operations against a mailbox that can expose data. For example, an administrator can export the contents of a mailbox to a PST by using the New-MailboxExportRequest cmdlet or open a mailbox by using MFCMAPI. In these cases, the access is known to be privileged and is referred to as administrative access.

Normally, administrators do not concern themselves with what happens inside user mailboxes. However, some mailboxes contain sensitive information that might need to be protected against attempts to conceal or remove items that are required by the company, typically to justify actions the company or its employees took in a particular situation such as discussions with other companies relating to a merger, sale, or acquisition. Internally, sensitive information is often captured in discovery mailboxes that must be monitored to detect any attempt to interfere with the data.

Best Practice: Protect information by restricting access

Best practice in protecting information is to restrict access to people who can justify their access. You wouldn't open your CEO's mailbox to all, but the CEO might delegate access to her executive assistant. In the same way, if a human resources (HR) investigation uses a specific discovery mailbox to capture information relating to potential harassment of an employee, you grant full access to the discovery mailbox only to the members of the HR department who absolutely need that access. Another best practice is to always remove access from users as soon as they no longer can justify the access.

Mailbox auditing enables administrators to configure mailboxes so that Exchange captures details of specified actions. Audit entries are captured in the Audit subfolder of the Recoverable Items folder and can be interrogated with the Search-MailboxAuditLog or the New-MailboxAuditLogSearch cmdlets. Mailbox actions are divided into three categories:

- **Mailbox owner** It is not normal to audit user actions because users typically have full control over their mailbox contents. In addition, because mailbox owners use their mailboxes on a consistent and ongoing basis, the volume of audit entries is highest when auditing is enabled for the mailbox owner. For these reasons, when you compare the mailbox audit configuration for owners against the other categories, you see that the default list of audit actions for owners is blank.

- **Delegates** Other users who have been assigned delegate (FullAccess) permission can access a mailbox and take actions to affect its contents. It is possible for users who have SendAs access or other forms of delegated access assigned through Outlook to affect mailbox content, so these actions are captured, too.

- **Administrative operations** These are operations such as mailbox moves, mailbox imports from PST, and eDiscovery searches that administrators perform and that affect mailbox contents in some way, if only to open folders. In fact, because these operations can process a lot of data in a single command, they can compromise mailbox data in a wholesale manner if that is the intention of the permissioned user who initiates the command. Think of performing an eDiscovery search against executive mailboxes. This is perfectly acceptable if done within the context of a properly approved and overseen legal discovery action; it is unacceptable if an administrator launches a search and copies items from executive mailboxes to go on a fishing expedition in an attempt to discover the next acquisition target for the company.

Table 11-8 lists the various actions Exchange can audit for a mailbox. Setting an audit to None is supported for all types and disables any previous audit settings that were in place. You can see that a reduced set is available for owner auditing because it does not

make much sense to audit some actions (such as SendOnBehalf), and others (such as MessageBind) generate a huge number of logged actions.

The actions that are marked with an asterisk (*) are part of the default set of actions that are recorded when you enable auditing for a mailbox. For example, if you enable auditing for a mailbox, Exchange records details of all instances when a delegate sends a message using the SendAs permission. However, instances when delegates access the mailbox and send a message by using SendOnBehalf permission are not captured unless you specifically mark this action for auditing. The decision to include one action but not the other in the default set of logged actions is probably because a message sent using the SendAs permission represents a higher degree of impersonation than one sent using the SendOnBehalf permission. How to configure actions for auditing is discussed later in this section.

TABLE 11-8 Actions that can be audited for a mailbox

Action	Description	Admin	Delegate	Owner
Create	An item is created in the mailbox.	Yes*	Yes*	Yes
Copy	A message is copied to another folder in the mailbox or personal archive.	Yes	No	No
FolderBind	A mailbox folder is accessed (opened) by a client.	Yes*	Yes*	No
HardDelete	A message is deleted permanently from the database (removed from the Recoverable Items folder).	Yes*	Yes*	Yes
MessageBind	A message is opened or viewed in the preview pane.	Yes	No	No
Move	A message is moved to another folder.	Yes*	Yes	Yes
MoveToDeletedItems	A message is deleted and moved to the Deleted Items folder.	Yes*	Yes	Yes
SendAs	A message is sent from the mailbox by using the SendAs permission.	Yes*	Yes*	No
SendOnBehalf	A message is sent from the mailbox by using the SendOnBehalf permission.	Yes*	Yes	No
SoftDelete	A message is deleted from the Deleted Items folder (and moved to the Recoverable Items folder).	Yes*	Yes*	Yes
Update	The properties of an item are updated.	Yes*	Yes*	Yes

To reduce the number of audit log entries that are generated, Exchange consolidates the entries for FolderBind actions when performed by delegates. One audit entry is generated for each folder that is accessed within a three-hour period. If a delegate opens the Inbox in another user's mailbox 18 times within an hour, Exchange generates only a single audit entry to note that the delegate accessed the Inbox. In addition, the FolderBind and MessageBind actions are not logged for the default calendar.

INSIDE OUT **No message binding for delegates**

Mailbox auditing does not support the MessageBind action for delegates, which occurs when a message is opened or read. On the surface, this position is a little surprising because it prevents mailbox auditing from answering the "who read this message" question that sometimes occurs when users think that they never read a message but someone or something has obviously read the message. Suspicion then falls on the email administrators. The Exchange developers decided not to audit MessageBind actions for delegates because doing so would capture a very large number of audit entries, and a fear existed that so many audit entries would be created that they would overwhelm the RecoverableItemsQuota setting for a mailbox. (By default, this value is set at 30 GB.) It could be argued that it's more important to audit message access at the expense of having to increase RecoverableItemsQuota than not capturing any access, but that's not how the developers decided to proceed.

Enabling mailboxes for auditing

The first step in the process is to enable the mailboxes you want to audit by running the Set-Mailbox cmdlet. You cannot configure mailbox auditing with EAC. In this example, you enable auditing for the default discovery mailbox:

```
Set-Mailbox –Identity 'Discovery Search Mailbox' –AuditEnabled $True
```

You can then check that the audit setting is in place with the Get-Mailbox cmdlet:

```
Get-Mailbox –Identity 'Discovery Search Mailbox' | Format-List Name, Aud*
```

The act of enabling auditing for the mailbox means that the default set of actions against the mailbox is now audited. An AuditLogAgeLimit property is also present. This controls how long Exchange retains audit entries in the mailbox; the default value is 90 days. When audit entries expire, the MFA removes them from the mailbox the next time it processes the mailbox.

You can set the value of AuditLogAgeLimit to anything up to 24,855 days. This amounts to just over 68 years, which should be sufficient for even the most retentive administrators. Oddly, the actual coded maximum is 24,855 days, 3 hours, 14 minutes, and 7 seconds, which produces an audit age limit like this:

```
AuditLogAgeLimit : 24855.03:14:07
```

You can clear all existing audit entries by setting the AuditLogAgeLimit property to 00:00:00. If you do this, Exchange prompts you to confirm that all the entries should be deleted and proceeds if you confirm that this is what you want to do.

You can decide to include or exclude audit actions for administrators, delegates, or owners by writing the required actions in the AuditAdmin, AuditDelegate, and AuditOwner properties. For example, the owner of the default discovery mailbox folder will never log on to it, so you can set the audit settings for the owner to null. However, you might want to tweak the settings applied when administrators access the mailbox. Here's how you can make the change:

```
Set-Mailbox -Identity 'Discovery Search Mailbox' -AuditOwner $Null -AuditAdminUpdate,
Move, MoveToDeletedItems, SoftDelete, HardDelete, SendAs, SendOnBehalf -AuditEnabled
$True
```

To reverse the process and turn off auditing for a mailbox, set the AuditEnabled flag to $False as follows:

```
Set-Mailbox -Identity 'Discovery Search Mailbox' -AuditEnabled $False
```

TIP

It is possible to have a process that must access user mailboxes by using delegate or administrative rights. In the normal course of events, every loggable action these processes perform in the mailboxes generates audit events, all of which are perfectly acceptable and not suspicious in any way. In this situation, you might want to exclude the Windows account used for the process from mailbox auditing to prevent accumulation of a large number of interesting but useless audit entries. The Set-MailboxAuditByPassAssociation cmdlet excludes an account from mailbox auditing. For example:

```
Set-MailboxAuditByPassAssociation -Identity 'SomeFormofServiceAccount'
-AuditByPassEnabled $True
```

Accessing mailbox audit data

Mailbox audit information is written into the Audit subfolder of the Recoverable Items folder. However, this folder is invisible to any client, so you cannot just log on to the mailbox and browse through the audit entries. You can gain some insight from MFCMAPI, which shows details of the individual audit entries stored in the folder. For example, Figure 11-45 shows an entry for a SoftDelete operation. However, scanning audit entries with MFCMAPI is an exercise that is likely to end in frustration.

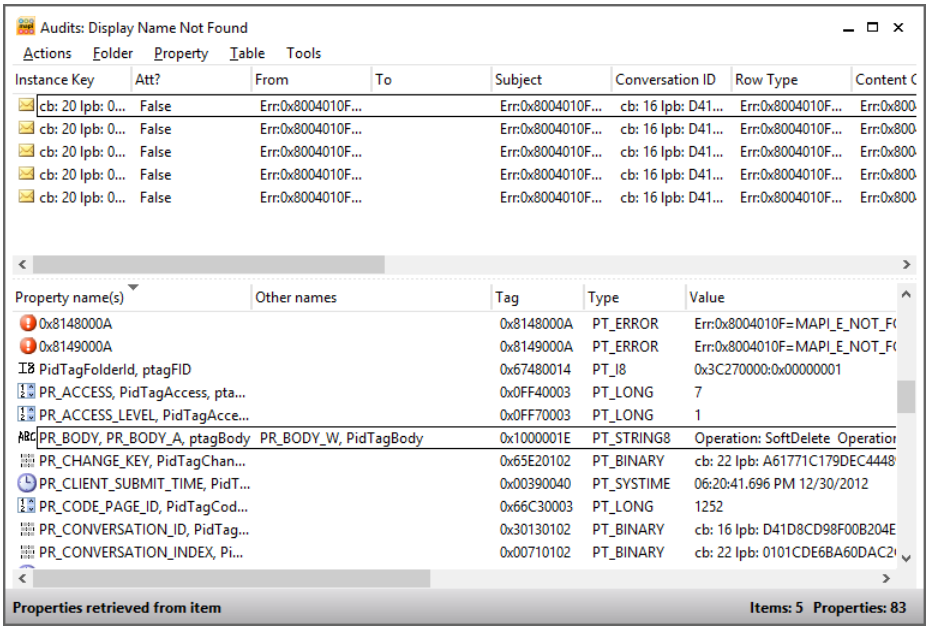


Figure 11-45 Examining mailbox audit log entries with MFCMAPI

Because the Audit subfolder is not exposed to clients, you have to submit a search of mailbox audit log data and have Exchange retrieve and display the found entries to you to interrogate audit data. Searches are performed in two ways:

- **Through EMS** The Search-MailboxAuditLog cmdlet performs a synchronous search for one or more mailboxes and returns the results on screen or by using the New-MailboxAuditLogSearch cmdlet to search across one or more mailboxes asynchronously in the background and return the results by email.

- **Through EAC** Search by using Export Mailbox Audit Logs in the Auditing section of Compliance Management. This option uses the New-MailboxAuditLogSearch cmdlet to fetch the desired information.

First, perform a simple search for audit entries for a single mailbox. In this example, you look for entries for a particular day so you can pass a start and end date. Select a number of fields to be output for each entry you find. Specify the –ShowDetails parameter to instruct Exchange to output details for each audit entry it locates and pass Delegate to the –LogonType parameter to restrict output to entries performed by a user who has delegate access to the mailbox. If you search many mailboxes for entries from an extended period, it is likely that Exchange will return thousands of entries. In this situation, you can use the –ResultSize parameter to specify how many entries you want to be returned. By default, Exchange outputs 1,000 entries.

```
Search-MailboxAuditLog -Identity 'Ruth, Andy' -ShowDetails -StartDate '5/12/2012
00:01'  -EndDate '5/18/2013 23:59' -LogonType Delegate -ResultSize 1000 | Format-
Table Operation, OperationResult, LogonUserDisplayName, ItemSubject, LastAcccessed
```

```
Operation      OperationResult  LogonUserDisplay     NameItemSubject      LastAccessed
---------      ---------------  ----------------     ----------------     -------------
SendOnBehalf   Succeeded        Executive Assistant  Travel Requests      11/05/2013 15:52:31
SendAs         Succeeded        Andrews, Lisa        Note from Peter      11/05/2013 15:54:41
SoftDelete     Succeeded        Smith, John                               11/05/2013 15:58:46
SendOnBehalf   Succeeded        Executive Assistant  Business directives  11/05/2013 16:05:09
```

This output is what you'd expect from a mailbox that has granted access to users to perform actions on their behalf, which you see in the SendOnBehalf and SendAs entries. Full access (to the complete mailbox or to a specific folder) is available to user John Smith because this user has been able to delete an item from the mailbox.

If you change the value passed to the –LogonType parameter to Admin, you see any operations performed against the mailbox as a result of administrative activity. For example, if a mailbox search is performed, you will probably see entries like this:

```
FolderBind       Succeeded         Administrator          11/05/2013 18:24:13
```

The New-MailboxAuditLogSearch cmdlet is designed to operate behind the scenes to fetch audit entries for perhaps many mailboxes on servers across the organization and respond with an email with an XML attachment that contains the search results. The XML data are complete but need to be poured through a formatter to make sense of them or, at least, to make sense for those who are not fluent in interpreting raw XML. The command to create a typical background mailbox audit log search looks like this:

```
New-MailboxAuditLogSearch -Name 'Unauthorized Delegate Access review'
-LogonTypes Delegate -Mailboxes 'CEO Executive Assistant', 'CEO@contoso.com',
'SeniorVPFinance@contoso.com' -StartDate '1/1/2012' -EndDate '12/31/2013'
-StatusMailRecipients 'ComplianceAuditMailbox@contoso.com'
```

Figure 11-46 shows how EMS acknowledges the submission of a new mailbox audit log search. The command that is run creates a search through the audit entries for delegate access that are stored in the three specified mailboxes between the start and end dates. If you don't specify any mailboxes, Exchange returns audit data for every mailbox on an Exchange 2010 or Exchange 2013 server in the organization that has been enabled for auditing.

```
Machine: Exserver2.contoso.com
[PS] C:\Windows\system32>New-MailboxAuditLogSearch -Name 'Unauthorized Delegate Access review' -LogonTypes Delegate -Mai
lboxes 'CEO Executive Assistant', 'CEO@contoso.com', 'SeniorVPFinance@contoso.com' -StartDate '1/1/2012' -EndDate '12/31
/2013' -StatusMailRecipients 'ComplianceAuditMailbox@contoso.com'


RunspaceId            : 99adafe5-fa1c-46b6-8295-7c227f4877ee
Mailboxes             : {contoso.com/Exchange users/CEO Executive Assistant, contoso.com/Exchange users/Michael Kerle,
                        contoso.com/Exchange users/Rob Young}
LogonTypes            : {Delegate}
ShowDetails           : False
ExternalAccess        :
Name                  : Unauthorized Delegate Access review
StartDateUtc          : 01/01/2012 00:00:00
EndDateUtc            : 31/12/2013 00:00:00
StatusMailRecipients  : {ComplianceAuditMailbox@contoso.com}
CreatedBy             : contoso.com/Users/Administrator
Identity              : AuditLogSearch\3f9691b0-bd8c-4bf0-a729-942b5ae87915
IsValid               : True
ObjectState           : Unchanged


[PS] C:\Windows\system32>
```

Figure 11-46 Running a mailbox audit log search with New-MailboxAuditLogSearch

EAC presents an easier-to-use alternative to specifying the details necessary to submit mailbox audit log searches, even if the data returned is in exactly the same XML format. To begin, open the Compliance Management section of EAC, select Auditing, and click Export Mailbox Audit Logs (seen earlier, in Figure 11-44). You then have the chance to enter the search criteria (Figure 11-47) before you click Export to initiate the search.

Exchange conducts mailbox audit log searches in the background, so don't expect to see the results in the destination mailbox quickly. In fact, searches are usually delayed until 22:00 local time or later to ensure that search processing doesn't affect other more important processes. Until this time, Exchange holds the search details in the MailboxAuditLogSearch folder of the search arbitration mailbox.

Exchange 2010 on-premises servers return mailbox audit log searches much faster; the change to delayed searches was first seen in Exchange Online. Delaying searches makes eminent sense in massive, multitenant infrastructures because you want to minimize the possibility that any action a tenant takes might have consequences for other tenants. The same logic has been taken forward into Exchange 2013 even though the case for such a change is far less obvious.

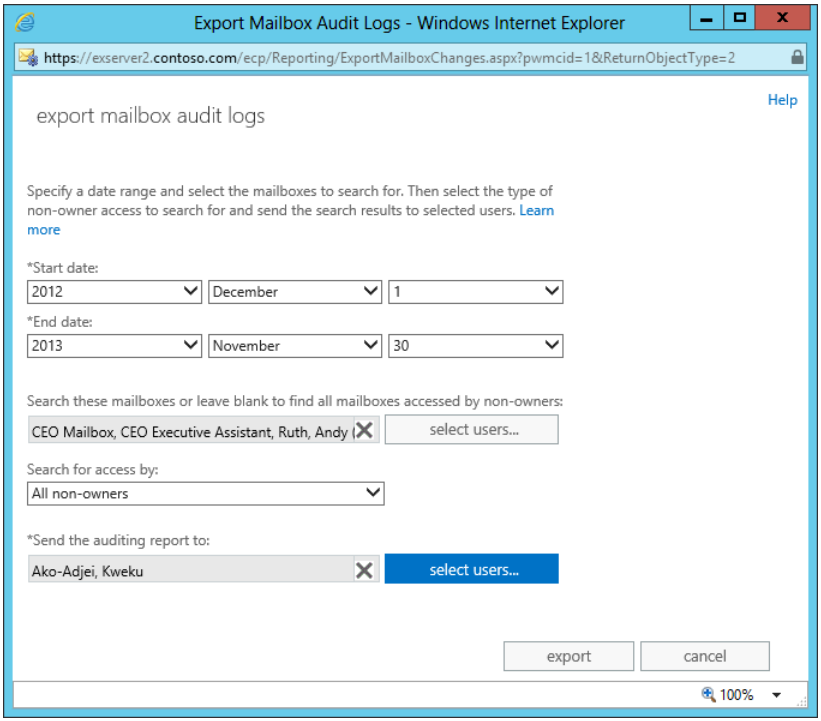


Figure 11-47 Defining mailbox audit log search options

When a mailbox audit log search using either EMS or EAC is complete, Exchange records the fact in event 4003 in the Application Event Log and sends an email containing the results to the email address or addresses specified in the –StatusMailRecipients parameter. The body of the message delivered to the user contains the search criteria used to examine mailbox audit data, and the actual audit data is contained in an XML format attachment. The recipients for mailbox audit reports must be mail-enabled objects known to the organization. Normally, they are mailboxes or groups, but you can arrange for the email reports to go to external recipients such as your auditors if you create a mail-enabled contact that contains their addresses.

No one could pretend that mailbox auditing is complete in terms of functionality or presentation. Forcing the majority of interaction through EMS would be acceptable if the output were easier to extract and interpret, but that's not the case. Eventually, Microsoft might integrate mailbox auditing more thoroughly into EAC and make this interesting and worthwhile functionality more accessible to administrators in a future version or service pack.

Other compliance features

Limited space halts this discussion about compliance, but many other features within Exchange 2013 can be considered when you build a compliance strategy. These features include the following:

- Active Directory Rights Management Services
- Outlook Protection Rules
- Message classifications
- Transport and journal rules
- Data loss prevention policies

Because no great changes have occurred in the first three of these topics since Exchange 2010, you can find details about their use and deployment in *Exchange Server 2010 Inside Out*, by Tony Redmond (Microsoft Press, 2010). The last two topics are covered in *Exchange Server 2013 Inside Out: Connectivity, Clients, and UM*.

CHAPTER 12

Public folders and site mailboxes

Public folders 765
Migration to modern public folders 787
Site mailboxes 791
Summarizing public folders and site mailboxes 804
Moving on 805

Collaboration means different things to different people. Email is one of the most basic and long-standing methods of collaboration known to the modern world. Today's collaboration landscape is remarkably different from the world that existed when Exchange 4.0 appeared. The web was fully functioning but hardly as pervasive as it is now. The thought that a social networking service such as Facebook would have a billion subscribers was far from the mind of an Exchange Server administrator given the task of installing Exchange public folders on her Windows NT 3.51 server. Progress has been made, and we are perhaps more collaborative now than at any time in the past if collaborative can be taken to mean "connected electronically." Public folders, the cockroaches of Exchange, persist in both modern and ancient forms and are joined by site mailboxes. The question is whether Exchange 2013 is any more collaborative than previous versions. Let's see.

Public folders

When they were first introduced, public folders were the cornerstone of the Microsoft collaboration strategy for Exchange. Many of the features public folders provide are very interesting, among which are support for electronic forms and the ability to replicate data to multiple replicas so that data would be close (in network terms) to end users. Time and technology have left public folders in their slipstream, and the features that seemed so attractive in the past are now relics of the way we used to work.

Even so, a great deal of public folders remain in use. For the most part, these public folders act as read-only repositories for documents that large numbers of users need to access, such as company policies and procedures, project documentation, and so on. Other solutions exist for such needs, but up to now, none has been able to offer the tight integration that exists between Microsoft Outlook and public folders. Compare the ease with which someone can publish a document in a public folder by dragging and dropping it from his personal folders (online or in a PST) to the gyrations that are often necessary to publish on a website. Given that so many people regard Outlook as their work hub, it's not surprising that public folders still remain in use. The advent of site mailboxes might force companies

to reassess public folders but only if they can afford to deploy Outlook 2013 and Microsoft SharePoint 2013, both of which are prerequisites for successful deployment of site mailboxes. Public folders are, after all, free because they are included in the Exchange base functionality. Modernizing part of their base functionality to bring public folders back into the Exchange mainstream, including making them available for the first time to Exchange Online, makes a lot of sense.

Modernization means that Exchange 2013 public folders are:

- Managed through the Exchange Administration Center (EAC).
- Stored in mailbox databases rather than in separate public folder databases.
- Protected by native high availability in Exchange rather than by using a completely different (and not always well-understood) replication mechanism.

Older public folders can still be present inside an Exchange 2013 organization, but only if they are already in use when an organization upgrades from Exchange 2010 or Exchange 2007. As discussed in the "Migration to modern public folders" section later in this chapter, the transformation of earlier public folders into the newer, mailbox-based variety takes a lot of planning and work. You cannot deploy the old-style public folders in a new Exchange 2013 organization in which public folders have never been used before. In addition, you cannot manage the old-style public folders through Exchange Administration Center (EAC). Instead, you use the separate Microsoft Management Console (MMC)–based public folder management console available in the Exchange toolbox or Exchange Management Shell (EMS).

New concepts

Older public folders are organized in a hierarchy that is split into system public folders and user-visible public folders. Although some system public folders remain in Exchange 2013 to support features such as the recovery of deleted public folders, modern system folders do not store user-accessible data such as the Offline Address Book (OAB) and Free/Busy information. The basic principles for the new architecture are as follows:

- Public folders are stored in public folder mailboxes in regular mailbox databases. Think of public folder mailboxes as containers for public folders similar to the way databases are containers for mailboxes and user mailboxes are containers for folders.
- A public folder hierarchy still exists, but only a single writeable copy of the hierarchy is maintained for the entire organization. The writeable copy of the hierarchy is stored in the first public folder mailbox created within the organization. You cannot change the mailbox that holds the writeable copy of the hierarchy.

- The public folder hierarchy is held as a set of items in the public folder mailbox. Each item is just a few kilobytes in size, meaning that even the hierarchy can be safely accommodated by a public folder mailbox.
- After the first public folder mailbox is created, all subsequent public folder mailboxes hold a read-only copy of the hierarchy. These copies are secondary to the primary, writeable copy and are updated by reference to the primary copy.
- Clients connect to their home hierarchy on their home site. The Client Access Server (CAS) is responsible for routing client connections to the correct home hierarchy.
- In addition to the folder hierarchy, public folder mailboxes also hold folder content. No replicas exist, and clients always connect to the single copy of content. Replication clashes caused by edits being applied to an item in multiple replicas as seen in the older implementation do not occur in the new public folders.
- Public folders are protected just as user mailboxes are by creating multiple copies of the databases that host public folder mailboxes in a Database Availability Group (DAG).

Because other collaborative solutions existed, organizations that began using Exchange in the recent past might not use public folders at all. Those that do can migrate their current public folder infrastructure to the new equivalent in a one-time migration operation. You cannot run old-style public folders alongside their modern equivalents. Migration is intended to be a one-time operation that moves the public folder hierarchy, content, and electronic forms over to populate a new set of public folder mailboxes and then switches client connections over to use the new public folder mailboxes.

Users whose mailboxes are on Exchange 2007 or Exchange 2010 servers cannot access modern public folders. You must migrate user mailboxes to Exchange 2013 before attempting to switch public folder implementations. To facilitate preparation, users whose mailboxes have been moved to Exchange 2013 can access public folders on Exchange 2007 or Exchange 2010 servers.

Creating public folder mailboxes

Assuming that you do not migrate data from old-style public folders and want to use public folders with Exchange 2013, you must create one or more public folder mailboxes and then build out the hierarchy with some new public folders. The first step is to create a public folder mailbox. Open the Public Folders section of EAC, choose Public Folder Mailboxes, and then click Create (Figure 12-1). Because this is the first public folder mailbox in the organization, it will hold the primary or writeable copy of the hierarchy.

Remember that you cannot change the mailbox that holds the primary copy of the hierarchy. Therefore, it is important for this mailbox to be protected by being placed in a database within a DAG that has multiple copies, mostly because if the mailbox holding the primary copy of the hierarchy is offline for any reason, users will not be able to post new items to public folders in that mailbox or create new public folders, administrators might not be able to perform some operations against public folders, and the synchronization of the public folder hierarchy to other mailboxes cannot occur. Users will still have view-only access to public folders stored in other mailboxes.

It is a good idea to follow a naming convention for public folder mailboxes so that the right mailbox can be located easily. For instance, you might decide to create public folder mailboxes on a departmental or geographical basis so that all the folders belonging to a particular department are held in a specific folder mailbox or to distribute the public folder load on a site basis. In this case, create a suitable folder to act as the base of your deployment and put it into a database that you know is part of a DAG so that the hierarchy is protected from the start.

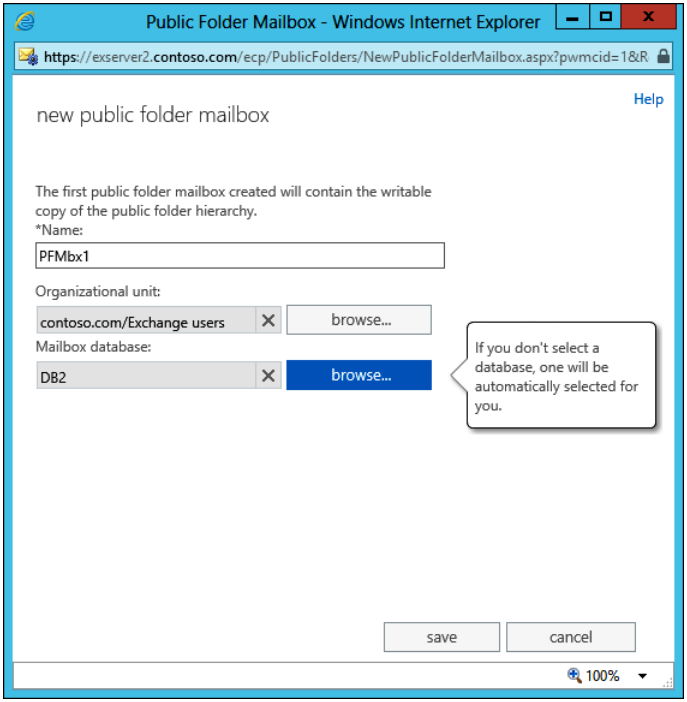


Figure 12-1 Creating a new public folder mailbox

Behind the scenes, Exchange creates a mailbox that is marked to be used for public folders and copies the public folder hierarchy to the new mailbox. All the properties you'd expect for a mailbox are present and can be managed with the Set-Mailbox command. Public folder mailboxes are hidden from address lists and have a disabled Microsoft Windows account. Although they have an email address, you cannot send messages to the mailboxes by using the address because public folder mailboxes are intended to store folders rather than content, so Exchange will reject the messages. To see the set of known public folder mailboxes with EMS, use the following command:

```
Get-Mailbox -PublicFolder
```

Note that *RecipientTypeDetail* is PublicFolderMailbox.

Follow the same process to create as many public folder mailboxes as you think you'll need (Figure 12-2). You can add public folder mailboxes as you need them, but some thought should be given to how to distribute public folders across mailboxes so that they are stored and accessed most effectively.

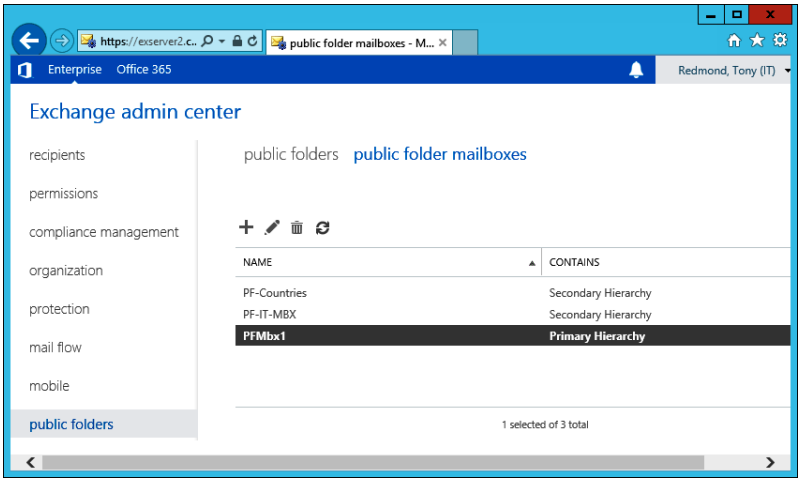


Figure 12-2 A set of public folder mailboxes

To see a list of the public folders that are held in a particular public folder mailbox, run the Get-PublicFolder cmdlet and specify the *ResidentFolders* parameter together with the name of the mailbox that you want to examine, For instance, this command generates a list of all the folders in the PFMBX1 public folder mailbox. Note the use of the *Recurse* parameter to request Exchange to return details of child folders.

```
Get-PublicFolder -Mailbox 'PFMBX1' -ResidentFolders -Recurse
```

INSIDE OUT **Knowing which public folder mailbox holds the writeable hierarchy**

As you can see from Figure 12-2, many public folder mailboxes can be active in an organization, but only one holds the writeable copy of the public folder hierarchy. EAC shows you which public folder mailbox contains the primary hierarchy, but you can also discover how EAC knows which mailbox holds the writeable copy of the public folder hierarchy by issuing this command:

```
Get-OrganizationConfig | Select RootPublicFolderMailbox
```

This returns the globally unique identifier (GUID) of the public folder mailbox. Note the GUID or capture it in a variable and then, to resolve the name of the mailbox that holds the primary copy of the hierarchy, run this command:

```
Get-Mailbox -PublicFolder GUID
```

Alternatively, you can run this command:

```
Get-Mailbox-PublicFolder | Select Name, ExchangeGuid
```

This returns a list of all the public folder mailboxes in the organization and shows their name and GUID. You can now compare the GUID returned by Get-OrganizationConfig to the GUIDs returned by Get-Mailbox. The mailbox in which a match exists is the one that holds the writeable copy of the hierarchy.

How many public folder mailboxes are needed?

A single public folder mailbox serves a complete organization only if the organization is small or makes limited use of public folders. When an organization grows beyond a few servers, or public folders are used extensively, thoughts turn to creating additional public folder mailboxes. The reasons you might want to use extra public folder mailboxes include:

- Spreading the client load across multiple mailboxes. If every public folder is stored in a single mailbox, all client connections will be directed to the database holding that mailbox. This is acceptable if you only need to use a few public folders, but it would not be feasible to operate a single public folder mailbox to hold 10,000 folders accessed by 100,000 users.
- Keeping data close to users. It makes sense to place public folders in mailboxes stored in databases whose copies are hosted on servers that are close (in network terms) to end users. For example, if you had a set of public folders that members

of the company located in Germany use, it would be better to host those folders in databases on servers located in Germany rather than in the United States.

- Keeping the public folder mailbox to a reasonable size. The total content of all the public folders held in a public folder mailbox count toward its overall size, which is constrained by the *ProhibitSendReceiveQuota* setting on the mailbox. When the limit is met, Exchange won't allow new items to be posted to the folders in the mailbox. As you create new public folder mailboxes, make sure the mailboxes are assigned an appropriate quota. On an ongoing basis, you should monitor the size of public folder mailboxes and arrange to move folders between mailboxes (first creating a new mailbox if required) if the size of any mailbox approaches the limit.

As part of the migration process to move older public folders to the new format, you have to determine how many public folder mailboxes should be used and how to assign the public folders to those mailboxes. (See the "Migration steps" section later in this chapter.) The same process should be undertaken for new deployments. Remember that you can start with a single public folder mailbox and introduce new mailboxes as required, moving folders between mailboxes to rebalance load or concentrate connections. Alternatively, you can start by creating a set of public folder mailboxes and distribute folders as they are created across the set of mailboxes. If any mailbox turns out to be hot because of the load that user demand creates, you can then redistribute folders across the other mailboxes.

INSIDE OUT The meaning of IsExcludedFromServingHierarchy

All public folder mailboxes include a copy of the public folder hierarchy. This allows clients to connect to any public folder mailbox and discover where the content is located. By default, Exchange load-balances client requests across available public folder mailboxes to distribute the load generated by hierarchy lookups. The IsExcludedFromServingHierarchy flag enables you to control whether a specific public folder mailbox services hierarchy lookups. If you don't want a public folder mailbox to be used for this purpose, run the Set-Mailbox cmdlet and set the flag to $True. However, you cannot stop a client from connecting to a specific public folder mailbox if the client's mailbox DefaultPublicFolderMailbox property directs it to connect to that mailbox.

INSIDE OUT **Synchronizing the public folder hierarchy**

Exchange automatically synchronizes the public folder hierarchy to all mailboxes that contain secondary copies at least once every 24 hours. Synchronization is performed using the Incremental Change Synchronization (ICS) mechanism by a mailbox assistant and happens every 15 minutes if clients connect to a mailbox that contains a secondary copy. You can force synchronization to occur by running the Update-PublicFolderMailbox cmdlet and check that synchronization is working by running the Get-PublicFolderMailboxDiagnostics cmdlet. For example, the following commands force synchronization for the PFMBX3 mailbox and then retrieve information about the last synchronization status:

```
Update-PublicFolderMailbox -Identity 'PFMBX3' -InvokeSynchronizer -FullSync
$Info = Get-PublicFolderMailboxDiagnostics -Identity 'PFMBX3'
```

Calculating public folder mailbox size

In terms of its current size, assigned quota, and maximum size, Exchange treats a public folder mailbox like any other mailbox. When a public folder mailbox is created, the default is to set the *UseDatabaseQuotaDefaults* property to $True; then Exchange applies the default quota settings used for the database in which the mailbox is created. Usually, these settings allow a mailbox to grow to a much smaller size than 100 GB. The default *ProhibitSendReceiveQuota* setting used in Exchange 2013 is 2.3 GB, so this quota governs the size of a public folder mailbox unless you set specific quotas for the mailbox or define new defaults for the database. For instance, to set new quotas for a public folder mailbox, you can run the Set-Mailbox command as follows:

```
Set-Mailbox -PublicFolder -Identity PF-IT-Mbx -UseDatabaseQuotaDefaults $False
-ProhibitSendReceiveQuota 50GB
```

In this case, setting UseDatabaseQuotaDefaults to $False tells Exchange to ignore the quota settings used for the database; setting ProhibitSendReceiveQuota to 50 GB sets the new quota.

You can determine the size of a public folder mailbox in exactly the same way as any other mailbox by running the Get-MailboxStatistics cmdlet. For example, to return the total number of items in the mailbox and their size:

```
Get-MailboxStatistics -Identity PF-IT-Mbx | Select ItemCount, TotalItemSize
```

To determine the number of items in an individual public folder and the size of those items, run the Get-PublicFolderStatistics cmdlet and pass the identifier for the folder. For instance:

```
Get-PublicFolderStatistics –Identity "\Departments\IT\Exchange 2013" | Select
ItemCount, TotalItemSize
```

Alternatively, to list all the public folders in a mailbox and report the size of the folder, run the same command but pass the name of the mailbox as a parameter:

```
Get-PublicFolderStatistics –Mailbox 'PF-IT-Mbx' | Format-Table Name, ItemCount,
TotalItemSize -AutoSize
```

When you run Get-PublicFolderStatistics against a mailbox, the command returns a list of all folders in the mailbox and the number of items contained in each folder. If a folder has recently been moved to another mailbox, you will see that some items are reported for the folder in the original mailbox. The Managed Folder Assistant removes items from the original mailbox when the deleted items retention period for the mailbox expires. (Note that the Managed Folder Assistant does not apply retention policies to public folder mailboxes.) In addition, if you run Get-PublicFolderStatistics against the mailbox that holds the primary copy of the hierarchy, it returns a complete list of all the folders in the hierarchy. If you run Get-PublicFolderStatistics without passing the name of a public folder mailbox, Exchange extracts information from all mailboxes and reports a complete picture of how many items exist in public folders no matter in which mailbox the folders are located.

If a public folder mailbox grows too large, you can split it with the *Split-PublicFolderMailbox.ps1* script that is available in the \v15\Scripts directory. The script takes the name of a public folder mailbox as the input and the name of another mailbox as the output and then attempts to move sufficient folders out of the mailbox to reduce the mailbox size to less than 60 percent of the assigned quota. The script is intelligent enough to move child folders with their parents. Folders are moved using the New-PublicFolderMoveRequest cmdlet described in the "Moving public folders" section later in this chapter.

INSIDE OUT How Office 365 controls public folder mailbox size

Office 365 tenants cannot create new public folder mailboxes because this is done for them automatically. An Office 365 background procedure checks the size of public folder mailboxes regularly, and when a mailbox exceeds 24.5 GB, the procedure attempts to move folders from the mailbox to other public folder mailboxes (if they exist) to rebalance storage across available mailboxes. If this is not possible, the procedure creates a new public folder mailbox and moves folders to it.

Controlling the root

After one or more public folder mailboxes have been created, but before you create some public folders, explore how to control the public folder root. Exchange 2013 follows the same naming convention as previous versions, meaning that the root of the hierarchy is \. Thus, a folder named \Projects is at the top level of the hierarchy, whereas a folder named \Projects\Training is the Training folder under the Projects top-level folder. This is referred to as the full folder path.

You can create as many top-level folders as you like, but experience demonstrates that it is wise to restrict folder creation to as small a number of people as possible by editing permissions for the root. Companies that ease the rules around public folder creation usually end up with a proliferation of folders, many of which are used and quickly discarded, contributing to a situation in which users literally can't find the information they need.

Users who are members of the Public Folder Management or Organization Management role group can manage public folders through EAC or EMS. This includes the ability to set public folder permissions and create folders. Most users will work with public folders through a client such as Outlook. The permissions granted through EAC or EMS determine which public folder management a user can perform through Outlook. To restrict top-level folder creation, open Public Folders and click the ellipses (... [More]) to expose the option to work with Root Permissions (Figure 12-3).

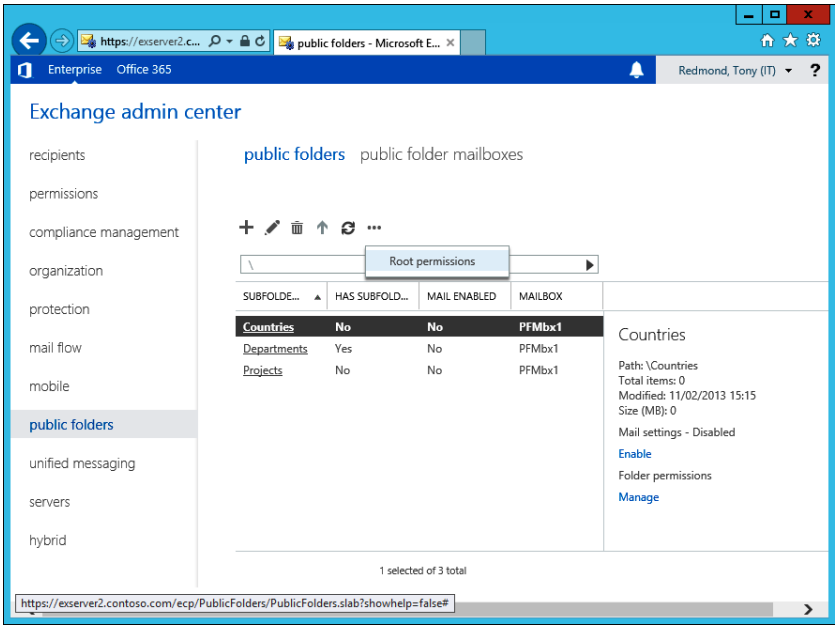


Figure 12-3 Selecting the option to work with the properties of the public folder root

When you edit the properties of the public folder root, you edit an object called the IPM_SUBTREE. This is the same name used in older public folders; it means the hierarchy of user-visible public folders. To assign a user sufficient permission to create top-level public folders, you browse through the Global Address List (GAL), select the user, and then assign her one of the predefined roles or create a custom role containing the desired set of permissions needed for the user to work with the folder (Figure 12-4). You can assign permissions to individual users or to mail-enabled security groups but not to normal distribution groups because these objects cannot hold Windows security principals.

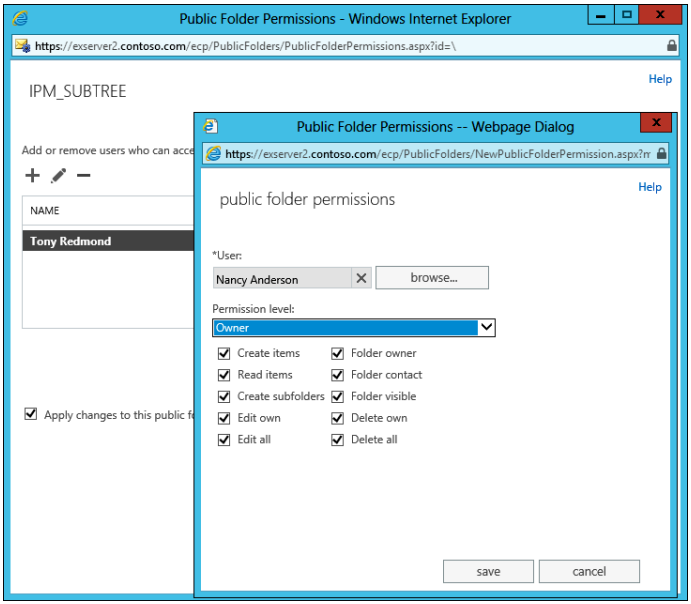


Figure 12-4 Assigning Owner permission for the public folder root to a user

Owner is an all-powerful role and should be restricted to people who manage the entire hierarchy; the Publishing Editor role is sufficient to create folders. If you assign this role to people for the root folder, they can create top-level public folders.

Note the Apply Changes To This Public Folder And Its Subfolders check box that can be seen to the far left of Figure 12-4. If selected, the permissions applied to the level of the hierarchy in which you are working will be applied to folders at lower levels. In this case, you are working with the root at the top level of the hierarchy, and the permissions that are assigned here will apply to every folder in the hierarchy, including folders that are created in the future. You probably will want to assign control over different parts of the folder hierarchy to different users. To maintain maximum control, assign permissions as the folder hierarchy is built out by creating new folders under the root.

INSIDE OUT **Public folders and content indexing**

Because public folders are now stored in mailboxes, you might assume that data held in public folders are indexed by Search Foundation and therefore available to discovery searches. In fact, this is partially true. Data is indexed, but discovery searches that Exchange executes do not interrogate this information when performing a search. This situation might well change in the future, but for now, public folder data is never uncovered by a discovery search.

Creating public folders

With the appropriate permissions in place to prevent the wholesale creation of public folders through Outlook, you can now lay out the structure of the public folder hierarchy and create the folders in which users will store content. For the purpose of illustration, I use a simple folder structure based on the departments that exist within the organization and create a single top-level folder for each department, which results in folders with names such as Sales, Marketing, Operations, and so on. The intention is to create subfolders under these folders to reflect the interests and needs of the people who work in the various departments. IT might have folders for major projects such as the deployment of Exchange 2013, whereas Sales might have folders for sales campaigns. The idea is that you create the basic skeleton of the hierarchy in a way that eases assignment of permissions over sections of the hierarchy to users who then control the content, which leads to the situation illustrated in Figure 12-5 in which the Analytics public folder is created under the Departments folder. Remember, an Exchange administrator never creates content in a public folder: administrators create and manage the hierarchy; users manage content.

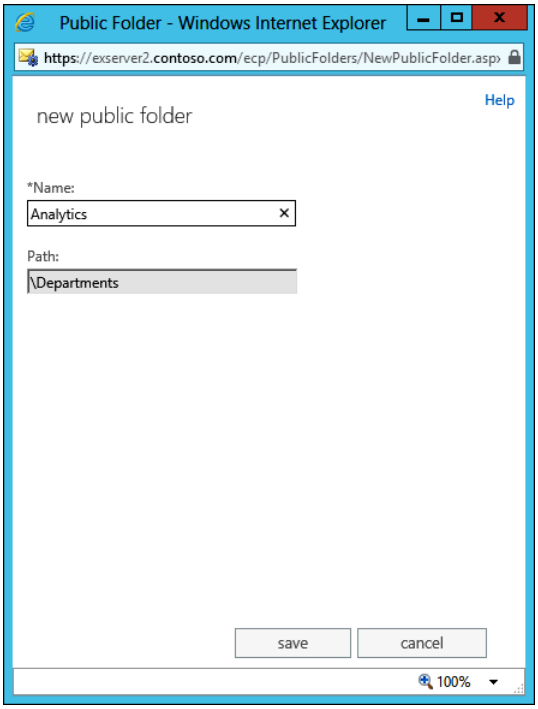


Figure 12-5 Creating a new public folder

Creating a new public folder with EMS is done with the New-PublicFolder command. For instance, here's how to create a public folder called Administration under the Departments folder:

```
New-PublicFolder -Name 'Administration' -Path '\Departments'
```

To be more precise, pass the name of the mailbox in which you want Exchange to create the folder; otherwise, the folder will be created in the mailbox that holds the primary copy of the hierarchy. Of course, you can move the folder afterward, but it is better to put it in the correct place from the start. A better command is therefore something like this, which puts the new folder in the PF-Departments public folder mailbox:

```
New-PublicFolder -Name 'Administration' -Path '\Departments' -Mailbox
'PF-Departments'
```

Figure 12-6 shows a set of public folders in EAC, including those EAC creates and the one that was created previously with EMS. You can also see that the IT folder is slightly different because it is stored in the PF-IT-MBX mailbox, whereas all the other folders use the PFMbx1 mailbox.

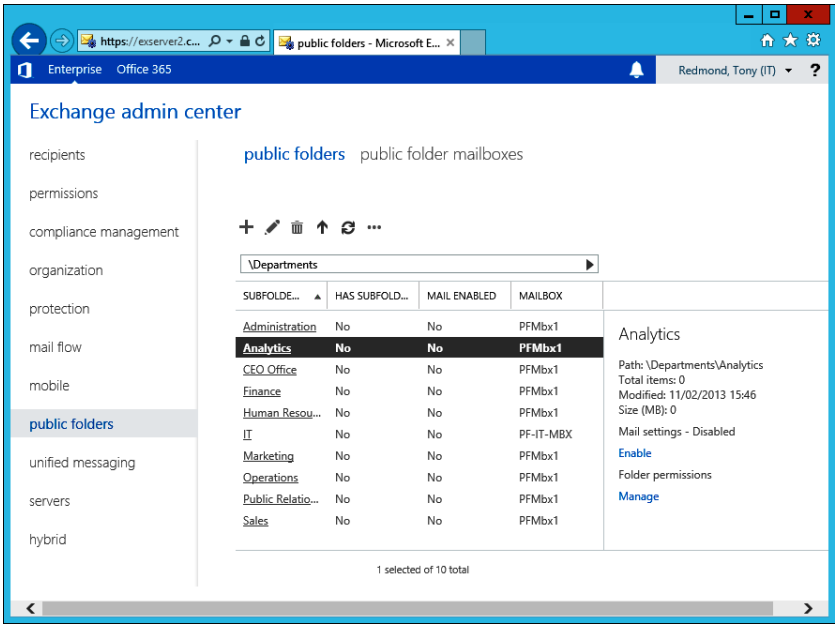


Figure 12-6 A set of public folders as shown by EAC

These public folders are empty and will remain in this state until users begin to populate the folders with content. Exchange does not support the ability for an administrator to import information into a public folder in the same way that you can import data from a PST into a user mailbox, so you rely on users to create content in the public folders through a client. Figure 12-7 shows the set of public folders that you've created as viewed through Outlook 2013. Users can post items to the folders, email items to mail-enabled public folders, or drag and drop items from other folders into the public folders.

You can also access public folders, but only modern public folders, through Outlook Web App. This client takes a different approach than Outlook does; you have to add the public folders with which you want to work to your *Favorites* (the list containing folders such as Inbox and Sent Items). Then, you can access content in the public folders that you have chosen or post new items. The approach Outlook Web App takes might seem surprising because every other client that has ever accessed public folders has allowed users to browse through the complete hierarchy. The argument is that it is more efficient to select the folders you want to access and be able to select them quickly through Favorites than having to open and navigate down what could be a very large and complex hierarchy to arrive at the desired public folder. In addition, if you add a public folder to your Favorites with Outlook Web App, Outlook also displays it as a Favorite (and vice versa) because the two clients share a common set of Favorites.

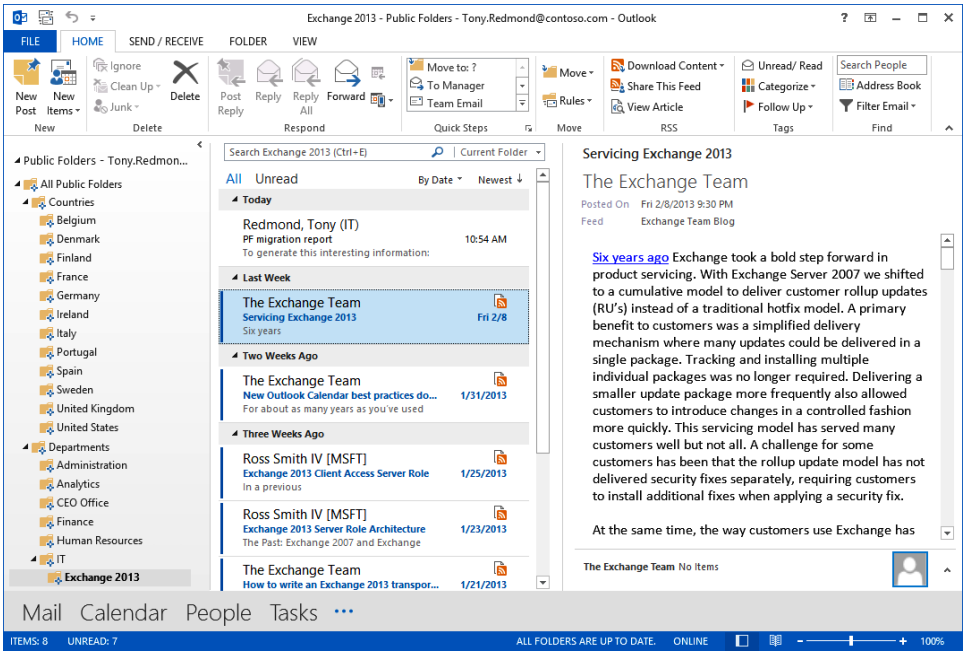


Figure 12-7 Exchange 2013 public folders accessed from Outlook 2013

Mail-enabling public folders

By default, public folders are not mail-enabled; users cannot add content by including a public folder as an addressee for a message. In fact, one of the most common uses for public folders is as a repository to capture the messages sent to distribution groups that would otherwise be unavailable to people who are not part of the group or who join the group after a discussion has occurred.

Mail-enabling a public folder is easy. Select the folder in EAC and then click the Enable link under Mail Settings in the details pane (Figure 12-8). EAC prompts you to confirm the action. If confirmed, Exchange populates a set of mail-related properties for the public folder so that the folder can participate in message transport.

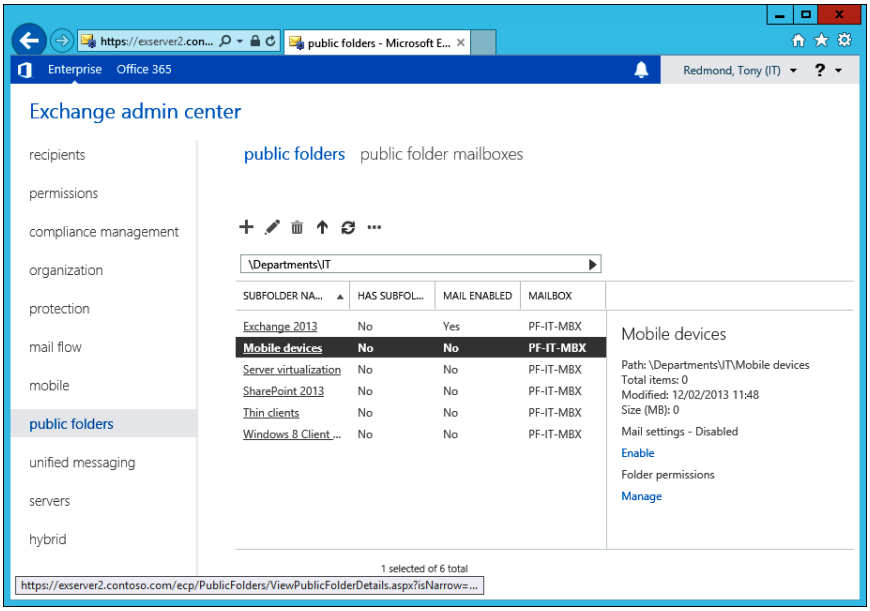


Figure 12-8 Mail-enabling a public folder

The Get-PublicFolder command does not reveal all the properties for a mail-enabled public folder. Instead, use the Get-MailPublicFolder command to retrieve the properties and Set-MailPublicFolder to update the properties. To show the kinds of properties that are populated when a public folder is mail-enabled, here's an edited version of the output from Get-MailPublicFolder. Apart from the ContentMailbox property (pointing to the public folder mailbox that contains the content for the folder), these properties look very similar to what you'd see for a user mailbox:

```
ContentMailbox                        : contoso.com/Exchange users/PF-IT-MBX
AcceptMessagesOnlyFrom                : {}
AcceptMessagesOnlyFromDLMembers       : {}
AcceptMessagesOnlyFromSendersOrMembers : {}
AddressListMembership                 : {\MailPublicFolders(VLV), \All
Recipients(VLV), \Default Global Address List,\Public Folders}
Alias                                 : Mobiledevices
OrganizationalUnit                    : contoso.com/Microsoft Exchange System
Objects
EmailAddresses                        : {SMTP:Mobiledevices@contoso.com}
GrantSendOnBehalfTo                   : {}
HiddenFromAddressListsEnabled         : False
LegacyExchangeDN                      : /CN=Mail Public Folder/CN=Version_1_0/
CN=e66f783b-adf3-48a7-bed3-bb74f5c79be2/CN=xxxx
MaxSendSize                           : Unlimited
MaxReceiveSize                        : Unlimited
```

In addition, mail-enabled public folders are represented as Active Directory objects in the Microsoft Exchange System Objects organizational unit (OU), as shown in Figure 12-9. This allows the public folders to be found when Active Directory is consulted to know how best to route a message for delivery.

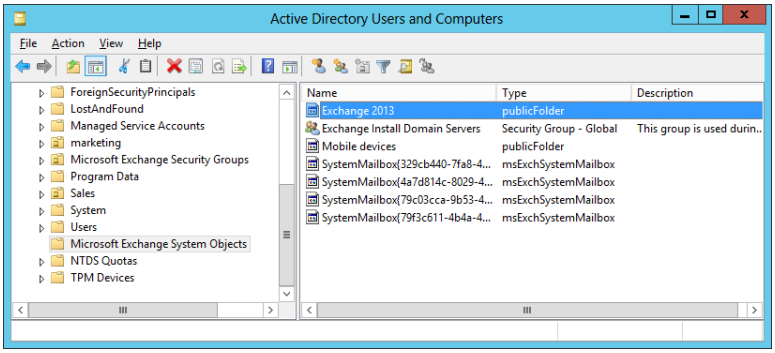


Figure 12-9 Mail-enabled public folders in Active Directory

Read through the properties listed previously and note that public folders are included in some address lists and are exposed to users when they browse the GAL. In Figure 12-10, you can see the entries for two mail-enabled public folders. Users can send messages to the public folders by including them in the set of addresses. However, there's no real need to reveal the presence of mail-enabled public folders in the GAL unless you really want to. The normal approach is to include the mail-enabled public folder as a recipient in the distribution group for which it will serve as a repository and then hide the public folder from address lists. A command similar to the one shown here will do the job:

```
Set-MailPublicFolder -Identity "\Departments\IT\Exchange 2013"
-HideFromAddressListsEnabled $True
```

Alternatively, select the folder with EAC and click the pencil icon to reveal all the properties with which you'd normally expect to work for a user mailbox. Figure 12-11 shows that this folder is a member of a distribution group, so a copy of every message posted to the members of the group will be delivered to and available in the public folder.

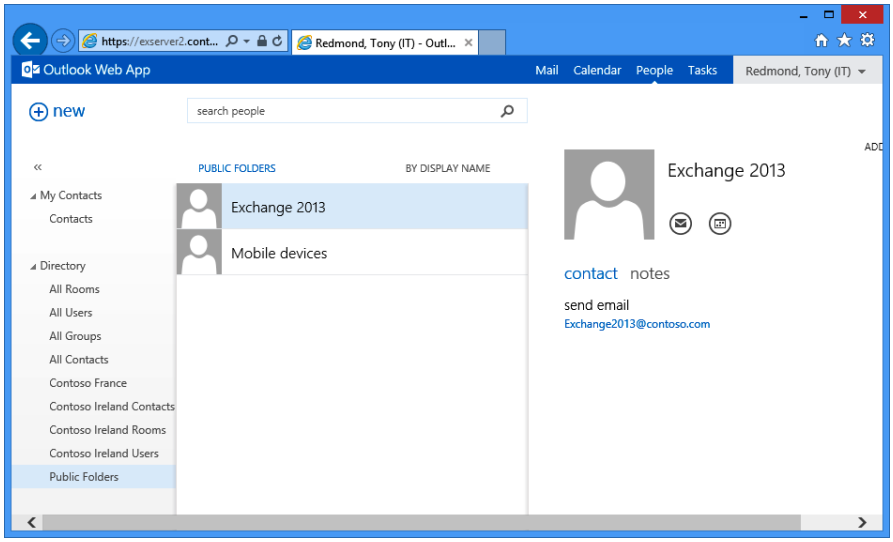


Figure 12-10 Mail-enabled public folders shown in the GAL

The Delivery Options section of a mail-enabled public folder's properties permits an administrator to enable other users to Send On Behalf Of the folder or Send As the folder. In the past, public folders were often used as the source of service-type messages such as those generated to inform a user when something was done on his behalf (for example, granting him a larger mailbox quota). Shared mailboxes are more popular now, but the facility still exists to use a public folder for this purpose.

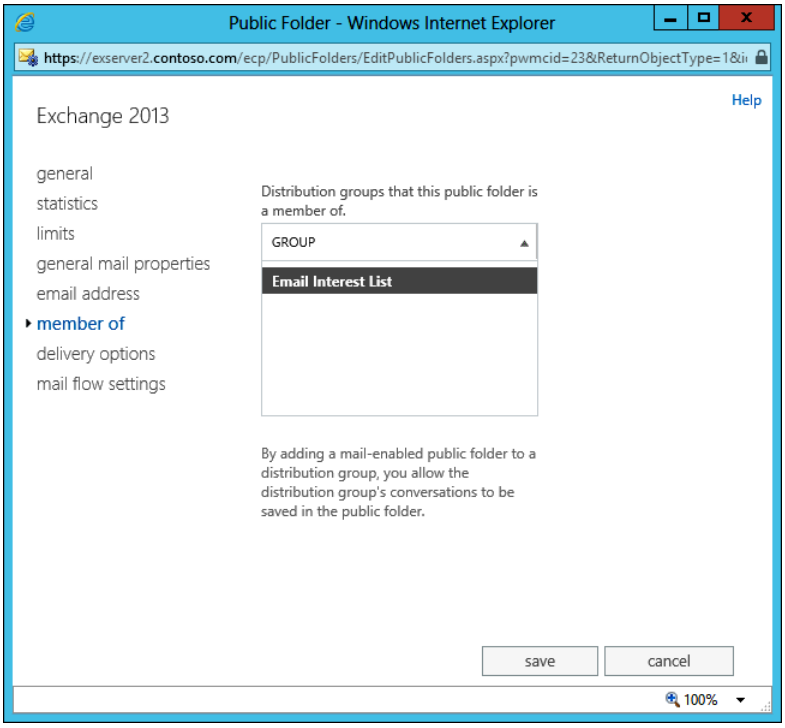


Figure 12-11 Viewing the properties of a mail-enabled public folder

Moving public folders

When you create a new public folder, Exchange automatically directs client-created content for the newly created public folder in the public folder mailbox that contains its parent folder unless you specify the name of the public folder mailbox to host the new folder by passing the –Mailbox parameter to the New-PublicFolder cmdlet. When you use EAC to create a new public folder, you cannot specify the mailbox in which the new folder is to be created, so the new folder is always created in the same mailbox as its parent. In addition, if only one public folder mailbox is in use, the new folder is created there.

It can be preferable to exert more control over the mailbox that stores a public folder. For example, if you create a new public folder that is intended for use by a group of people located in a particular country, it is a good idea to create a public folder mailbox in a database in a DAG or server that is usually close to those users instead of requiring them to make extended network connections whenever they need to access content.

Two methods are available to move public folder content. These are:

- Use New-PublicFolderMoveRequest to move the folder from one public folder mailbox to another. This is a permanent movement that is comparable to moving a user mailbox from one database to another. The move is performed by the Mailbox Replication service (MRS) and occurs using the same framework as exists for mailbox moves, so some delay occurs between scheduling the move request and its completion. In addition, MRS only processes a single public folder move job at a time, which could cause further delay. This is the preferred method when content already exists in the folder.

- Use Set-PublicFolder with the OverrideContentMailbox parameter to redirect content creation from the current public folder mailbox to another. The Store executes a redirect immediately. Using this method is acceptable when you need to redirect client connections to a different public folder mailbox and it is not important to retain any content that exists for the folder in its current public folder mailbox.

The New-PublicFolderMoveRequest command works in a very similar manner to normal mailbox moves. One difference is that you can provide a list of folders to move in a single operation. This is important because if you attempt to move a public folder that has child folders, the child folders will not be moved if you specify only the name of the top-level folder. In other words, you have to tell Exchange exactly what folders should be moved to the target public folder mailbox. For example:

```
New-PublicFolderMoveRequest -Folders "\Countries\Germany", "\Countries\Italy"
-TargetMailbox PF-Countries -BadItemLimit 10 -Priority Highest
-SuspendWhenReadyToComplete
```

This command creates a job that moves the public folder content to the target public folder mailbox and then halts to allow the administrator to check that everything went well. The administrator can then allow the move to progress to completion by running the Resume-PublicFolderMoveRequest command.

Only one public folder move job can be processed at a time, so MRS will refuse to accept a new job if one is already present. You can use the Get-PublicFolderMoveRequest command to check whether a job is already running before you attempt to schedule another and the Get-PublicFolderMoveRequestStatistics command to see what's happening with the job if one is present. ("\PublicFolderMove" is the identity assigned to these jobs.)

```
Get-PublicFolderMoveRequestStatistics -Identity "\PublicFolderMove" | Format-List
```

Like mailbox moves, public folder moves can complete only if sufficient resources exist and system health allows. For example, if the target public folder mailbox is in a database that is part of a DAG, the replication status within the DAG must permit the move to complete or it will be stalled due to a high-availability constraint.

Public folder move requests are cleaned up automatically after 30 days. If you don't want to wait because you want to move some folders, assuming that the job is complete, you can remove the current public folder move request by running the following command. When the old job is removed, you will be able to start a new public folder move.

```
Remove-PublicFolderMoveRequest –Identity "\PublicFolderMove"
```

Note that child folders are not moved unless you explicitly instruct MRS to include these folders. The easiest way to do this is to run the Move-PublicFolderBranch.ps1 script that is found in the \Scripts directory.

Redirecting content

When you run the Set-PublicFolder command to override the current destination for new client content, you specify the name of the target public folder mailbox in the *OverrideContentMailbox* parameter. This command redirects all future client connections to the public folder mailbox 'PF-IT-MBX':

```
Set-PublicFolder –Identity '\Department\IT' –OverrideContentMailbox 'PF-IT-MBX'
```

Any new content that clients create will now be stored in the PF-IT-MBX public folder mailbox. This command does not affect any subfolders, which remain in the original public folder mailbox. If you make a mistake, reverse course by running the command to redirect back to the original public folder mailbox.

Moving a folder to the mailbox that will serve as its permanent home is always preferable. However, it might be that the mailbox that hosts an essential folder is unavailable for some reason, and the need exists to be able to create new content in the folder. In this situation, you could redirect new content to a different mailbox during the period of the outage. When the other public folder mailbox comes online again, you can redirect the folder back to its original location.

It is critical to realize that a redirection does not move content. Thus, when you redirect a public folder to another mailbox, any content that existed in the folder is left in the original mailbox, where it will be retained for 14 days. Users will lose access to the content that remains in the original mailbox. For this reason, before you redirect any public folder, make sure that you take a copy of the existing data so that it can be imported into the new location after the redirection is effective. You can do this by copying the items to a PST.

Two other situations exist when you might want to redirect public folder content. First, you recover a soft-deleted public folder mailbox, and you want to restore the folders contained in the mailbox to an active public folder mailbox. In this scenario, you first restore the content by using the New-MailboxRestoreRequest command and then use Set-PublicFolder –OverrideContentMailbox to redirect any new content for the folders to the appropriate public folder mailbox.

Second, you recover information from a public folder mailbox that is in a recovery database. Use the same steps as taken in the first scenario to restore the information.

Removing a public folder

Removing a simple public folder is easy. From EAC, select the folder and click the wastebasket icon. Confirm when prompted, and the folder and any subfolders will be removed. The equivalent EMS command is something like this:

```
Remove-PublicFolder -Identity "\Countries\Italy"
```

Removing a mail-enabled public folder is slightly more complicated because it exists in Active Directory and in its host public folder mailbox. A two-step process is needed. First, you disable the mail-enabled folder. Second, you remove the public folder. To remove a mail-enabled public folder by EMS, you run these commands:

```
Disable-MailPublicFolder -Identity "\Departments\Administration" -Confirm:$False
Remove-PublicFolder -Identity "\Departments\Administration" -Confirm:$False
```

When a public folder is deleted, it is moved to a hidden system folder under the root *\Non_IPM_Subtree\Dumpster_Root,* where it is kept until its deleted retention period expires (normally 30 days). While it is still available, you can recover a deleted public folder by moving its path back to its original point in the public folder hierarchy. For instance, to recover the folder deleted by the commands previously shown, you can search the system folder to locate the identity of the folder you want to restore:

```
Get-PublicFolder -Identity "\Non_IPM_Subtree\Dumpster_Root" -Recurse | ? { $_.Name
-eq "Administration" } | Select Identity
```

The identity returned will be a value such as *"\NON_IPM_SUBTREE\Dumpster_root\cd0788e8-ae5b-432c-a2a6-9095580bbc2d\Administration"*. You can use it to recover the public folder with:

```
Set-PublicFolder -Identity "\NON_IPM_SUBTREE\Dumpster_root\cd0788e8-ae5b-432c-a2a6-
9095580bbc2d\Administration" -Path "\Departments"
```

The commands are straightforward if not obvious. Hopefully, you will not have to use this procedure often.

Mobile access

All the client protocols Exchange supports to enable mobile devices to synchronize (POP3, IMAP4, or ActiveSync) focus exclusively on user mailboxes and take no account of collaborative resources such as public folders or site mailboxes. If users need to access content from these repositories on their mobile devices, they have to copy information from the repository to a folder in their mailbox and then synchronize that folder. The exception is the Outlook Web App for Devices app, first released for iOS in July 2013. This app is scheduled to support public folders in a future version.

Organizational forms library

An organizational forms library is a repository for electronic forms associated with public folders that need to be accessible by all users. Public folders have used electronic forms since their inception to enable users to enter and access data that is stored in items in the folder. For example, a form might be used when someone wants to request the creation of a new mailbox. A single organizational forms library can be created for each language used in the organization. The language is identified using the standard locale codes as in en-us for U.S. English, fr-fr for French, and so on. Most organizations attempt to use just one language for electronic forms because this simplifies form creation and maintenance.

Exchange 2013 does not include an organizational forms library. If you need to use one, it has to be created using EMS as follows:

1. Create the new public folder to contain the organizational forms library.

   Note that the public folder is created under Non_IPM_Subtree, meaning that it is intended for system use and will not be listed alongside other public folders.

   ```
   New-PublicFolder –Name "Organizational Forms Library" –Path "\Non_IPM_Subtree\
   Eforms Registry"
   ```

2. Set the proper locale for the new public folder. In this instance, you set the locale to be U.S. English:

   ```
   Set-PublicFolder -Identity "\Non_IPM_Subtree\Eforms Registry\Organizational
   Forms Library" –EformsLocaleID en-us
   ```

3. Add access rights to the organizational forms library.

   General users need to be able to read the forms held in the folder; those who maintain the forms need permission that is more elevated. This command assigns the Author role to the members of a security group:

   ```
   Add-PublicFolderClientPermission -Identity "\Non_IPM_Subtree\Eforms Registry"
   -User 'Forms Developers' -AccessRights Author
   ```

4. Repeat to create the organizational forms library for other languages as required.

Migration to modern public folders

Migrations tend to be complex affairs. Migration from old-style to modern public folders is unlikely to be simple for many organizations, especially when the hierarchy is large. It is a one-time cutover in that after you perform the switchover to modern public folders, it is difficult if not impossible to revert without losing data because no synchronization facilities exist between one type of public folders and the other.

Microsoft includes a set of scripts with Exchange 2013 to assist in the migration of content from public folder databases. It has also documented an approach to the migration that you can read at *http://technet.microsoft.com/en-us/library/jj150486.aspx*. This is a useful framework on which to build a customized migration plan for your organization after you do some up-front planning. However, it's almost inevitable that new and valuable techniques will be discovered that help perform public folder migrations more effectively or quickly, so it's always wise to read available sources of information to establish the current state of the art before you start.

No migration can be successful if you do not possess full information about the data that has to be moved. Accordingly, some basic questions have to be answered, including:

- What business use exists for public folders today? Is the correct go-forward decision to move to modern public folders, or should another solution be used? For instance, site mailboxes are a better choice for document-centric collaboration, albeit at the expense of requiring an investment to deploy SharePoint 2013 and Outlook 2013.
- What applications (electronic forms) are used with public folders? Are all these applications still required?
- How many public folders exist in the hierarchy?
- How much data (items, size) are held in the public folders? How old is this data?
- Who manages the data and permissions for public folders? How long is the data retained, and what arrangements are in place to maintain (clean up) the data? This information is necessary to set the retention period for the new public folder mailboxes.
- How many public folder databases are in use, and why are they in use? For example, do large populations of public folder users exist in particular locations that depend on a local public folder database?

You can answer some of the more fundamental questions by running the AggregatePFData.ps1 script (available on Exchange 2010 servers) to provide an overview of the current public folder infrastructure, including the names of folder owners, the date a folder was last accessed, whether a folder is mail-enabled, and how many items it contains. It is a good idea to run the script and then review the output to cull unwanted and obsolete public folders from the hierarchy before migration begins. If a need exists to retain information from an obsolete public folder, the data can be copied to a PST before the folder is removed.

Before commencing any migration, you should move user mailboxes to Exchange 2013. Exchange 2013 users can still access old-style public folders, but the reverse is not true because Exchange 2010 and Exchange 2007 mailboxes do not understand the new format.

After rationalizing the current public folder infrastructure, the migration process can begin. The recommended steps are:

1. Take a snapshot of the current public folder infrastructure. You can do this by running the AggregatePFData.ps1 script or by following the steps outlined in TechNet. The purpose of this step is to have a baseline that you can use for post-migration comparison to ensure that all the necessary information has been moved across to modern public folders.

2. Prepare Exchange 2010 by removing any trace of a previous migration attempt. Do the same for Exchange 2013. This means that you remove any public folder migration jobs and any public folders that you previously moved in an incomplete migration. In other words, you create a clean environment for the migration to commence.

3. Run the Export-PublicFolderStatistics.ps1 script on Exchange 2010 to generate a file containing a list of public folder names and their sizes.

4. Run the PublicFolderToMailboxMapGenerator.ps1 script to read the set of public folder names and map them against a set of public folder mailboxes. One of the parameters for the script is the maximum mailbox size (in bytes), so you have some control over the number of public folder mailboxes that will be used. The aim is to distribute public folders so that the public folder mailboxes take equal load. This is done purely by folder size rather than by taking account of user demand, so you might have to move public folders between mailboxes afterward.

5. The output from the script is a CSV file. However, Exchange does not read the CSV file and create the necessary public folder mailboxes for you. Instead, you use the contents to know how many public folder mailboxes you have to create. You then create the mailboxes and place them in appropriate databases. These public folder mailboxes are created with the *HoldForMigration* parameter set so that (as the name implies) they are reserved for migration purposes and won't be exposed to clients.

6. The CSV file is now used as input to a public folder migration job that the Migration service runs to move content from the old public folders to the new. Make sure that the contents of the CSV file reflects the names of the public folder mailboxes that you created. Microsoft suggests that the migration will proceed at between 2 GB and 3 GB per hour, depending on system resource availability and other load. The migration job, which uses the New-PublicFolderMigrationRequest command (similar

in concept to the New-MigrationBatch command to move batches of mailboxes), copies all the content from the source folders that is in place when the migration starts to the public folder mailboxes and then holds the migration.

7. After checking the migration job report and making sure that everyone is ready to make the switchover, lock down the old public folders by running the Set-OrganizationConfig command to put the organization into locked mode. The flag used for this purpose is replicated through the organization, and clients cannot add new content to public folders at this time.

8. Release the suspended state of the public folder migration job. The Migration service performs an incremental synchronization of the source public folders and moves any new content it finds to the public folder mailboxes.

9. Take a snapshot of the data in the public folder mailboxes and compare it to the snapshot taken prior to the migration to assess whether all the folders and data have been migrated successfully.

If the migration is complete, run the Set-OrganizationConfig command to set the *PublicFolderMigrationComplete* flag to $True. The CAS now redirects users to public folder mailboxes instead of to the old public folder databases.

If more than a few days elapse between when the Migration service copies over data in a migration job (step 6) and when the cutover occurs, you can perform incremental synchronizations to ensure that minimal data needs to be copied from the old public folders during the finalization process (step 7 onward). An incremental public folder synchronization is performed by running the Resume-PublicFolderMigrationRequest cmdlet, which then copies any new data and updates that have occurred in the old public folders since the initial synchronization occurred. You can run as many incremental synchronizations as are necessary to keep the two sets of public folders in steps pending the cutover.

Only the finalization stage, when the cutover actually occurs, requires downtime during which public folders will be unavailable to users. However, the ideal situation is to perform public folder migrations over a weekend or holiday period when it is less likely that users will need to access public folders, and administrators can take the necessary time to prepare, execute, validate, and complete the migration. It is also valuable to run a practice migration by using a test environment that contains a complete copy of the production public folder environment because this gives real-life guidance about the effectiveness of the procedure and the time the migration is likely to require.

Site mailboxes

Everyone will have his own definition of what collaboration means and how this can be achieved best within Exchange. Some believe that email (still the collaborative application par excellence) is good enough, provided it is used well. Others consider public folders capable of satisfying the needs of their organization. Many have invested heavily in SharePoint and are annoyed that Microsoft has not been able to connect Exchange to SharePoint in any coherent manner since SharePoint was first released in 2001.

Up front, the most important point to remember about site mailboxes is that they operate on the principle of keeping the content where it belongs. In other words, Exchange looks after email, and SharePoint takes care of documents. This is an appropriate and intelligent division because SharePoint is optimized for documents, and Exchange is not.

Public folders make excellent repositories for email discussions, but they are a pretty useless repository for document-based collaboration, such as when a group of people works together to author a document or presentation that will go through many editing cycles involving multiple contributors. This can be done by using email, but it's hard to keep track of the many versions of documents that circulate as message attachments unless someone is designated as the editor in chief. It's also possible to accomplish a collaborative authoring process by using public folders as long as great care is exercised over when different users access the document to update its content. Unfortunately, humans are often not so good at exercising the necessary care.

Site mailboxes are designed to fill the gap by using the strengths of Exchange (for email) and SharePoint (for document management). In effect, a site mailbox creates a virtual container that includes a shared mailbox and the document libraries on a SharePoint site. A SharePoint site is a website hosted by a SharePoint server that is identified by a virtual URL. Each site can be configured with different SharePoint components to create whatever functionality is required by the site's users. A site mailbox is an example of a component that can be added to a SharePoint site; tasks, calendar, and wiki apps are other examples.

The shared mailboxes created by the site mailbox app are fully functional mailboxes that have email addresses. The ability to accept email means that the site mailbox can be used as a form of email archive for the project if you add the site mailbox to any distribution groups used for project team communications. Each mailbox contains default folders such as the Inbox and Sync Issues, a folder that is particularly important because it captures problems that occur when Exchange and SharePoint swap information.

A synchronization process creates, populates, and maintains folders in the mailbox to represent the contents of the SharePoint document libraries that are associated with the site. Appropriately, these folders use a document-centric view to display their contents through Outlook rather than the conversation-centric view normally used for email messages in other folders. You don't realize where the join exists between Exchange mailbox folders and

the folders populated through synchronization with SharePoint because Outlook creates the impression that all the information in a site mailbox is held in one place. The fact that two servers work together to manage the site mailbox is immaterial to users. People never worried that old-style public folders used a completely different database and replication mechanism from Exchange mailboxes, so why should they worry that some items are in an Exchange mailbox and some are in a SharePoint site? The point is that they have access to the information they need to perform a task.

Two obvious dependencies leap off the page here. You need to deploy SharePoint 2013 to support the sites, and you need Outlook 2013 to be able to create the invisible join between mailboxes and sites. No other client is currently capable of accessing site mailboxes. Companies might well be considering a deployment of Outlook 2013 alongside Exchange 2013 because this version of Outlook exposes all the functionality in Exchange 2013, but a SharePoint deployment might represent more of a barrier to overcome. Deploying SharePoint requires additional investment in hardware, expertise, and software, and these factors have to be taken into account in any discussion about site mailboxes.

Nevertheless, assuming that all is well, that software and hardware have been procured and deployed, and that SharePoint and Exchange are both humming along like a well-tuned engine, the question of functionality is the next topic of discussion.

How site mailboxes work

The steps required to implement site mailboxes are described on TechNet and in many other web articles and do not need to be repeated here. In summary, the deployment of site mailboxes depends on:

- Exchange 2013 mailbox servers to host the shared mailboxes associated with the SharePoint sites on which the site mailbox app is installed.
- A SharePoint 2013 farm (or standalone server) to host the SharePoint sites that contain the document libraries and membership lists that are synchronized with Exchange and combined with the shared mailboxes to form site mailboxes. The SharePoint servers that host the sites and the Exchange servers that host the mailboxes must be on the same premises (on-premises or cloud), whereas the personal mailboxes of users can function cross-premises. In other words, an on-premises Exchange user can access site mailboxes that are stored in Office 365.
- Outlook 2013 clients to present a single user interface to the site mailboxes that include both the Exchange shared mailbox and the SharePoint document libraries.
- Exchange Web Services (EWS) and SharePoint representational state transfer (REST) application programming interfaces (APIs) to synchronize information between Exchange and SharePoint.

A considerable amount of work and cooperation is necessary between the Exchange and SharePoint administrators to configure the products to work together smoothly to enable the creation and operation of site mailboxes. Do not anticipate that this work can be done without preparation.

SharePoint communicates with Exchange using EWS, which means that you must download and install EWS on your SharePoint servers. It is important for a matching version of EWS to be installed on the SharePoint servers so they can communicate with the mailbox servers. In other words, if the mailbox servers run Exchange 2013 CU2, a matching version of EWS should be installed on the SharePoint servers.

Membership information (both users and owners) for the site mailbox is maintained through SharePoint. Users are permitted access to site mailboxes by being added to the membership list of the SharePoint site. Users have to be added individually because gaining access through group membership is not supported. In addition, users must have an Exchange mailbox, and that mailbox has to be on an Exchange 2013 server before they can use site mailboxes through Outlook. Users who have not yet been moved to Exchange 2013 can still interact with site mailboxes by opening the SharePoint site with a browser.

Behind the scenes, SharePoint synchronizes the membership list with Exchange to grant members full access rights to the shared mailbox. Like all other operations affecting the life cycle of a site, provisioning and updates for site membership is controlled from SharePoint, and Exchange does not proactively query SharePoint to discover new site members. After full access has been granted, Autodiscover adds the shared mailbox to the list of resources available to Outlook the next time it queries Exchange for this information, and Outlook opens the shared mailbox as soon as the new resource becomes known.

Outlook 2013 Professional Plus is the only client currently capable of displaying the site mailbox data about document libraries that Exchange retrieves from SharePoint. An Outlook Web App–like interface is available to the email items in a site mailbox if you click the Mailbox link when accessing the SharePoint site in a browser; this invokes an explicit logon to the site mailbox. However, unlike Outlook, Outlook Web App does not include site mailboxes in the set of available resources a user sees when he accesses his personal mailbox, and there are no public plans to change this situation. By comparison, Outlook considers a site mailbox like any other resource available to a user—like a shared mailbox, archive, or PST—so moving information into a site mailbox is as easy as dragging an item from another Outlook resource. In effect, anyone who knows how to work with Outlook can work with a site mailbox, which reduces the cost of deployment and support.

Items can be added to the site mailbox through SharePoint or Outlook by dragging and dropping items into the folders in the document library within a site mailbox or the other (regular) folders in the mailbox. It's best to maximize the relative strengths of Exchange and SharePoint by putting email items in email folders and documents (such as those received

as attachments to messages) into document libraries. This approach means that you'll be able to use features such as threaded conversations for the messages stored in the site mailbox's Inbox and version control for the items stored in the document library.

The content stored in site mailboxes is indexed and discoverable by eDiscovery searches. This is because Exchange 2013 shares the Search Foundation technology with SharePoint. Email has long since been indexed and discoverable; Exchange 2013 and SharePoint 2013 combine to make the documents held in site mailboxes discoverable, too—a fact that will surely bring joy to lawyers.

Site mailboxes are mail-enabled objects and behave in the same way as mail-enabled public folders. In other words, you can add a site mailbox as an addressee to a message, and Exchange will route the message to the Inbox folder in the site mailbox. Site mailboxes appear in address lists and can be hidden by setting the HiddenFromAddressListsEnabled property to $True. This step is usually taken to prevent users from including a site mailbox in messages when the mailbox is being decommissioned. Messages can still be sent to the site mailbox by using its Simple Mail Transfer Protocol (SMTP) address.

```
Set-Mailbox –Identity 'Project Alpha' –GrantSendOnBehalfOf 'Paul Robichaux'
```

The Send As permission can also be assigned for a site mailbox. In this case, you use the Add-AdPermission cmdlet. For example:

```
Add-AdPermission –Identity 'Project Alpha' –ExtendedRights 'Send-As' –User 'Tony
Redmond'
```

SharePoint documents stored in site mailboxes remain in place when you add them as attachments to messages. A link is added to the message to enable recipients to access the content, but the documents stay in SharePoint rather than being circulated as attachments. It makes perfect sense to have a single definitive source for a document that's intended as a collaborative object as long as the recipients have sufficient network connectivity to access the document when they need to. (The old replication model for public folders, although derided by many, at least had the singular advantage of making content available close to users.)

Items stored in site mailboxes are not subject to Exchange retention policies. All the information held in the site remains under the control of the SharePoint information policy that applies to the site. However, items in a site mailbox can be placed on hold using the SharePoint eDiscovery Center.

The stub items that represent the content of the document libraries can be synchronized down to the client OST like the contents of other mailbox folders if you select the Download Shared Folders check box on the Advanced tab of your Outlook profile. If not, Outlook makes an online connection to the shared mailbox when you want to work with its contents. Remember that only the stub items are synchronized. The actual content of

document libraries always remains under SharePoint control. These stored documents can be made available offline by synchronizing the site to create a local copy of the document library on the PC. And, of course, you can always download a local copy of an individual document from SharePoint for use offline.

Synchronization between Exchange and SharePoint

Exchange synchronizes with SharePoint by using the SharePoint REST API to fetch the metadata used to display information about documents stored in the site mailbox, including document title, author, last modified date, checked-out status, and the user who last modified the item. Information held in the documents library is synchronized and presented alongside all the other folders in the site mailbox so that the data from the two sources seems as though it originates from a common repository. Items added to a document library through Outlook appear in italics to indicate that the synchronization is not yet complete. In other words, the item has been added to the Exchange mailbox but not yet copied (or published) to SharePoint. After SharePoint has accepted the item and stored it in the document library, the italics are removed and not displayed by Outlook.

If you add more than one document library to the SharePoint quick launch menu, Exchange will synchronize the items from each library into a separate folder. As used in the example discussed here, the default name for a document library on a SharePoint site is Documents, and this is the folder name seen in Outlook. The library could also be called Shared Documents, Project Documents, or whatever other name you assign to it.

In general, the synchronization process between Exchange and SharePoint is reasonably fast, and new items added through one interface appear in the other soon after they are added. If you add a new document library through SharePoint, it will soon appear as a folder in Outlook. Likewise, if you drag an item from Windows Explorer or File Explorer and drop it into the Outlook folder, the item will be synchronized to the SharePoint site within a few minutes. The same is true if you drag and drop items from other Outlook folders into the site folder. You can even create a subfolder for the site folder in Outlook, and it synchronizes with SharePoint. Items placed in the subfolder will also be synchronized with SharePoint. All of this works very well and makes the site mailbox a very approachable repository for anyone who knows how to work with Outlook. The sole downside is that you cannot drag items out of any folder that is synchronized with SharePoint to Windows Explorer (or File Explorer) or another Outlook folder; traffic is unidirectional—into the folder. In addition, if you forward or otherwise reference an item shored in a synchronized folder, the item is not sent as an attachment. Instead, a link to the SharePoint site is embedded in the message to allow the recipient to access the content by clicking the link.

Synchronization is user-driven. In other words, although information about a new site or document library or changes to site membership need to be updated as quickly as possible, information about the contents of a document library might not be so time-critical,

especially for document libraries that are not updated very often. To make sure that Exchange and SharePoint remain synchronized, a timer job runs on the Exchange mailbox servers every six hours to make a request to SharePoint for information about new members, documents, and so on. However, if a site is actively used (documents or members are added), Exchange synchronizes with SharePoint every five minutes, and if a site is open in either SharePoint or Outlook, synchronization occurs every fifteen minutes even if items are not added or otherwise updated.

Failure of items to appear in a timely manner in either Exchange or SharePoint is an indication that something is wrong with the synchronization process. If this happens, you can check the last synchronization status by selecting the site mailbox in Outlook. Right-click it and select Site Mailbox Properties from the menu. This leads to a webpage (using EAC) that shows information about the site mailbox, including a Sync Status option. Here you'll find the date and time of the last successful synchronization together with any error messages plus options to start synchronization. A useful Email Me Details option is provided that requests Exchange to send you a message containing some diagnostic information about the synchronization.

INSIDE OUT SharePoint document name limitation

Exchange is perfectly happy to have multiple items with the same name in its folders. SharePoint is not so happy because it requires documents to have unique names. If you attempt to add a new item to a document library from Outlook, and the item already exists, everything will seem to be okay for a moment until SharePoint realizes that Exchange is attempting to publish an item that it cannot deal with. SharePoint will then drop the item and send the equivalent of a nondelivery notification (including an attachment with the failed item) back to the user to tell her that a SharePoint Server Exception has occurred because "the document already exists." The failure is also noted in the Sync Issues folder of the site mailbox. The solution is to copy the item to Windows, rename it there, and then drag and drop the renamed file into Outlook. SharePoint will be much happier when Outlook synchronizes with the document library, and all will be well. The process is not perfect, but it works.

Information about each document is stored as properties of a stub item in the folder used to represent the document library in the site mailbox, including the URL pointing to the item in SharePoint. If you examine the contents of the Documents folder by using MFCMAPI, you can see the properties of the SharePoint items that Exchange synchronizes. When Outlook connects to the site mailbox, it retrieves data about email items from other

folders as usual and uses the stub items in the Documents folder to construct the view of the information held in SharePoint.

Using stub items enables Outlook to have faster display access to items in the document library. When Outlook needs to retrieve content for an item held in SharePoint, it passes the URL of the item to the native application for the item. The application checks the Office Document Cache to determine whether an offline copy exists or whether it needs to retrieve a copy from SharePoint. A local copy might exist in the PC's local cache or because of a previous synchronization with the document library to create an offline copy. When the correct location for the content is determined, the application retrieves and opens the item. As far as the application is concerned, it has received a request to open and process content and does not care about its source. Although it is usual to store mail messages in Exchange rather than in a document library, you can move messages to a document library for purposes such as marking the messages as immutable records. Messages are stored in document libraries as .msg files, and Outlook opens these items. Logically, if you open an item such as a Word document or a PowerPoint presentation that is checked out to another user, you will not be able to update the content.

INSIDE OUT Synchronization only goes so far

You are probably used to the per-item read status map Exchange maintains to show users which items they have viewed and which they have not within a folder. Synchronization to make items in a document library visible to Outlook does not preserve such a status. For instance, if you add a new item to a document library in SharePoint, Exchange will subsequently synchronize it, and it will appear as an item in Outlook. However, the item is bolded, which is the normal convention that indicates a new item in a folder. The item is new to other users, but it's certainly not new to you because you just added it to SharePoint.

Figure 12-12 shows how items in a site mailbox are accessed through Outlook 2013. In this case, a site mailbox called Project Alpha is open. The Inbox, Deleted Items, and Junk Email folders are stored in a mailbox database that Exchange manages. The view of items in the Documents folder is constructed from the stub items. The properties synchronized to the stub items from SharePoint are sufficient to present an informative view to the end user.

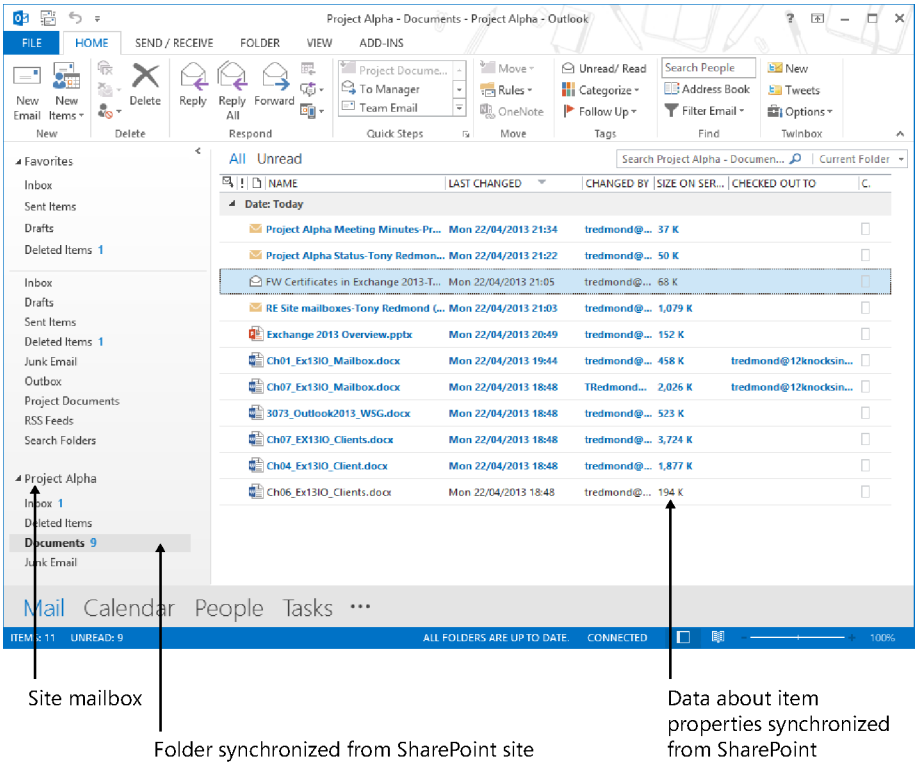


Figure 12-12 How Outlook 2013 displays a site mailbox and its contents

If you run the Get-MailboxFolderStatistics cmdlet to report on information held in a site mailbox, it reports the number of items and the size of the items held in the Documents folder. The number of items is correct, but the item size represents only the much smaller stub items that hold the properties describing the real documents that are synchronized from SharePoint.

Because Outlook provides the interface for site mailboxes, some of the advanced document management features available within SharePoint are not exposed. For example, you can't see prior versions of documents that are stored on the SharePoint site with Outlook, and you can't check out a document to reserve it against editing by other users. These features are available by connecting to the underlying SharePoint site with a browser. The easiest way to access the underlying site for a document library is to select the library in Outlook, right-click it, and select Open In Web Browser from the menu. Figure 12-13 illustrates the situation after you have connected to the same Project Alpha site as accessed through SharePoint through a browser. The same items Outlook presents in the Documents folder are visible, but now some additional options such as document checkout are available because the web interface supports the options.

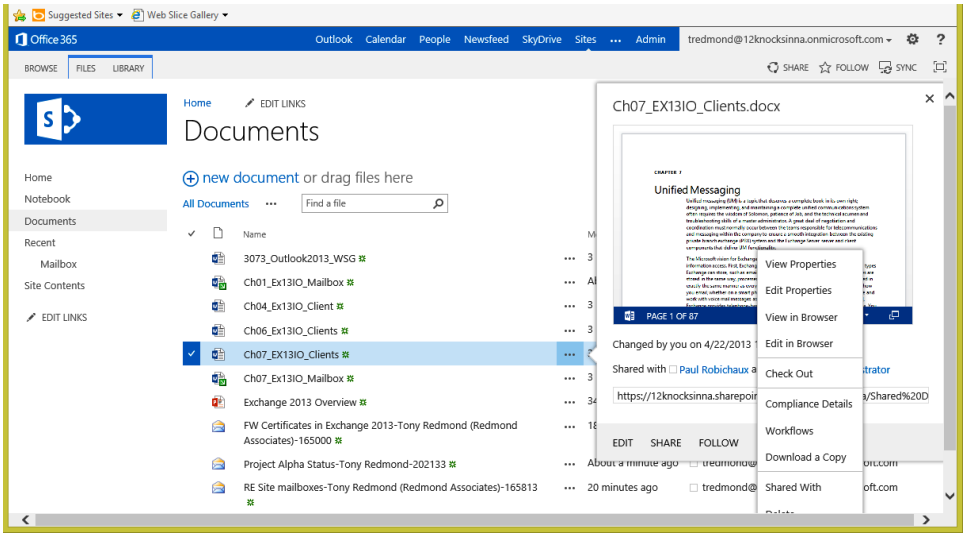


Figure 12-13 The same site viewed through Microsoft Internet Explorer

Accessing information about site mailboxes from Outlook

If you right-click your mailbox name at the top of the folder list, you see Manage All Site Mailboxes listed in the menu. Selecting this option causes Outlook to make a call to EAC to display the My Site Mailboxes page, listing the site mailboxes to which you have access. The list is displayed on a webpage rather than by Outlook. The following fields are shown:

- **Show In Outlook** Typically, this check box is selected, meaning that Outlook should open the site mailbox and display its contents. If you clear the box, the site mailbox will be removed from the Outlook list of resources. Selecting the box again restores matters, and the site mailbox will reappear.
- **Name** This is the name of the site mailbox.
- **Your Role** This will be *Owner* if you created the site mailbox or *Contributor* if the site mailbox has been shared with you.
- **Active** This is a flag showing whether the site is still active.

The details pane for the selected site shows the SMTP address of the site mailbox, the URL to the SharePoint site, the list of owners, and the number of members. If you click a site, EAC retrieves and displays status data for the site. You can discover the current synchronization status and the list of users who have access to the site here.

INSIDE OUT How many site mailboxes can Outlook open at one time?

Outlook 2013 is restricted to opening a maximum of 10 site mailboxes. The limitation exists for both usability and performance reasons. Having more than 10 site mailboxes listed in Outlook would make navigation very unwieldy and impose a large performance penalty on the client. Ten seemed to provide a good balance between the desire to make information available to users and the need to maintain performance. Exchange imposes the limitation by restricting Autodiscover to being able to include only 10 site mailboxes in the manifest that it returns to Outlook, so Outlook never knows that more site mailboxes might exist. If you have access to more than 10 site mailboxes, you can choose the mailboxes to open by choosing Manage All Site Mailboxes as previously described and selecting whatever mailboxes are required. After a short pause to allow Autodiscover to update Outlook, the selected mailboxes will appear in the resource list. Alternatively, select a site mailbox that you do not want to access anymore, right-click it, and select Close Site Mailbox from the menu. The site mailbox remains intact, but Outlook will no longer display it to you, again after a short pause to allow Autodiscover to update Outlook. A site mailbox that has been closed can be reopened by choosing Manage All Site Mailboxes.

Site mailboxes and Office 365

Site mailboxes are also available in many of the Office 365 plans Microsoft offers. Office 365 tenants might find it easier to adopt site mailboxes, if only because Microsoft takes care of all the heavy lifting to deploy and manage Exchange Online and SharePoint Online. All the tenant has to do before she can use site mailboxes is make sure that she uses Outlook 2013. In fact, the easiest way to review the functionality of site mailboxes and assess whether they offer value to your company is to take out a 30-day trial for an Office 365 plan that includes site mailbox functionality and use that tenant domain to test how site mailboxes work.

The life cycle of site mailboxes

Unique among Exchange mailbox types, you cannot create a site mailbox through EAC. Instead, you create a site mailbox as an app on a SharePoint team site. A site mailbox is treated like any other app such as a list that forms part of the functionality offered to team members. The logic driving this decision is that creating a site mailbox is done as part of the process to build the site to meet its business needs. If you think the site requires email to enhance the collaboration among team members, create a site mailbox. If not, don't create a site mailbox.

If you've invested in setting up processes for creating SharePoint sites at the enterprise level, you can have those processes create site mailboxes at the same time that a classic SharePoint site is created. It's also a good idea for the SharePoint administrators to determine whether new sites are subject to a life-cycle policy from the start to avoid many sites being created, used for a short time, and then remaining in an unused and unwanted state, gently withering away while occupying valuable resources. You might, for instance, decide that all sites will be removed by policy after three years to ensure that obsolete sites are removed after a reasonable time and that sites that are in active use are noted as such and site owners can take action to retain them. See *http://technet.microsoft.com/en-us/library/jj219569.aspx* for more information about site life-cycle policies in SharePoint 2013.

Most site mailboxes are created by administrators adding the mailbox to a site after it has been created. Figure 12-14 shows the default set of options presented to an administrator after a new site is created. The option to add a site mailbox is to the far right and called Keep Email In Context. Clicking this icon starts the process of creating a site mailbox.

Behind the scenes, SharePoint calls an EAC page to invoke the New-SiteMailbox cmdlet to create the new site mailbox and populate the properties of the mailbox with details such as mailbox ownership and access. Two distinct objects are created within Exchange. The first is the shared mailbox in which email items are stored; the second is the site mailbox object, which holds the properties that describe the connection between Exchange and SharePoint such as the URL that points to the SharePoint site. These objects can be examined with the Get-SiteMailbox cmdlet.

A certain amount of background processing and synchronization between Exchange and SharePoint has to occur before the site mailbox is fully ready for use. The work to create a site mailbox fully can take between 30 minutes and an hour to complete. The first user who can access the site mailbox is the person who creates the site. Shortly afterward, the access list for the site mailbox is updated with details of all the people who have access to the SharePoint site.

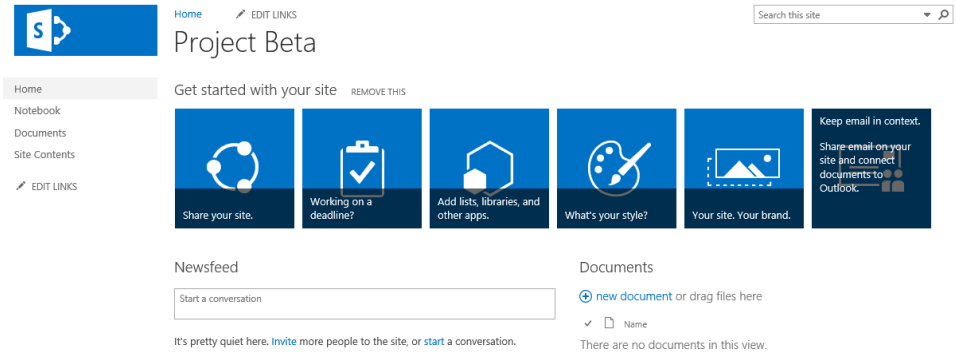


Figure 12-14 Adding a mailbox to a SharePoint site

When everything is ready, SharePoint sends an email to the site members to tell them that the mailbox is available. Autodiscover notices the presence of the site mailbox and adds it to the set of resources listed in the payload provided to Outlook. This information enables Outlook 2013 to list the site mailbox as part of the resources available to the user. Those who do not use Outlook 2013 can access the site mailbox through a web browser.

Exchange identifies site mailboxes by setting their RecipientTypeDetails property to TeamMailbox (reflecting an older name for site mailboxes). You can scan for these mailboxes with the following command:

```
Get-Mailbox -Filter {RecipientTypeDetails -eq "TeamMailbox"} | Select Name
```

It is possible to create new site mailboxes directly from Outlook. However, some caution should be exercised here because previous experience with public folders proves that users often create repositories they use for a few days and then move on, never going near the repository again. It is probably better to have users go through a more structured approach to site mailbox creation until everyone happily knows how to take best advantage of this feature. But if you want to allow users to have free rein, you can do this by updating the Exchange organization configuration with a pointer to the SharePoint new site–creation page. For example, this URL works for an Office 365 tenant:

```
Set-OrganizationConfig -SiteMailboxCreationURL
"https://contoso.sharepoint.com/_layouts/15/SelfServiceCreate.aspx?Context=Site"
```

After the organization configuration is updated, users can use the Create New Site Mailbox option in the menu that Outlook reveals when they click their account name in the Outlook navigation pane. The option redirects the user to the selected webpage.

At the other end of a site mailbox's life cycle, when the site is no longer required, you do not remove the site mailbox from Exchange. Instead, you remove the site from SharePoint, and SharePoint makes an EWS call to Exchange to request the deletion of the site mailbox object.

When a site that includes a mailbox is removed from SharePoint by the life-cycle policy application, the site mailbox object is removed from Exchange, but the shared mailbox persists. In this instance, the mailbox is not removed immediately. Instead, it is tagged as obsolete by adding the "MDEL" prefix to its name by a job that runs daily. The mailbox is left in place and is not removed until an administrator deletes it by running the Remove-Mailbox or Remove-StoreMailbox cmdlet. (By comparison, site mailboxes hosted on Office 365 are deleted without the need for administrator intervention.) The mailboxes are left in place so their contents can be accessible for discovery searches for as long as necessary. Because the mailboxes are in place and not disabled, they also receive email.

It is possible to decouple the mailbox from the SharePoint site, which effectively leaves the mailbox in place with no connection to the site. You can do this by running the Set-SiteMailbox cmdlet. It's not recommended to do this unless directed by support personnel, but for the record, an example command is:

```
Set-SiteMailbox –Identity "Project Alpha" –SharePointUrl $Null
```

INSIDE OUT **Sizing site mailboxes**

At this point, we do not have much data to assess the impact of site mailboxes on system performance. However, it is fair to say that site mailboxes are essentially passive repositories when compared to user mailboxes. Some synchronization activity occurs behind the scenes to keep Exchange up to date with SharePoint, and users access the site mailboxes with Outlook. Aside from this, site mailboxes do not impose much strain on an Exchange server. It's hard to say how to account for site mailboxes when you size a server. Until harder real-life data is available, it is reasonable to expect that a site mailbox will take a tenth of the resources of a user mailbox and to size on that basis.

Site mailbox provisioning policy

Like many other places in Exchange where resources have to be controlled, site mailboxes come under the control of the site mailboxes provisioning policy that determines:

- The maximum size of a message that can be sent to a site mailbox. The default value is 36 MB (MaxReceiveSize).
- The threshold for mailbox size that causes Exchange to send warnings to owners of a site mailbox that their quota is becoming exhausted. The default value is 4.5 GB (IssueWarningQuota).
- The threshold at which no further items can be stored in the site mailbox. The default value is 5 GB (ProhibitSendReceiveQuota).

The quota values apply to the Exchange mailbox and do not have any effect on the data held on the SharePoint site. You cannot work with the site mailbox provisioning policy through EAC; you have to make any changes by using the *-SiteMailboxProvisioningPolicy cmdlet set through EMS. In most cases, the default policy will serve until some hard data is available to indicate how much usage site mailboxes receive and what kind of data they contain. At that point, you might decide to increase the quotas assigned to site mailboxes.

For example, to increase the warning and quota thresholds to 9.5 GB and 10 GB, respectively, for the default policy, you run this command:

```
Set-SiteMailboxProvisioningPolicy -Identity 'Default' -IssueWarningQuota 9.5GB
-ProhibitSendReceiveQuota 10GB
```

Although you can create new site mailbox provisioning policies, only a single policy applies to all site mailboxes in an organization. The current policy that applies is identified by setting the IsDefault property to $True.

INSIDE OUT **RBAC and site mailboxes**

If you look at the default role assignment policy included in Exchange 2013, you'll see that the MyTeamMailboxes role is included. This is the RBAC role that permits users to create new site mailboxes. (The team mailboxes name is an old one for the feature.) You can remove this role from the policy if you don't want users to be able to create site mailboxes, or you can create a new role assignment policy that omits MyTeamMailboxes and assign that policy to users who should not create site mailboxes.

Summarizing public folders and site mailboxes

Public folders have been in use since 1996; the evolution to modern public folders in Exchange 2013 is the first major technical change made in public folders in that time. The older form of public folders is important to many companies that have accumulated gigabytes if not terabytes of information in this repository. The challenge now is to make a flawless transition to the new world of modern public folders.

Site mailboxes are at the start of their journey, and I expect that a huge amount of change will take place in this space over the next few years, not least automation and simplification of the setup and automation of their management. No real operational experience exists yet for site mailboxes because companies have not deployed the necessary technical infrastructure to support site mailboxes. After the technical refresh is accomplished, companies will move to discover whether site mailboxes fit in their collaboration strategy and, if so, how best to deploy and manage site mailboxes to meet business needs.

The promise of site mailboxes is that they will be a much better collaborative tool than public folders in many ways, especially when complex documents need to be developed by several people. It will be interesting to see how site mailboxes are deployed and used in production and whether, after a year or so of real-life use, customers who deploy site mailboxes have gained the promised advantages.

Moving on

This ends the discussion about Exchange 2013 Mailbox server without having exhausted the full set of topics that could have been reviewed. The architectural change made in Exchange 2013 forced the transport functionality to move to the mailbox server role, whereas unified messaging has been relocated from the Client Access Server role to make it more of a stateless server. These topics, along with other important subjects, such as client choice and control, are covered in *Exchange Server 2013 Inside Out: Connectivity, Clients, and UM* by Paul Robichaux (Microsoft Press, 2013). It's a good suggestion for your next destination.

Index

Numbers and Symbols

$_ variable, 101–102
$_. variable, 102
{ } (braces), 110
& (ampersand), 385
*** (asterisk), 138, 481**
- (hyphen), 103, 110
. operator, 100
" (quotation marks), 102–103
_ (underscore), 26
| operator, 102, 109

A

ABPs (address book policies)
about, 348–350
creating and implementing, 351–357
groups and, 350
accepted domains, 340
–AcceptLargeDataLoss cmdlet parameter, 632
AcceptMessagesOnlyFrom property, 279–280
AcceptMessagesOnlyFromDLMembers property, 279–280
AcceptMessagesOnlyFromSendersOrMembers property, 279
access control, role-based. *See* RBAC (role-based access control)
access control lists (ACLs), 131–132, 143, 282
ACID test, 419
ACLL (attempt copy last logs) process, 473, 478–479
ACLs (access control lists), 131–132, 143, 282
activating mailbox database copies, 533–534
activation blocks, 542–544
activation preference for databases, 477
ActivationPreference property, 521
Active Directory
about, 34–37
address book policies and, 350
ADSIEdit utility, 36–37, 59–60
deployment support, 45–47
ensuring full visibility, 364
Federated Services role, 313
forest functional mode, 26
preparing, 47–48
removing tags from, 687
storing information, 35–36
user accounts, 49
Windows PowerShell and, 120–123
Active Directory Domain Services (AD DS), 47–48
Active Directory Rights Management Services (AD RMS), 734
Active Directory Users and Computers
audit entries and, 754
changing group properties, 285
creating computer objects, 497
displaying group membership, 268
Find option, 115
moving groups, 289
USGs and, 141
Active Manager
about, 461, 465–467
ACLLs, 478–479
automatic database transitions, 467–469
best copy and server selection, 472–477
DAC and, 552
failover clustering and, 464
high availability and, 465–466, 469–472
Managed Availability and, 469–472, 536
scanning for failed copies, 458
ActiveCopy property, 521
ActiveDirectoryConnectivityConfigDCRestart responder, 470
ActiveSync, 29, 125–126
–ActiveSyncMailboxPolicy cmdlet parameter, 193
AD DS (Active Directory Domain Services), 47–48
AD RMS (Active Directory Rights Management Services), 734
Add-AdPermission cmdlet, 232, 794
Add-DatabaseAvailabilityGroupServer cmdlet, 494, 496
Add-DistributionGroupMember cmdlet, 92, 193, 296, 298
Add-MailboxDatabaseCopy cmdlet, 525, 527, 545
Add-MailboxPermission cmdlet, 202, 233–234, 236–237, 260
Add-PublicFolderClientPermission cmdlet, 787
Add-RoleGroupMember cmdlet, 150
Add-WindowsFeature cmdlet, 51, 53

AddNewRequestsTentatively property, 258
AddOrganizerToSubject property, 258
address book policies (ABPs)
about, 348–350
creating and implementing, 351–357
groups and, 350
address lists, 345–349
addressing messages
about, 333
address book policies, 348–357
address lists, 345–348
display templates, 382–386
email address policies, 333–345
hierarchical address books, 349, 371–373
MailTips and group metrics, 373–382
Offline Address Book, 357–371
ADDriver (Exchange Authorization Library), 90–91
Admin Audit Log agent, 750
AdminAuditLogs folder, 749
Administrator user account, 163
administrators
auditing actions of, 748–754
delivery reports, 321–327
page zeroing, 436
quorum voting, 494
running EAC without Exchange mailbox, 329–330
security and, 319–320
user accounts, 163
ADSIEdit utility
about, 36–37
arbitration mailboxes, 59
auditing and, 754
email address formats, 344
removing server objects, 60
viewing group properties, 374
Advanced Search Syntax (AQS), 736
AgeLimitForRetention property, 666, 670
AggregateCounts corruption type, 446
AggregatePFData.ps1 script, 788–789
alerts
EAC, 172–173
health mailboxes, 227
mail flow and rules, 318–319
moving mailboxes and, 638–639
self-signed certificates, 78
–Alias cmdlet parameter, 192, 253, 307
Alias property, 185
aliases, 93
All Contacts container, 345
All Groups container, 345
All Rooms container, 345
All Users container, 345
AllBookInPolicy property, 262–264
AllRequestInPolicy property, 262, 264
AllRequestOutofPolicy property, 262–263
AllSigned execution policy, 118
AlternateWitnessDirectory property, 505
AlternateWitnessServer property, 505
ambiguous name resolution (ANR), 365–366
ampersand (&), 385
ANR (ambiguous name resolution), 365–366
ANSI format, 623
APIPA (Automatic Private IP Addressing), 492
Application and Service Logs, 502
Application Event Log
diagnostic levels and, 332
ecTooManyMountedDatabases error, 544
excessive growth, 443
usage examples, 364, 433
Application Impersonation role, 159
AQS (Advanced Search Syntax), 736
arbitration mailboxes
about, 182
alerts and, 638–639
checking, 547
discovery mailboxes, 224
migration, 573
moving, 59–60
organization mailboxes, 362
suppressing, 312
ArbitrationMailbox property, 249
architecture, Exchange Server, 10–11
–Archive cmdlet parameter, 193, 446, 648, 652
archive mailboxes
about, 645–647, 655–656
checking space usage, 652
compliance management and, 644
DAG design, 557
default archive and retention policy, 652–655
development priorities, 4
disabling, 656–657
enabling, 647–650
limitations of, 647
managing properties, 651–652
server failure and, 649
troubleshooting, 656
updating properties, 652
–ArchiveDatabase cmdlet parameter, 648–649
ArchiveDomain property, 651
ArchiveRelease property, 651
ArchiveState property, 651
ArchiveStatus property, 651
arrow keys, 95
–AssociatedMessagesCopyOption cmdlet parameter, 633
asterisk (*), 138, 481
asynchronous moving, 571–572
AsyncOperationNotification system retention tag, 639
attachments, 398
attempt copy last logs (ACLL) process, 473, 478–479
attributes. *See* specific properties

audit mailbox, 749–750
AuditAdmin property, 758
AuditDelegate property, 758
auditing
administrator actions, 748–754
mailbox access, 239, 754–762
AuditLogAgeLimit property, 757–758
AuditOwner property, 758
authentication
certificates and, 314
cmdlets and, 89
Kerberos, 79
sending messages to groups, 280–281
auto-mapping mailboxes, 235–237
–AutoDatabaseMountDial cmdlet parameter, 478–479, 539–542
AutoDatabaseMountDial setting, 478–479, 539–542
–Autodiscover cmdlet parameter, 612
Autodiscover feature, 235–237
AutomateProcessing property, 257, 261
automatic database transitions, 467–469
Automatic Private IP Addressing (APIPA), 492
automatic reseeding of failed databases, 458
automating mailbox settings, 205–206
–AutoSize cmdlet parameter, 99
–AutoStart cmdlet parameter, 593
autosynchronization, 588
Azure File Server, 490

B

B+ tree defragmentation, 435
backend (Mailbox server role), 1, 314
background maintenance
about, 390, 432–433
content maintenance tasks, 432, 436–437
continuous, 482
database checksums, 432–435
database compaction, 432, 436
database defragmentation, 432, 435
page zeroing, 433, 436
Backspace key, 95
backups
database, 416–417
streaming, 29, 389
VSS, 458
bad-item limit, 584–587, 590, 608
–BadItemLimit cmdlet parameter, 637
BadItemLimit property, 593
BalanceDbsByActivationPreference mode, 535
BalanceDbsBySiteAndActivationPreference mode, 536
–BatchName cmdlet parameter, 575, 634
BCS (best copy selection), 472
BCSS (best copy and server selection), 472–477
Begin log record type, 425
best copy and server selection (BCSS), 472–477
best copy selection (BCS), 472
binary large objects (BLOBs), 401
Binpatch.oab file, 361
BLOBs (binary large objects), 401
block mode replication, 484–486
BookInPolicy property, 262–264
BPOS (Business Productivity Online Services), 6
braces, 110
bulk mailbox creation, 195
bulk updates, 114–116
business cases, building, 16
Business Productivity Online Services (BPOS), 6
–BypassDelegateChecking cmdlet parameter, 328
–ByPassNestedModerationEnabled cmdlet parameter, 245
–BypassSecurityGroupManagerCheck cmdlet parameter, 277

C

Calcheck (Calendar Checking Tool for Outlook), 445
Calendar Assistant, 259, 261, 454
Calendar Checking Tool for Outlook (Calcheck), 445
Calendar folder, 658, 695
Calendar Repair Assistant, 746
CalendarLoggingQuota property, 746
calendars, retention policies and, 677
CalendarVersionStoreDisabled property, 746
Calender Logging folder, 746
CALs (Client Access Licenses)
about, 33
applying retention policy to mailboxes, 678
archiving and, 644
mailbox management and, 184, 191
requirements, 27, 72–73
Can Opener screen, 55
canceling repair jobs, 447
capacity planning, 389–392
CAS (Client Access Server)
about, 1
address book policies, 349
architectural overview, 10–11
client interaction, 376–377
DAG and, 461
development problems, 7–8
MailTips and, 376–377
MRS and, 567
MRSProxy and, 612–613
namespace planning, 79–81
public folders and, 767
testing programming and customizations, 30–31
transport service, 27
case sensitivity, date/time formats, 199
CCR (cluster continuous replication), 29
cd command, 117
Central Help Desk security group, 323
CER file, 317

certificates
managing, 313–317
protocols and, 314
self-signed, 77–78, 314, 317
signed scripts, 118
SSL, 30
X.509, 26, 313
ChangedGroups.txt file, 376
ChangePermission right, 260
Checkbox control type, 385
CheckDatabaseRedundancy script, 522
checkpoint files, 408, 426–427
checksums
database, 432–435
transaction logs, 427
CI (Content Indexing) catalog, 554
circular logging
about, 429–431
DAG and, 510–514, 554
default state, 414
transaction log truncation, 486
clearing move requests, 610–611
Client Access Licenses (CALs)
about, 33
applying retention policy to mailboxes, 678
archiving and, 644
mailbox management and, 184, 191
requirements, 27, 72–73
Client Access Server (CAS)
about, 1
address book policies, 349
architectural overview, 10–11
client interaction, 376–377
DAG and, 461
development problems, 7–8
MailTips and, 376–377
MRS and, 567
MRSProxy and, 612–613
namespace planning, 79–81
public folders and, 767
testing programming and customizations, 30–31
transport service, 27
client-side filters, 110–113
client-side rules, 318
closed groups, 270
cloud service, 5, 9, 646
cluster continuous replication (CCR), 29
Cluster Name Object (CNO), 490, 496–499
cluster technology, 463–464, 492–493
cmdlets. *See also* specific cmdlets
about, 21, 83–86, 92–95
Active Directory module, 120–122
bulk updates, 114–116
command editing, 95–97
default view, 94
determining availability of, 86
execution policies, 118–119
filtering, 110–113
/full switch, 94
handling information returns, 98
HTML reports, 127–128
online repair, 445–448
output from, 100, 125–127
/parameter switch, 94
piping, 109
quarantined mailboxes, 439
RBAC restrictions, 154
remote Windows PowerShell and, 86–92
repair, 445–448
role assignment policies, 153–155, 163–166
roles and, 139
in scripts, 93, 117
selective output, 98–101
transcripts, 114
variables and, 101–103
/verbose switch, 129
/whatIf switch, 113
CNO (Cluster Name Object), 490, 496–499
Codeplex website, 38
columns, EAC, 177–178
comma-separated-value (CSV) files
analyzing, 68
CSVDE utility, 114
exporting EAC information to, 181–182
migration batches, 592
modern public folders and, 789
moving mailboxes, 582, 590
outputting, 125–127, 281–282
command editing, PowerShell, 95–97
Commit log record type, 425
compacting database, 432, 436
comparison operators, 110
Complete-MigrationBatch cmdlet, 594
compliance management
about, 641–642
archive mailboxes, 645–657
auditing administrator actions, 748–754
auditing mailbox access, 754–762
development priorities, 4
evolving needs of, 3
legal discovery actions, 642–645
messaging records management, 657–689
preserving information, 696–737
Recoverable Items structure, 737–747
testing operational processes, 29
upgrading considerations, 18
Compliance Management role group, 74
compression
attachments and, 398
LZ77, 506

LZX, 361
mailbox database, 394, 397–398
transaction log, 483–484
Computer Administrator account, 329–330
concurrent searches, 733
conditional checking in scripts, 103
conditional parameters, 307–313
–ConditionalCompany cmdlet parameter, 307, 309
–ConditionalCustomAttribute cmdlet parameter, 307, 309
–ConditionalDepartment cmdlet parameter, 309
–ConditionalStateOrProvince cmdlet parameter, 307, 309
–ConfictResolutionOption cmdlet parameter, 632
configuration
Active Directory information, 35
MailTips, 377–378
MRS, 621–623
XML files, 51–52, 205–206
Configuration read scope, 142
Configuration write scope, 142
–ConfigurationOnly cmdlet parameter, 545–546
–Confirm cmdlet parameter, 113, 656
ConflictPercentageAllowed property, 260
Connect-ExchangeOnline function, 90
Connect-Mailbox cmdlet, 222–223
contacts, mail-enabled, 179, 250–251
Contacts folder, 658
content indexes, 701, 776
Content Indexing (CI) catalog, 554
content maintenance tasks, 432, 436–437
ContentIndexState property, 542
ContentMailbox property, 780
contiguous space, 392, 394–395, 397–398
continuous replication circular logging (CRCL), 511
controller process, 402–407
ConvertTo-HTML cmdlet, 127
cookie files, 376
copies, database. *See* database copies
Copy action, 756
CopyQueueLength property, 521
corrupt items
detection and isolation, 437–448
moving mailboxes, 569
corruption, logical, 526
Country property, 305
CrashCount system registry setting, 438
CRCL (continuous replication circular logging), 511
Create action, 756
–Credentials cmdlet parameter, 612
crimson channel, 475–476, 502–504
Crowley, Mike, 124
CSV (comma-separated-value) files
analyzing, 68
CSVDE utility, 114
exporting EAC information to, 181–182
migration batches, 592
modern public folders and, 789
moving mailboxes, 582, 590
outputting, 125–127, 281–282
CSVDE utility, 114
Ctrl+Click key combination, 361
CU (cumulative updates), 14, 27, 64–67
cumulative updates (CU), 14, 27, 64–67
Cunningham, Paul, 124
custom filters, 308–313
–CustomAttribute cmdlet parameter, 352–355, 385
–CustomConfigWriteScope cmdlet parameter, 143
–CustomRecipientWriteScope, 143, 157

D

%d variable, 339
DAC (Datacenter Activation Coordination), 551–552
DACP bit, 552
DACP (Datacenter Activation Coordination Protocol), 552
DAG (Database Availability Group)
about, 19–20, 32–33, 311, 457, 461–463, 489–494
activating mailbox database copies, 533–534
activation blocks, 542–544
Active Manager, 465–479
adding database copies, 514–517
adding database copies with EMS, 525
architectural overview, 10–11
AutoDatabaseMountDial setting, 539–542
building, 494–496
building DAG, 494–496
changes in high availability, 457–459
changes in message submission, 489
circular logging, 510–514, 554
cleaning up, 510
crimson events, 475–476, 502–504
DAG networks, 506–509
data center activation coordination, 551–552
databases and, 362, 411, 459–464
day-to-day management and operations, 489–551
design considerations, 552–562
handling storage failures, 549–551
I/O performance improvements, 401
lagged database copies, 525–532
managing properties, 504–506
migrating, 464–465
monitoring database copies, 517–522
moving database locations within, 544–546
moving databases, 539–542
naming conventions, 179
passive database copies, 460
performing server switchovers, 536–539
pre-staging Cluster Name Object, 497–499
public folders and, 767–768
rebalancing database copies, 534–536
removing database copies, 546–548
removing servers, 548–549

reseeding database copies, 523–525
role of FSW, 499–501
server maintenance, 562–565
server names, 44
sizing mailboxes, 394
stressed servers, 565–566
task logging, 502
transaction log replays, 479–489
virtual servers, 24
Windows Failover Clustering, 463–464
DAMs (deferred action messages), 318–319
dark zone, 60
Data Guarantee API, 609
data loss prevention (DLP), 18, 642, 646
Database Availability Group (DAG)
about, 19–20, 32–33, 311, 457, 461–463, 489–494
activating mailbox database copies, 533–534
activation blocks, 542–544
Active Manager, 465–479
adding database copies, 514–517
adding database copies with EMS, 525
architectural overview, 10–11
AutoDatabaseMountDial setting, 539–542
building, 494–496
building DAG, 494–496
changes in high availability, 457–459
changes in message submission, 489
circular logging, 510–514, 554
cleaning up, 510
crimson events, 475–476, 502–504
DAG networks, 506–509
data center activation coordination, 551–552
databases and, 362, 411, 459–464
day-to-day management and operations, 489–551
design considerations, 552–562
handling storage failures, 549–551
I/O performance improvements, 401
lagged database copies, 525–532
managing properties, 504–506
migrating, 464–465
monitoring database copies, 517–522
moving database locations within, 544–546
moving databases, 539–542
naming conventions, 179
passive database copies, 460
performing server switchovers, 536–539
pre-staging Cluster Name Object, 497–499
public folders and, 767–768
rebalancing database copies, 534–536
removing database copies, 546–548
removing servers, 548–549
reseeding database copies, 523–525
role of FSW, 499–501
server maintenance, 562–565
server names, 44
sizing mailboxes, 394
stressed servers, 565–566
task logging, 502
transaction log replays, 479–489
virtual servers, 24
Windows Failover Clustering, 463–464
Database Availability Group management service (MsExchangeDAGMgmt.exe), 503
–Database cmdlet parameter, 193, 634
database copies
activating mailbox, 533–534
activation blocks, 542–544
adding with EMS, 525
automatic database transitions, 467–469
cleaning up, 510
DAGs and, 465, 514–517, 555
failover, 466
identifying specific, 525
lagged, 525–532
monitoring, 517–522
passive, 460, 544
rebalancing, 534–536
removing, 546–548
reseeding, 523–525
scanning for, 458
switchover, 466, 533–534
database management
about, 407–409
activation preference, 477
automatic database transitions, 467–469
automatic reseeding of failed databases, 458
backups, 416–417
circular logging and, 429–430
cleaning up before creating copies, 510
creating mailbox databases, 410–414
custom filter to address mailboxes in databases, 310–313
DAGs and, 362, 555–556
database checksums, 432–435
database compaction, 432, 436
database copies, 510, 514–536
database defragmentation, 432, 435
database redundancy, 411, 522
database replication, 390, 411, 459, 479–489
database schemas, 398–400
database usage statistics, 451–453
debugging swelling databases, 444–445
dismounting databases, 422, 447, 469, 545
estimating database size, 217–218
log placement, 428–429
mailbox resources provisioning management agent, 209–213
maximum database size, 389–392
maximum number of active databases, 402–403
mounted databases, 407, 426–427, 462–463, 546
moving databases, 536, 539–542, 544–546

naming conventions, 179
numbers supported, 407
OAB and, 371
protection against excessive database growth, 443–444
rebuilding databases, 448–451
removing databases, 416–418
renaming databases, 414–415
seeding databases, 489
updating after installation, 414–416
write smoothing, 395
database portability, 459–464
database scope, 158–159
database usage statistics, 451–453
DatabaseAvailabilityGroupIpAddresses property, 505
Datacenter Activation Coordination (DAC), 551–552
Datacenter Activation Coordination Protocol (DACP), 552
–DatacenterActivationMode cmdlet parameter, 552
DataMoveReplicationConstraint property, 620–621
DataMoveReplicationConstraint setting, 481, 609
Data.oab file, 361
DataPath property, 410–411
date formats, 199
debugging
generating transcripts, 114
swelling databases, 444–445
Default Archive and Retention Policy, 653–655
Default Archive Policy, 653
default folders, 200–201, 393
default group location, 288–289
Default MRM Policy, 652, 655
default policy tags (DPT), 658–659, 666
Default Role Assignment Policy, 290, 295
DefaultPublicFolderMailbox property, 771
deferred action messages (DAMs), 318–319
Deferred Actions folder, 318
defragmentation, database, 432, 435
Defragmentation Tasks counter, 435
Delegated Setup role group, 63, 74, 141
delegating
groups, 275
mailboxes, 229–232
Outlook access, 239–240
role assignments, 160
roles, 140, 163, 625
–Delegating cmdlet parameter, 154
Delete key, 95
delete tags, 672
DeleteAndAllowRecovery action, 672
DeleteAttachments property, 257
DeleteComments property, 257
–DeleteContent cmdlet parameter, 703
Deleted Items folder, 393, 649, 658, 695
–DeleteExistingFiles cmdlet parameter, 524
DeleteNonCalendarItems property, 258
DeleteSubject property, 257
Deletions folders, 746
delivery reports
about, 320–322
administrator searches for, 322–327
EMS searches for, 327–329
troubleshooting, 323
designing
DAGs, 552–562
retention policies, 663–665
detail templates, 382–386
Details Templates Editor, 382–386
detecting corrupt items. *See* corrupt items
–DetectOnly cmdlet parameter, 446
development priorities, 2–5
DHCP (Dynamic Host Configuration Protocol), 490
diagnostics for Exchange Server, 330–332
Direction property, 598
dirty pages, 425
Disable-Mailbox cmdlet, 92, 219, 656
Disable-MailPublicFolder cmdlet, 786
disabling
administrative auditing, 748
archive mailboxes, 656–657
mailboxes, 218–220, 223
retention tags, 687
disaster recovery mechanisms, 529
DisconnectDate property, 220
discovery actions, 642–643, 646
discovery mailboxes, 76, 182, 188, 224–226, 721
Discovery Management role group
about, 74
controlling access to discovery mailboxes, 722
default discovery mailboxes and, 234
linking to tasks, 135
performing searches, 225
RBAC support, 174
role assignment and, 152
discovery metadata mailboxes, 224
discovery search mailboxes, 224
DiscoveryHolds folder, 726–727, 746
DiscoveryMaxConcurrency property, 725
DiscoveryMaxKeywords property, 725
DiscoveryMaxKeywordsPerPage property, 725
DiscoveryMaxMailboxes property, 725
DiscoveryMaxMailboxResultsOnly property, 725
DiscoveryPreviewSearchResultsPageSize property, 725
DiscoverySearchMailbox account, 76
DiskFreeSpace property, 522
DiskFreeSpacePercent property, 522
DiskTotalSpace property, 522
Dismount-Database cmdlet, 415, 512, 545
dismounting databases, 422, 447, 469, 545
display templates, 382–386
–DisplayName cmdlet parameter, 192, 307, 348
DisplayName property, 185, 222

DistinguishedName property, 185
distribution groups
about, 267–269
address book policies and, 350
completing mailboxes, 193
creating, 270–275
defining default location, 288–289
dynamic, 179, 246, 300–313
group expansion, 277–278
group naming policy, 287, 289–295
group owners, 271, 275–277, 350
managing, 269–270, 286–297
metrics for, 275
moderated, 244–246, 270, 283
protected, 278–281
security groups, 252, 284–285
tracking usage, 285–286
viewing group members, 268–269, 281–283
Distribution Groups role, 147
DistributionGroupNamingPolicy property, 292
DLL (dynamic link library), 344–345
DLP (data loss prevention), 18, 642, 646
DNS (Domain Name System), 327, 462, 508
document library, 796
domain controllers, 495
Domain Name System (DNS), 327, 462, 508
–DomainController cmdlet parameter, 91
domains
accepted, 340
Active Directory information, 35
preparing, 48
–DoNotIncludeArchive cmdlet parameter, 731
DPT (default policy tags), 658–659, 666
drizzle synchronization, 578
.dsc file extension, 376
DSN error code 5.7.1, 280
dynamic buffer allocation, 402
dynamic distribution groups
about, 270, 300
Active Directory and, 176
creating, 302–305
creating with EMS, 306–308
custom filters, 308–313
OPATH queries, 300–302
validating query results, 305–306
Dynamic Host Configuration Protocol (DHCP), 490
dynamic link library (DLL), 344–345
dynamic quorums, 494
dynamic throttling, 580

E

E2EComplete migration product, 575
EAC (Exchange Administration Center)
about, 24
accessing data, 176–177
accessing mailbox audit data, 759
address book policies, 352–353
applying retention policy to mailboxes, 677
archive mailboxes, 650–652
bulk mailbox creation, 195
certificate management, 313–317
changing columns, 177–178
Compliance Management section, 171, 668, 673, 759
creating DAGs, 490–491
delivery reports, 320–329
development priorities, 4
distribution groups, 267–313
exporting information to CSV files, 181–182
filtering capabilities, 177, 342
Groups section, 302, 331
Hybrid section, 172
importing and exporting mailbox data with, 629–632
in-place holds, 705–706
language support, 196–200
mail flow and rules, 318–320
Mail Flow section, 172, 325, 333, 335
Mailbox Delegation section, 229–230, 232–233
mailbox import and export, 629–632
management interfaces, 21–22, 170–173
managing migration batches, 581–592
managing recipients, 180–183
migration batches with, 574, 581–592
Mobile section, 172–173
moving databases and, 544
mysterious mailboxes, 182–183
naming conventions, 178–180
Offline Address Book and, 362
Organization section, 171, 313, 346, 352
permissions, 143–144, 233–234
Permissions section, 133–134, 143–144, 171, 295
Protection section, 171
public folders and, 766
Public Folders section, 172, 188, 767
RBAC support, 132
Recipients section, 146, 171, 177, 182, 241, 250, 273–274, 581
refresh option, 176
replication queue lengths, 481
reporting licenses, 70–71
running without mailboxes, 329–330
Servers section, 172, 315, 409–410, 494, 537
setting diagnostics for Exchange Server, 330–332
setting up mailboxes, 191–193
sharing policies, 313
starting, 173–175
Unified Messaging section, 172
viewing correct, 174
viewing migration batch properties, 595–596
Windows PowerShell and, 84
ECP (Exchange Control Panel), 21–22, 170, 269, 707

ecTooManyMountedDatabases error, 544
EDB (Exchange database), 388
edge servers, 48
eDiscovery searches, 224–225, 233, 702, 726
Edit control type, 385
editions, Exchange Server, 22–23, 32–33
ELC (Email Lifecycle Assistant), 726–727
email address policies
about, 333–334
creating, 337–340
email policy priority, 334–336, 340
filters and, 341–345
mask variables, 339
Email Lifecycle Assistant (ELC), 726–727
email notification, 595–596
email service. *See* addressing messages; mailbox management
–EmailAddress cmdlet parameter, 612
EMC (Exchange Management Console), 21–22, 650
EMS (Exchange Management Shell)
about, 24, 31, 83, 92–120
accessing mailbox audit data, 759
Active Directory for PowerShell, 120–123
adding database copies, 525
address book policies, 352–353
creating dynamic distribution groups, 306–308
delegated setup and, 63
email address policies, 343–345
enabling archives, 648
executing searches, 731–732
handling information returns, 98
mailbox import and export, 632–637
MailTips management, 377–378
management interfaces, 21–22, 170
migration batches with, 574, 592–595
public folders and, 766
room lists, 270, 298
searches for delivery reports, 327–329
useful examples, 123–128
Windows PowerShell, 83–92, 129
Enable-CmdletExtensionAgent cmdlet, 206
Enable-Mailbox cmdlet, 193–194, 206, 649, 651
Enable-QuarantineMailbox cmdlet, 440
end-user roles, 164–165
endpoints, migration, 611–612
Enterprise Administrators group, 48
Enterprise edition, 33, 462
EOP (Exchange Online Protection), 16
–eq comparison operator, 110
–Equipment cmdlet parameter, 195
equipment mailboxes
about, 188, 254–255, 265
creating, 195
defining custom properties, 255–256
naming conventions, 178–179
ESE (Extensible Storage Engine)
about, 388–389
DAG and, 466
handling storage failures, 549
memory allocation and, 402–403
timestamps, 425
transaction logs, 419–425, 479
view tables, 400
ESEUTIL utility
about, 57, 445, 450–451
/D parameter, 448
database compaction, 436
/K switch, 482
/MH parameter, 422
/MS parameter, 450
/p switch, 427
/R switch, 482
rebuilding databases, 451
–EstimateOnly cmdlet parameter, 731
event-based assistants, 454
Event Viewer, 471
EWS (Exchange Web Services)
about, 40–41
ESE versus, 389
message tracking, 326
synchronizing information, 792
testing considerations, 28, 30
EWSEditor (Exchange Web Services Editor), 40–41
examining search results, 720–724
$ExBin variable, 101
Exchange Administration Center (EAC)
about, 24
accessing data, 176–177
accessing mailbox audit data, 759
address book policies, 352–353
applying retention policy to mailboxes, 677
certificate management, 313–317
changing columns, 177–178
Compliance Management section, 171, 668, 673, 759
creating DAGs, 490–491
delivery reports, 320–329
development priorities, 4
distribution groups, 267–313
exporting information to CSV files, 181–182
filtering capabilities, 177, 342
Groups section, 302, 331
Hybrid section, 172
I/O performance improvements, 401
importing and exporting mailbox data with, 629–632
in-place holds, 705–706
language support, 196–200
mail flow and rules, 318–320
Mail Flow section, 171–172, 325, 333, 335
Mailbox Delegation section, 229–230, 232–233
mailbox import and export, 629–632

management interfaces, 21–22, 170–173
managing recipients, 180–183
migration batches with, 574, 581–592
Mobile section, 172–173
moving databases and, 544
mysterious mailboxes, 182–183
naming conventions, 178–180
Offline Address Book and, 362
Organization section, 171, 313, 346, 352
permissions, 143–144, 233–234
Permissions section, 133–134, 143–144, 171, 295
Protection section, 171
public folders and, 766
Public Folders section, 172, 188, 767
RBAC support, 132
Recipients section, 146, 171, 177, 182, 241, 250, 273–274, 581
refresh option, 176
replication queue lengths, 481
reporting licenses, 70–71
running without mailboxes, 329–330
Servers section, 172, 315, 409–410, 494, 537
setting diagnostics for Exchange Server, 330–332
setting up mailboxes, 191–193
sharing policies, 313
starting, 173–175
Unified Messaging section, 172
viewing correct, 174
viewing migration batch properties, 595–596
Windows PowerShell and, 84
Exchange Authorization Library (ADDriver), 90–91
Exchange Control Panel (ECP), 21–22, 170, 269, 707
Exchange database (EDB), 388
Exchange Install Domain Servers group, 75
Exchange LegacyInterOP role group, 74
Exchange Management Console (EMC), 21–22, 650
Exchange Management Shell (EMS)
about, 24, 31, 83, 92–120
accessing mailbox audit data, 759
Active Directory for PowerShell, 120–123
adding database copies, 525
address book policies, 352–353
creating dynamic distribution groups, 306–308
delegated setup and, 63
email address policies, 343–345
enabling archives, 648
executing searches, 731–732
mailbox import and export, 632–637
MailTips management, 377–378
management interfaces, 21–22, 170
migration batches with, 574, 592–595
public folders and, 766
room lists, 270, 298
searches for delivery reports, 327–329
useful examples, 123–128
Windows PowerShell, 83–92, 129
Exchange-MBX.xml file, 52
Exchange Online-ApplicationAccount, 76
Exchange Online Protection (EOP), 16
Exchange Online (Service), 5–9
Exchange Organization Administrators security group, 48
Exchange Personal Information property set, 35
Exchange Server
about, 1–2
Active Directory support, 34–37, 45–48
addressing, 333–386
architectural overview, 10–11
creating organization, 49–50
Database Availability Group, 457–566
deploying, 50–63
development priorities, 2–5
editions supported, 22–23, 32–33
Exchange Administration Center, 267–332
Exchange Store, 387–456
influence of cloud service, 5–9
installing, 43–81
mailbox management, 169–266
moving mailboxes, 567–640
preparing for, 26–32, 47–48, 63–73, 77–81
public folders, 765–790
role-based access control, 131–168
setting diagnostics, 330–332
site mailboxes, 791–805
synchronization with SharePoint and, 795–799
upgrading, 11–22, 63–73
utilities supported, 37–41
virtualization, 24–25
Windows PowerShell, 83–130
Exchange Servers role group, 74–75
Exchange Store. *See also* Information Store
about, 19–20
background maintenance, 432–437
corrupt item detection and isolation, 437–448
database management, 407–418
database usage statistics, 451–453
I/O operation, 392–402
mailbox assistants, 454–455
managed availability, 226–227, 407
maximum database size, 389–392
rebuilding databases, 448–451
transaction logs, 419–432
workers, controller, and memory, 402–407
Exchange Trace Analyzer (ExTRA) utility, 57
Exchange Trusted Subsystem role group
about, 74, 77
FSW and, 500
remote PowerShell and, 89
split permissions model, 166–167
Exchange Web Services Editor (EWSEditor), 40–41

Exchange Web Services (EWS)
about, 40–41
ESE versus, 389
message tracking, 326
synchronizing information, 792
testing considerations, 28, 30
Exchange Windows Permissions role group, 74, 166–167
ExchangeGuid property, 107
–ExchangeRemoteMove cmdlet parameter, 612
ExchangeSetup.log file, 56–57
–ExcludeDuplicateMessages cmdlet parameter, 713, 731
–ExcludeFolders cmdlet parameter, 633, 636
executing searches, 731–732
execution policies, PowerShell, 118–119
$ExInstall variable, 101
–ExpansionServer cmdlet parameter, 277
Expert diagnostic level, 331
expiry dates, 437, 684
export. *See* import and export (mailbox)
Export-CSV cmdlet, 125–127, 281–282
Export-PublicFolderStatistics.ps1 script, 789
Export-RetentionTags.ps1 script, 681
$ExScripts variable, 101
Extensible Storage Engine (ESE)
about, 388–389
DAG and, 466
handling storage failures, 549
memory allocation and, 402–403
timestamps, 425
transaction logs, 419–425, 479
view tables, 400
ExternalURL property, 366
ExTRA (Exchange Trace Analyzer) utility, 57

F

F2 key, 95
F4 key, 95
F7 key, 95–96
F8 key, 95
facilities provisioning, 191
FAI (folder-associated item), 207
failover, database copies, 466
Failover Cluster Manager, 463–464, 498
$False variable, 101
Federated Services role, 313
FederatedEmail account, 76, 638
file share witness (FSW), 491, 499–501
–Filter cmdlet parameter, 109–110
filters
address mailboxes in databases, 310–313
client-side, 110–113
custom, 308–313
database, 158–159
dynamic distribution groups, 308–313
EAC, 177
email address policies, 341–345
OPATH, 110, 346
recipient, 180–181, 306, 334, 337, 341–342
server-side, 110–113
firewall rules, 492
–FirstName cmdlet parameter, 192
FirstName property, 185
folder-associated item (FAI), 207
FolderBind action, 756
folders. *See also* public folders
default, 200–201, 393
larger, 124–125
managed, 688–689
naming conventions, 179
retention policy tags, 661
setting retention policy on, 685–687
–FolderScope cmdlet parameter, 746
–ForceGroupMetricsGeneration cmdlet parameter, 376
Forefront Protection for Exchange (FPE), 16
forest functional mode, 26
Format-List cmdlet, 98
Format-Table cmdlet, 98
$FormatEnumerationLimit variable, 100
***Fortune* magazine, 643**
FPE (Forefront Protection for Exchange), 16
FQDN (fully qualified domain name), 90, 314
free disk space, 432, 522
front-end. *See* CAS (Client Access Server)
FSW (file share witness), 491, 499–501
Full Access permission, 226, 230, 232–235
FullScanMoveJobsPollingInterval property, 622
fully qualified domain name (FQDN), 90, 314

G

%g variable, 339
GAL (Global Address List)
about, 35
address book policies and, 348–349, 351–357
address lists and, 346
group naming policy and, 287, 292
mail-enabled contacts, 250–251
mail users, 252
naming mailboxes, 178, 185–187
Offline Address Book and, 360, 367–371
public folders, 775
resource mailboxes, 254–255
tracking group usage, 286
gap coalescing technique, 394–395
Get-ActiveSyncDevice cmdlet, 125–126
Get-ADDomainController cmdlet, 121
Get-AddressList cmdlet, 347
Get-AdminAuditLogConfig cmdlet, 749
Get-ADUser cmdlet, 122
Get-CalendarProcessing cmdlet, 202, 258–259, 261
Get-CASMailbox cmdlet, 112

Get-ClusterNode cmdlet, 564
Get-Command cmdlet, 85, 93–94
Get-Contact cmdlet, 98
Get-Credential cmdlet, 90, 612
Get-DatabaseAvailabilityGroup cmdlet, 467, 493
Get-DatabaseAvailabilityGroupNetwork cmdlet, 507, 509
Get-DistributionGroup cmdlet, 112
Get-DistributionGroupMember cmdlet, 217, 305
Get-DynamicDistributionGroup cmdlet, 112–113, 278, 304, 307–308
Get-EmailAddressPolicy cmdlet, 334, 341–342
Get-EventLog cmdlet, 98
Get-EventLogLevel cmdlet, 331–332
Get-ExchangeDiagnosticInfo cmdlet, 576
Get-ExchangeServer cmdlet
about, 92
finding property values, 410
identities, 105–106
reporting issues, 71–72
selective output, 99
version numbers, 67–68
Get-ExCommand cmdlet, 86
Get-ExecutionPolicy cmdlet, 118
Get-FailedContentIndexDocuments cmdlet, 734
Get-Group cmdlet, 98, 107, 350
Get-HealthReport cmdlet, 471
Get-InboxRule cmdlet, 320
Get-Mailbox cmdlet
about, 83
address book policies, 354–356
applying retention policies, 678
arbitration mailboxes, 59, 547
archive mailboxes, 650
checking language, 196
checking mailboxes, 312, 362–363
checking quotas, 217
EAC support, 174, 182–183
equipment mailboxes, 195, 265
filter examples, 110, 112–113
health mailboxes, 228, 547
larger folders and, 125
limiting number of objects returned, 109–111
multi-domain forests, 123
OAB generation mailboxes, 367
object versions, 69
removing databases, 417–418
returning properties, 98, 108
room mailboxes, 195, 262
selective output, 99–100
server names and, 44
shared mailboxes, 241
Get-MailboxAutoReplyConfiguration cmdlet, 202–203
Get-MailboxCalendarConfiguration cmdlet, 202–204, 261
Get-MailboxCalendarSettings cmdlet, 261
Get-MailboxDatabase cmdlet
about, 92
checking mounted databases, 517
checking OABs, 413
debugging swelling databases, 444
moving database locations, 544
reconnecting mailboxes, 220
removing database copies, 547
removing mailboxes, 510
removing servers, 548
retrieving maintenance schedule, 434
updated status information, 481
viewing activation preferences, 535
white space, 449–450
worker processes, 402
Get-MailboxDatabaseCopyStatus cmdlet
content index, 482
database copy information, 520–522, 536, 542
moving database locations, 545–546
replay queue length, 529
replication status, 516, 518–519
server location, 362
wildcard characters, 481
Get-MailboxExportRequest cmdlet, 626, 637
Get-MailboxExportRequestStatistics cmdlet, 626, 637
Get-MailboxFolder cmdlet, 202, 204, 320
Get-MailboxFolderPermission cmdlet, 240
Get-MailboxFolderStatistics cmdlet, 92, 125, 746, 798
Get-MailboxImportRequest cmdlet, 625, 633–634
Get-MailboxImportRequestStatistics cmdlet, 625, 631, 634–635
Get-MailboxJunkEMailConfiguration cmdlet, 202, 204–205
Get-MailboxMessageConfiguration cmdlet, 202, 204
Get-MailboxRegionalConfiguration cmdlet, 199, 202
Get-MailboxSearch cmdlet, 702, 707, 731
Get-MailboxServer cmdlet, 375, 455, 519
Get-MailboxSpellingConfiguration cmdlet, 202
Get-MailboxStatistics cmdlet
about, 92, 108, 207
accessing move report histories, 600
auditing mailboxes, 184
checking space usage, 652
getting more information about a move, 597
health mailboxes, 228
quarantined mailboxes, 439
reconnecting mailboxes, 222
remote procedure calls and, 128
removing database copies, 547
source of mailbox move information, 599
Get-ManagementRole cmdlet, 136, 138, 142
Get-ManagementRoleAssignment cmdlet, 152–155, 162–164, 296
Get-ManagementRoleEntry cmdlet, 136, 138, 146, 156
Get-Member cmdlet, 112
Get-MessageTrackingLog cmdlet, 286

Get-MigrationUser cmdlet, 594–595
Get-MigrationUserStatistics cmdlet, 585, 595, 599
Get-MonitoringItemIdentity cmdlet, 471
Get-MoveRequest cmdlet
about, 605
checking status of move request, 609
clearing move requests, 610
removing databases, 417–418
suspending mailbox moves, 617
viewing move request properties, 598
Get-MoveRequestStatistics cmdlet
about, 576, 605
percentage of move completed, 176, 598
retrieving updated information, 481
source of mailbox move information, 599
status detail, 579
suspending mailbox moves, 616–618
total mailbox size, 597
troubleshooting with, 599
Get-Notification cmdlet, 638
Get-OfflineAddressBook cmdlet, 362, 364–366
Get-OrganizationConfig cmdlet, 292, 770
Get-PublicFolder cmdlet, 769, 780, 786
Get-PublicFolderMailboxDiagnostics cmdlet, 772
Get-PublicFolderMoveRequest cmdlet, 784
Get-PublicFolderStatistics cmdlet, 773
Get-Queue cmdlet, 564
Get-Recipient cmdlet
archive mailboxes, 650
checking mailboxes, 312
discovering hidden recipients, 357
filters and, 112, 343
validating query results, 305–306
Get-RetentionPolicy cmdlet, 662, 674–675
Get-RetentionPolicyTag cmdlet, 662, 666, 670, 672
Get-RoleGroup cmdlet, 139, 144, 151, 162
Get-RoleGroupMember cmdlet, 144
Get-SearchDocumentFormat cmdlet, 730
Get-SendConnector cmdlet, 70
Get-SiteMailboxDiagnostics cmdlet, 148
Get-StoreUsageStatistics cmdlet, 442, 451–453
Get-ThrottlingPolicy cmdlet, 724
Get-TransportService cmdlet, 324
Get-User cmdlet
bulk updates and, 115–116
filters and, 112–113
handling information returns, 98
variables and, 102, 107
Get-WebServicesVirtualDirectory cmdlet, 613
–GetEffectiveUsers cmdlet parameter, 163
GivenName property, 385
Global Address List (GAL)
about, 35
address book policies and, 348–349, 351–357
address lists and, 346
group naming policy and, 287, 292
mail-enabled contacts, 250–251
mail users, 252
naming mailboxes, 178, 185–187
Offline Address Book and, 360, 367–371
public folders, 775
resource mailboxes, 254–255
tracking group usage, 286
global tables, 400
globally unique identifier (GUID)
active copy of user's mailbox, 44
archive mailboxes, 651
database portability and, 459
default OAB operation, 363
as mailbox identifiers, 107–109
system registry and, 438
target mailboxes, 601
GlobalWebDistributionEnabled property, 366
GoDaddy certificate vendor, 314
Goodman, Steve, 124
Grant Full Access permission, 230
Grant Send As permission, 229
Grant Send On Behalf Of permission, 229
–GrantSendOnBehalfTo cmdlet parameter, 232
Grogan, Andy, 124
group metrics, 373–382
Group Metrics mailbox assistant, 375–376
Group-Object cmdlet, 92–93, 286
Groupbox control type, 385
groups. *See also* distribution groups; role groups
ABPs and, 350
authentication and, 280–281
closed, 270
delegating, 275
group metrics, 373–382
moderated, 244–246, 270, 283
public, 270
with searches, 728–730
searches and, 728–730
security, 252, 284–285
tracking usage, 285–286
user-maintained, 286–299
USGs, 73–77, 139–141, 267
version numbers, 268
viewing members, 268–269, 281–283
GroupType property, 284
–gt operator, 110
GUID (globally unique identifier)
active copy of user's mailbox, 44
archive mailboxes, 651
database portability and, 459
default OAB operation, 363
as mailbox identifiers, 107–109
system registry and, 438
target mailboxes, 601
Guid property, 108

H

HABs (hierarchical address books), 349, 371–373
HardDelete action, 756
hardware
development influenced by, 5
freeing disk space, 432, 522
protection against high latency, 442
recycling, 23
transaction log I/O, 428
health mailboxes, 182–183, 226–229, 547
Health Manager Service process (MSExchangeHMHost.exe), 226
Health Manager Worker process (MSExchangeHMWorker .exe), 226
health sets, 471
Help Desk role group, 74, 135, 141, 145–146
Hidden From Address Lists property, 357
hidden recipients, 357–358
hidden system folder, 786
hierarchical address books (HABs), 349, 371–373
hierarchy, public folders, 766–767
high availability. *See also* DAG (Database Availability Group)
about, 457–459
Active Manager and, 469–472
handling move request errors, 609
lagged copies and, 527
mailbox moves and, 619–621
High diagnostic level, 331
high latency, 442, 508, 550, 553
HighAvailability channel, 503
HoldForMigration parameter, 789
$home variable, 101
HomeMDBBL property, 59
HomeMTA property, 44
Hotmail, 6
HR provisioning, 190
HTML-format messages, 397–398
HTML reports, 127–128
HTTP proxy logs, 367
HTTPS protocol, 8
Hygiene Management role group, 74, 152
Hyper-V, 24–25
hypervisors, 24–25
hyphen (-), 103, 110

I

I/O operations
about, 392–397
database schema, 398–400
high availability and, 457–458
import of PST data, 626
improvements to, 400–402
maintaining contiguity, 392, 394–395, 397–398
memory and, 463
sizing mailboxes, 394
transaction logs, 427–429
%i variable, 339
ICS (Incremental Change Synchronization), 578, 772
identities (cmdlets), 105–109
–Identity cmdlet parameter, 105–109, 116, 690
–IgnoreDefaultScope cmdlet parameter, 123
–IgnoreNamingPolicy cmdlet parameter, 290
–IgnoreNetworkParameter cmdlet parameter, 509
IIS (Internet Information Services), 53, 85, 88–90
–ilike operator, 110
import and export (mailbox)
about, 623–624
with EAC, 629–632
with EMS, 632–637
exporting mailbox data, 635–637
gaining permission to execute, 624–626
planning import of PST data, 626–629
running multiple concurrent, 633
Import-Module cmdlet, 52, 121
Import-PSSession cmdlet, 90
Import-RetentionTags.ps1 script, 681
in-place holds, 703–708, 726–728, 736
Inbox folder, 393, 658
–IncludedRecipients cmdlet parameter, 307, 347
–IncludeFolders cmdlet parameter, 633, 636
IncludeKeywordStatistics property, 713
–IncludeMoveHistory cmdlet parameter, 600
–IncludeMoveReport cmdlet parameter, 601
–IncludeReport cmdlet parameter, 599
–IncludeSystemTags cmdlet parameter, 662
IncludeUnsearchableItems property, 713
IncomingLogCopyingNetwork property, 522
Incremental Change Synchronization (ICS), 578, 772
incremental resynchronization, 487–488
incremental synchronization, 575, 615
incremental updates, 578
indexes
content, 776
secondary, 393, 395, 437, 578
Information Assistant process, 248
Information Store
about, 388
DAG and, 466
improvements in, 19–20
MailTips and, 374
MRS and, 568
processing overview, 402–407
remote procedure calls to, 128
startup process, 421
Information Store Integrity maintenance utility (ISINTEG), 445
initialization scripts, 90, 100
–Initials cmdlet parameter, 192
InPlaceHoldEnabled property, 707, 714

InPlaceHoldPeriod property, 707
Insert key, 95
Insert log record type, 425
Install-WindowsFeature cmdlet, 47, 51, 53
installing Exchange Server
about, 43–44
Active Directory support, 45–48
creating Exchange organization, 49–50
deployment overview, 50–63
domain controllers and, 495
namespace planning, 77–81
security groups and accounts, 73–77
updates and, 63–73
updating mailbox databases, 414–416
InstallWindowsComponent.ps1 script, 52
Integrated Scripting Environment (ISE), 103–105
Integrated Windows Authentication (IWA), 79–80
interfaces, management, 21–22
Internet Information Services (IIS), 53, 85, 88–90
IP addresses, 492–493, 508
IPM_SUBTREE, 775
ISA Server, 6
–IsArchive cmdlet parameter, 637
iSCSI networks, 509
ISE (Integrated Scripting Environment), 103–105
IsExcludedFromInitialProvisioning property, 210
IsExcludedFromProvisioning property, 210
IsExcludedFromProvisioningBySpaceMonitoring property, 210
IsExcludedFromServingHierarchy flag, 771
IsHierarchicalGroup property, 372
ISINTEG (Information Store Integrity maintenance utility), 445
IsMemberOfDL property, 385
isolating corrupt items. *See* corrupt items
IssueWarningQuota property, 214, 803
IsSuspendedFromProvisioning property, 210
IT provisioning, 190–191
IWA (Integrated Windows Authentication), 79–80

J

JET database engine, 511
journaling messages, 249
Junk Email Options Assistant, 454

K

Kerberos authentication, 79
keyboard commands, 95–97
keyword query language (KQL), 710–711, 736
KQL (keyword query language), 710–711, 736

L

Label control type, 385
lagged database copies
about, 525–530
Safety Net and, 531–532
self-maintained, 530–532
large-item limit, 584–585, 590
LastAvailableLogTime property, 521
LastCrashTime system registry setting, 438
LastLogGenerated property, 521
LastLogInfoIsStale property, 521–522
–LastName cmdlet parameter, 192
LastName property, 185
LastNumberOfRecords property, 365
LastStatusTransitionTime property, 521
LastTouched property, 364–365
latency, 442, 508, 550, 553
LCR (local continuous replication), 29
LDAP (Lightweight Directory Access Protocol), 110, 270, 301, 349
LDAPReads metric, 453
LDAPSearches metric, 453
LDIFDE utility, 114, 386
LegacyExchangeDN property, 34, 222
legal compliance. *See* compliance management
licenses, reporting, 70–73
life cycle of site mailboxes, 800–803
Lightweight Directory Access Protocol (LDAP), 110, 270, 301, 349
–like operator, 110
linked mailboxes, 188
linked value replication (LVR), 268–269
Listbox control type, 385
litigation hold (mailboxes), 19, 697, 699–701
Litigation Hold role, 704
–LitigationHoldDate cmdlet parameter, 700
–LitigationHoldOwner cmdlet parameter, 700
Live@EDU service, 6
load balancing
moving mailboxes, 568–569
upgrade considerations, 16
local continuous replication (LCR), 29
Log Record Checksum (LRCK) algorithm, 427
log sets (log streams), 419–427
logical corruption, 526
logical unit number (LUN), 395, 428
–LogLevel cmdlet parameter, 731
–LogonType cmdlet parameter, 760
LogRecordBytes metric, 453
LogRecordCount metric, 453
logs and logging. *See also* transaction logs
Application and Service Logs, 502
Application Event Log, 332, 364, 433, 443, 544
circular, 414, 429–431, 486, 510–514, 554
diagnostic levels, 330–332, 608
fluctuations in number of, 480
HTTP proxy, 367
message-subject, 324
migration batches, 592

placement considerations, 428–429
protection against excessive log growth, 443–444
rebuilding databases, 448–449
reserved, 408, 431–432
setup, 56–57
task, 502
temporary log files, 430–431
lossy failure, 580–581
Lotus Notes, 337, 344
Low diagnostic level, 331
Lowest diagnostic level, 331
LRCK (Log Record Checksum) algorithm, 427
–lt operator, 110
LUN (logical unit number), 395, 428
LVR (linked value replication), 268–269
Lync, integration with, 3, 18
LZ77 compression, 506
LZX compression, 361

M

%m variable, 339
mail-enabled contacts, 179, 250–251
mail-enabled security groups, 284–285
mail-enabling public folders, 779–783
Mail Recipient Creation role, 146–147
Mail Recipients role, 146–147, 156
mail users, 252–253
mailbox assistants, 375–376, 454–455
–Mailbox cmdlet parameter, 783
Mailbox Import Export role, 133, 140, 159, 174, 624
mailbox management. *See also* specific mailboxes
about, 169–170
activating database copies, 533–534
applying retention policy to, 677–681
attributes and names, 185
auditing mailbox access, 239, 754–762
AutoDatabaseMountDial setting, 539–542
automating settings, 205–206
bulk mailbox creation, 195
common problems, 191
completing mailboxes, 193–194
creating mailboxes, 187–209
custom attributes, 207–209
custom filter to address mailboxes in databases, 310–313
data retention states, 745
default folders, 200–201
disabling mailboxes, 218–220, 223
enabling mailboxes for auditing, 757–758
language support, 196–200
mail-enabled contacts, 250–251
mail users, 252–253
mailbox assistants, 375–376, 454–455
mailbox databases, 410–416
mailbox quotas, 213–218, 391–392, 396, 606
mailbox repair cmdlets, 445–448
managing recipients, 180–183
manipulating settings, 202–206
message quotas, 128
MFCMAPI utility and, 38–40
migrating DAGs, 464–465
moderated recipients, 243–250
moving mailboxes, 59–60, 567–640
naming mailboxes, 185–187
need for mailboxes, 183–184
OAB generation mailboxes, 366–367
operational flexibility and, 4
perfection and progress, 170–180
placing mailboxes on hold, 19
provisioning mailboxes, 607
quarantined mailboxes, 439–441
recalling messages, 241–243
reconnecting mailboxes, 220–223
removing mailboxes, 218–220, 223
resources provisioning management agent, 209–213
role assignment policies and, 164–166
setting mailbox permissions, 229–240
setting mailbox quotas, 213–218
sizing mailboxes, 391–394
transport service, 27–28
viewing properties, 176
Mailbox Replication Proxy service (MRSProxy), 612–614
Mailbox Replication Service (MRS)
about, 567–568
archive mailboxes and, 211
autosuspended state for moves, 587
configuration file, 621–623
corrupt item detection and isolation, 440–441
creating mailbox databases, 413–414
ensuring high availability, 620–621
improvements in, 8
mailbox import and export, 623
managing individual mailbox moves, 605–611
migration batches, 584–590, 593
Migration service and, 574
MSExchangeRepl versus, 465, 567
planning import of PST data, 627
planning mailbox moves, 615
preventing loss of data, 580–581
processing overview, 575–578
removing databases, 418
reporting mailbox moves, 599–604
suspending mailbox moves, 616–618
system resources and, 579–580
transaction log replay, 480
Mailbox Search role, 159, 704
Mailbox server role (backend), 1, 314
mailbox tables, 400
Mailbox Transport Submission service, 243
MailboxDatabaseFailureItems channel, 503
MailboxGuid property, 108

MailboxReplicationService policy, 579, 607
MailboxReplicationServiceHighPriority policy, 579, 607
MailboxType property, 593
MailTips
about, 373–376
auto-reply message, 203
client interaction, 376–377
configuring, 377–378
customizing, 380–381
distribution groups, 187
DLP and, 18
mail delivery indication, 251
moderated groups, 245–246
moderated mailboxes, 250
multilingual, 381
OAB support, 371, 381–382
setting up, 253, 275
user experience, 378–379
–MailTipsAllTipsEnabled cmdlet parameter, 377
–MailTipsExternalRecipientTipsEnabled cmdlet parameter, 378
–MailTipsGroupMetricsEnabled cmdlet parameter, 378
–MailTipsLargeAudienceThreshold cmdlet parameter, 378
–MailTipsMailboxSourcedTipsEnabled cmdlet parameter, 378
–MailTipTranslations cmdlet parameter, 381
MailUniversalDistributionGroup, 267
MailUniversalSecurityGroup, 267
maintenance, background. *See* background maintenance
Majority Node Set clusters, 493
Managed Availability (MSExchangeHWWorker)
Active Manager and, 469–472, 536
database redundancy, 522
health mailboxes and, 226–227
Managed Store and, 407
tracking actions, 472
Managed Folder Assistant (MFA)
about, 454
background maintenance, 390
cumulative updates, 65
Email Lifecycle Assistant, 726–727
expired items, 673
PST data and, 629
public folder mailboxes, 773
retention policies and, 653, 655, 665–666, 684–685, 689–696
retention settings for mailboxes, 513
writing policy information into mailbox, 678
managed folders, 688–689
Managed Store. *See* Exchange Store
–ManagedBy cmdlet parameter, 149, 151, 276
ManagedBy property, 113
–ManagedFolderMailboxPolicy cmdlet parameter, 193
ManageGroupManagementRole.ps1 script, 297
Management Committee group, 245
management interfaces, 21–22, 170
management roles. *See* roles
managing databases. *See* database management
–ManualDAGNetworkConfiguration cmdlet parameter, 509
MAPI (Messaging Application Programming Interface)
about, 8
applying retention policies, 689
DAG networks and, 507
LZ77 algorithm and, 506
MFCMAPI utility, 38–40
OAB support, 366
properties supported, 692–693
MaxActiveMovesPerSourceMDB property, 622
MaxActiveMovesPerSourceServer property, 622
MaxActiveMovesPerTargetMDB property, 622
MaxActiveMovesPerTargetServer property, 622
MaxConcurrentMigrations property, 614
maximum database size, 389–392, 614
maximum number of active databases, 402–403
maximum number of open site mailboxes, 800
MaximumActiveDatabases property, 402–404
MaxMoveHistoryLength property, 623
MaxReceiveSize property, 803
MaxRetries property, 623
–MaxSendSize cmdlet parameter, 584–585, 605, 627
MaxTotalMovesPerMRS property, 622
Medium diagnostic level, 331
meeting requests
meeting room conflicts, 260
processing, 261–265
room lists, 270, 298–299
MemberDepartRestriction property, 272, 284
MemberJoinRestriction property, 272, 284, 294
memory management
Exchange Store, 402–407
I/O operations and, 463
mailbox searches and, 726
PowerShell memory limits, 111
Message Tracking role, 137–138, 141, 147, 323
MessageBind action, 756–757
messages. *See also* addressing messages
approving, 274
auto-reply, 187
changes in submission, 489
compressing, 398
FAI, 207
health mailboxes, 227, 229
journaling, 249
message-subject logging, 324
moderated groups and, 245
processing moderation requests, 247–249
recalling, 241–243
recovering, 643
sending on behalf of other users, 237–239
troubleshooting, 443

MessageTrackingReportId property, 328
MessageTypes property, 713
Messaging Application Programming Interface (MAPI)
about, 8
applying retention policies, 689
DAG networks and, 507
LZ77 algorithm and, 506
MFCMAPI utility, 38–40
OAB support, 366
properties supported, 692–693
messaging connectors, 179
messaging records management (MRM)
about, 18, 641, 657
archive mailboxes and, 650
creating retention policies, 673–677
creating retention tags, 668–673
designing retention policies, 663–665
Managed Folder Assistant, 665–666
naming retention tags, 666–668
system tags, 662–663
types of retention tags, 657–662
metadata, 578, 712
MFA (Managed Folder Assistant)
about, 454
background maintenance, 390
cumulative updates, 65
Email Lifecycle Assistant, 726–727
expired items, 673
PST data and, 629
public folder mailboxes, 773
retention policies and, 653, 655, 665–666, 684–685, 689–696
retention settings for mailboxes, 513
writing policy information into mailbox, 678
MFCMAPI utility
about, 38–40
accessing mailbox audit data, 759
deferred action messages, 319
quarantined mailboxes and, 439–440
validating dates, 696
viewing properties, 693, 796–797
Microsoft Business Productivity Online Service, 6
Microsoft Exchange Replication service
Active Manager and, 465–479
handling storage failures, 549–550
MRS versus, 465, 567
verifying transaction logs, 482
Microsoft Exchange Security Groups OU, 73, 75, 148
Microsoft Exchange Server 2013. *See* Exchange Server
Microsoft Exchange Service Host process, 572–573
Microsoft Exchange System Objects OU, 781
Microsoft Exchange Transport Log Search service, 326
Microsoft Federation Gateway, 313
Microsoft Internet Information Services, 53, 85, 88–90
Microsoft Lync, integration with, 3, 18
Microsoft Management Console (MMC), 21, 36, 170, 382
Microsoft SharePoint, 3, 18, 389
Microsoft System Center Data Protection Manager, 527
Microsoft System Center Operations Manager, 442–443, 470, 520
Microsoft Technology Adoption Program, 14, 68
Microsoft Unified Communications Managed API, 53
migrating DAGs, 464–465
Migration account, 76
migration arbitration mailbox, 573
Migration Batch Wizard, 588
migration batches
clearing, 584
completing, 590
controlling processing, 615–616
with EAC, 574, 581–592
with EMS, 574, 592–595
logging, 592
removing, 584
removing move requests, 591, 610–611
viewing details, 589
viewing properties of, 595–596
migration endpoints, 611–612
Migration role, 148
Migration service
about, 570, 572–575
logging activity, 592
managing individual mailbox moves, 605–611
managing migration batches with EAC, 581–590
managing migration batches with EMS, 592–595
planning mailbox moves, 614–615
MMC (Microsoft Management Console), 21, 36, 170, 382
mobile access to public folder mailboxes, 786
mobile devices, 786
MobilePhone property, 154
moderated groups, 244–246, 270, 283
moderated mailboxes, 249–250
moderated recipients, 243–250
moderation requests, 247–249
modern public folders, 20–21, 420–421, 787–790
monitoring database copies, 517–522
Mount-Database cmdlet, 411–412, 512, 546
–MountDialOverride cmdlet parameter, 539
mounted databases, 407, 426–427, 462–463, 546
Move action, 756
Move-ActiveMailboxDatabase cmdlet, 538–541
Move-DatabasePath cmdlet, 415, 545–546
Move-Mailbox cmdlet, 606
Move Mailboxes role, 142, 148, 152
move reports
about, 597–598
accessing histories, 599–606
outputting to files, 600–601
move requests
checking and altering status, 609–611

clearing, 610–611
enforcing age limits for, 610–611
handling errors, 607–609
removing, 591, 610–611
viewing properties, 598
move tags, 672
MoveToArchive action, 672
MoveToDeletedItems action, 756
moving databases
moving locations within DAGs, 544–546
potential issues, 539–542
server health and, 536
transaction logs and, 480
moving mailboxes
about, 567–571
alerts and, 638–639
assigning priority, 607
asynchronous moving, 571–572
checking and altering move request status, 609–610
creating mailbox databases, 413–414
enabling MRSProxy, 612–614
ensuring high availability, 619–621
handling move request errors, 607–609
import and export processes, 623–637
managing individual moves, 605–611
migration batches with EAC, 581–592
migration batches with EMS, 592–595
migration endpoints, 611–612
Migration service, 572–575
MRS and, 567–568, 575–581, 621–623
planning, 614–619
removing databases, 418
reporting moves, 595–604
scheduling, 617
suspending, 615–619
transaction log replay, 480
transaction logs, 578
variations in speed, 619
moving public folders, 783–785
MRM (messaging records management)
about, 18, 641, 657
archive mailboxes and, 650
creating retention policies, 673–677
creating retention tags, 668–673
designing retention policies, 663–665
Managed Folder Assistant, 665–666
naming retention tags, 666–668
system tags, 662–663
types of retention tags, 657–662
MRS (Mailbox Replication Service)
about, 567–568
archive mailboxes and, 211
autosuspended state for moves, 587
configuration file, 621–623
corrupt item detection and isolation, 440–441
creating mailbox databases, 413–414
ensuring high availability, 620–621
improvements in, 8
mailbox import and export, 623
managing individual mailbox moves, 605–611
migration batches, 584–590, 593
Migration service and, 574
MSExchangeRepl versus, 465, 567
planning import of PST data, 627
planning mailbox moves, 615
preventing loss of data, 580–581
processing overview, 575–578
removing databases, 418
reporting mailbox moves, 599–604
suspending mailbox moves, 616–618
system resources and, 579–580
transaction log replay, 480
MRSProxy (Mailbox Replication Proxy service), 612–614
MsExchange Database counter, 435
MSExchange Database/Instances/DB Maintenance IO Reads/sec counter, 449
MsExchangeDAGMgmt.exe (Database Availability Group management service), 503
MSExchangeHMHost.exe (Health Manager Service process), 226
MSExchangeHMWorker.exe (Health Manager Worker process), 226
msExchangeHomeServerName property, 44
MSExchangeHWWorker (Managed Availability)
Active Manager and, 469–472, 536
database redundancy, 522
health mailboxes and, 226–227
Managed Store and, 407
tracking actions, 472
MSExchangeRepl (Replication service)
Active Manager and, 465–479
handling storage failures, 549–550
MRS versus, 465, 567
verifying transaction logs, 482
MSExchangeServicesAppPool setting, 614
msExchDataPath property, 236
msExchDelegateLinkList property, 236–237
msExchESEParamCacheSizeMax property, 405
msExchESEparamCacheSizeMin property, 406
msExchExtensionAttribute property, 385
msExchGroupExternalMemberCount property, 374
msExchGroupMemberCount property, 374
msExchMailboxMoveBatchName property, 577
msExchMailboxMoveFlags property, 577
msExchMailboxMoveRemoteHostName property, 577
msExchMailboxMoveSourceMDBLink property, 577
msExchMailboxMoveStatus property, 577
msExchMailboxMoveTargetMDBLink property, 577
msExchMaxActiveMailboxDatabases property, 34
MSExchMDBAvailabilityGroupLink property, 496

MSExchQueryFilter property, 301
multi-domain forest, 122–123
multi-forest design, 45–46, 172
Multivalued Listbox control type, 385
MyBaseOptions role, 164
MyContactInformation role, 164
MyDistributionGroupMembership role, 165, 350
MyDistributionGroups role, 165, 290, 295–296
MyMarketPlaceApps role, 165
MyProfileInformation role, 164
MyRetentionPolicies role, 165, 682
MyTeamMailboxes role, 165
MyTextMessaging role, 165
MyVoiceMail role, 164

N

–Name cmdlet parameter, 192, 251, 253, 307, 348, 731
Name property, 185
name resolution, 365–366
Name Service Provider interface (NSPI), 365
named parameters, 97
namespace planning
about, 77
handling connections with CAS, 79–80
protocol-specific, 80–81
self-signed certificates, 77–78
naming conventions
about, 43–44
applying different, 186
EAC, 178–180
mailboxes, 185–187
public folder mailboxes, 768
retention tags, 666–668
retroactive, 180
underscores, 26
user-maintained groups, 289–295
NDRs (nondelivery reports), 420
–ne comparison operator, 110
.NET Framework, installing, 44
network interface cards (NICs), 458, 499, 506–508
NetworkCompression property, 505
NetworkEncryption property, 505
networks
APIPA addressing, 492
DAG, 506–509
iSCSI, 509
transient conditions, 483
New Address List screen, 347, 353
New-AddressBookPolicy cmdlet, 354
New-AdminAuditLogSearch cmdlet, 751
New-DatabaseAvailabilityGroup cmdlet, 490, 493, 498, 508
New-DistributionGroup cmdlet, 295–296, 298
New-DynamicDistributionGroup cmdlet, 306–308, 310–311
New Email Address Policy screen, 337–338, 342
New-EmailAddressPolicy cmdlet, 343
New-FederationTrust cmdlet, 313
New-GlobalAddressList cmdlet, 353
New-InboxRule cmdlet, 319–320
New-Item cmdlet, 120
New-Mailbox cmdlet
about, 206
archive mailboxes, 648
database scope, 159
discovery mailboxes, 225
equipment mailboxes, 195
OAB generation mailboxes, 367
properties supported, 192–193
remote PowerShell, 91
room mailboxes, 195
New-MailboxAuditLogSearch cmdlet, 755, 759–760
New-MailboxDatabase cmdlet, 129, 212, 412
New-MailboxExportRequest cmdlet, 160, 625, 636, 722
New-MailboxFolder cmdlet, 202, 204
New-MailboxImportRequest cmdlet
about, 625, 632
importing PST data, 635
role assignments and, 160
tickles, 576
New-MailboxRepairRequest cmdlet, 446–447
New-MailboxRestoreRequest cmdlet, 581, 785
New-MailboxSearch cmdlet, 702–703, 731–732
New-MailContact cmdlet, 251
New-MailUser cmdlet, 253
New-ManagementRole cmdlet, 156, 161
New-ManagementRoleAssignment cmdlet, 157, 160–162, 323, 625
New-ManagementScope cmdlet, 143, 158–159
New-MigrationBatch cmdlet, 592–594, 611, 639, 790
New-MigrationEndpoint cmdlet, 612, 614
New-MoveRequest cmdlet
about, 148, 605–606
database scope and, 159
discovery mailboxes, 226
enforcing age limits for move requests, 610–611
immediate processing, 440
mailbox provisioning and, 607
migration processing, 573, 575, 587
suspending mailbox moves, 616, 618
New-OfflineAddressBook cmdlet, 354
New-PSSession cmdlet, 89–90
New-PublicFolder cmdlet, 777, 783
New-PublicFolderMigrationRequest cmdlet, 789
New-PublicFolderMoveRequest cmdlet, 773, 784
New-RetentionPolicyTag cmdlet, 661, 671, 673–674, 689
New-RoleGroup cmdlet, 145, 148, 158
New-SiteMailbox cmdlet, 148, 801
New-TestCASConnectivityUser.ps1 script, 183
New-ThrottlingPolicy cmdlet, 725
NICs (network interface cards), 458, 499, 506–508
noderunner processes, 406

nondelivery reports (NDRs), 420
nonsystem tags, 662–663
notification, email, 595–596
–NotificationEmails cmdlet parameter, 594
–notlike operator, 110
NSPI (Name Service Provider interface), 365
NTFS file system, 389
$Null variable, 101, 680

O

OAB generation mailboxes, 366–367
OAB (Offline Address Book)
about, 357–358
address book policies, 349, 354–355
contents of, 365–366
creating customized, 367–371
creating mailboxes, 366–367
default operation, 363–365
email address policies, 340
generating, 362–367
group metrics, 375
mailbox databases and, 412–413
MailTips support, 378, 381–382
mixed environment, 360–361
Outlook and, 359–362
user-maintained groups, 288
OABGeneratorAssistant, 363–365
OABGeneratorWorkCycle property, 363
OABGeneratorWorkCycleCheckPoint property, 363
Oab.xml file, 361
object versions, 69–70
ObjectModified property, 752
OffCAT (Office Configuration Analyzer Tool), 37
Office 365
about, 349
archive and, 649
influence of cloud service, 9
Managed Availability, 469
motivation for upgrading, 12
public folder mailboxes, 773
site mailboxes and, 800
Office Configuration Analyzer Tool (OffCAT), 37
Office property, 351–352
Offline Address Book (OAB)
about, 357–358
address book policies, 349, 354–355
contents of, 365–366
creating customized, 367–371
creating mailboxes, 366–367
default operation, 363–365
email address policies, 340
generating, 362–367
group metrics, 375
mailbox databases and, 412–413
MailTips support, 378, 381–382
mixed environment, 360–361
Outlook and, 359–362
user-maintained groups, 288
offline defragmentation, 448
OLD defragmentation, 435
OnComplete event, 206
online defragmentation, 435, 448
online repair cmdlets, 445–448
OOF (out-of-office) notices, 371, 374, 377–378, 749
OPATH filters, 110, 346
OPATH queries, 268, 270, 300–302
Operational channel, 503
operational processes, testing, 29–30
OR operator, 705
organization mailboxes, 362
Organization Management role group
about, 75, 134–135, 151
administrator searches and, 322
delegating roles, 140, 163
EAC and, 174–175
managing DAG properties, 504
public folders, 774
RBAC Address Lists role, 346
remote Windows PowerShell, 87
role assignments, 152, 160
security groups, 284
special roles, 159–160
organizational forms library, 786–787
organizational units (OUs)
hierarchical address books, 372
mail-enabled accounts, 75–76
mail-enabled objects, 34
security groups and, 73
setting role scopes, 157
–OrganizationalUnit cmdlet parameter, 193, 307
OrganizerInfo property, 258
OST files, 585, 627
OUs (organizational units)
hierarchical address books, 372
mail-enabled accounts, 75–76
mail-enabled objects, 34
security groups and, 73
setting role scopes, 157
Out-HTML cmdlet, 127
out-of-office (OOF) notices, 371, 374, 377–378, 749
Out-String cmdlet, 100
Outlook
archive mailboxes, 649
delivery reports, 321
expired items and, 673
group membership and, 281–282
MailTips and, 378–379
maintaining rules, 319
managing groups, 269–270
MFCMAPI utility and, 38–40

moderated groups, 283
Offline Address Book and, 359–362
retention policies, 684–686
retention tags, 666–667, 676–677
room list groups, 299
site mailboxes and, 792, 799–800
troubleshooting, 443
upgrading versions, 11, 13, 16
Outlook Web App (OWA)
about, 11
accessing personal tags, 662–663
address book policies, 356
archive mailboxes, 649
delegating access, 239–240
delivery reports, 321–322
EAC and, 177
expired items and, 673
group membership and, 281
language support, 196–200
MailTips settings, 377
MailTips support, 378–379
maintaining rules, 319
managing groups, 276, 287
opening another user's mailbox, 237
Options section, 154–156, 269, 276, 287, 331
Organize Email section, 321
People section, 348
retention policies, 684–686
retention tags, 666–667, 676
room list groups, 299
security groups, 284
Windows PowerShell and, 84
–OverrideContentMailbox cmdlet parameter, 784–785
OWA (Outlook Web App)
about, 11
accessing personal tags, 662–663
address book policies, 356
archive mailboxes, 649
delegating access, 239–240
delivery reports, 321–322
EAC and, 177
expired items and, 673
group membership and, 281
language support, 196–200
MailTips support, 377–379
maintaining rules, 319
managing groups, 276, 287, 350
opening another user's mailbox, 237
Options section, 154–156, 269, 276, 287, 331
Organize Email section, 321
People section, 348
retention policies, 684–686
retention tags, 666–667, 676
room list groups, 299
security groups, 284
Windows PowerShell and, 84
ownership of groups, 271, 275–277, 350

P

page zeroing, 433, 436
PagePreRead metric, 453
PageRead metric, 453
PAM (Primary Active Manager) role, 466–467
parameters. *See also* specific cmdlet parameters
named, 97
positional, 97, 105
Partial Attribute Set (PAS), 35
PAS (Partial Attribute Set), 35
passive database copies, 460, 544
–Password cmdlet parameter, 192
passwords, mailbox, 190–191
Patches system registry setting, 69
Performance Monitor, 435, 449, 517, 520
–Permanent cmdlet parameter, 220
PermanentlyDelete action, 672
permissions
discovery mailboxes, 226
EAC, 143–144, 233–234
mailbox import and export, 624–626
preparing Active Directory, 48
public folders, 775
revoking, 232
roles and, 139
setting for mailboxes, 229–240
split permissions model, 166–167
testing operational processes, 29–30
personal archive. *See* archive mailboxes
personal tags
about, 658, 660
accessing through OWA, 682
applying to items, 685–686
creating, 671–672
naming, 666–668
pipelines, 101–102, 109, 678
PKI (public key infrastructure), 315
planning
import of PST data, 626–629
mailbox moves, 614–619
policies. *See also* retention policies
address book, 348–357
EAC, 313
email address, 333–345
execution, 118–119
group naming, 287, 289–295
mailbox creation, 190–191
MRS, 579
processing meeting requests, 261–265
removing tags from, 687–688
Resource Booking Attendant and, 256–261

role assignment, 133–134, 153–155, 163–166
sharing, 313
site mailboxes, 803–804
user-maintained groups, 289–295
portability, database, 459–464
positional parameters, 97, 105
PowerShell. *See* Windows PowerShell
PR_ARCHIVE_DATE property, 693
PR_ARCHIVE_PERIOD property, 693
PR_ARCHIVE_TAG property, 693
pre-staging Cluster Name Object, 497–499
precoded filters, 310
preparing for Exchange Server
about, 26–27
Active Directory, 47–48
installation considerations, 63–73
namespace planning, 77–81
test plans, 28–29
testing for operational processes, 29–30
testing for programming and customizations, 30–31
transport considerations, 27–28
updating versions, 31–32
–PreserveMailboxSizeLimit cmdlet parameter, 606
preserving information
about, 696–698
creating searches, 708–716
examining search results, 720–724
Exchange searches, 733–736
groups with searches, 728–730
in-place holds, 703–708
putting mailboxes on litigation hold, 699–701
putting mailboxes on retention hold, 698–699
removing searches, 730–731
resource throttling for searches, 724–726
retrieving discovered content, 716–720
search syntax, 736–737
searching mailbox content, 701–703
PreviewResultsLink property, 714
Primary Active Manager (PAM) role, 466–467
PrimarySMTPAddress property, 185
priorities
assigning for moves, 607
development, 2–5
email policy, 334–336, 340
privacy
challenges of, 4–5
complying with laws, 208
ProcessExternalMeetingMessages property, 262, 264
Products system registry setting, 69
$Profile variable, 120
profiles, 119–120
ProhibitSendQuota property, 115, 214
ProhibitSendReceiveQuota property, 214, 771, 803
properties. *See* specific properties
protected distribution groups, 278–281
protocols and required certificates, 314
ProvisionedFolder corruption type, 446
provisioning
mailboxes, 607
servers, 63
site mailboxes, 803–804
ProxyAddresses property, 385
PR_POLICY_TAG property, 693
PR_RETENTION_DATE property, 692
PR_RETENTION_FLAGS property, 693
PR_RETENTION_PERIOD property, 692
PR_ROAMING_XMLSTREAM property, 691
PR_START_DATE_ETC property, 692, 694
PST Capture Tool, 628
PST files
about, 201
archive mailboxes and, 644–647
importing data directly into archives, 635
legitimate movement, 443
mailbox import and export, 623–624, 629–637
planning import, 626–629
public folder mailboxes
about, 766
calculating size, 772–773
controlling the root, 774–776
creating, 767–770
mobile access, 786
naming conventions, 768
number needed, 770–772
redirecting content, 785–786
writeable hierarchy, 770
Public Folder Management role group, 75, 135, 774
public folders
about, 765–766, 804
accessing, 778
content indexing and, 776
creating, 776–779
creating mailboxes, 767–770
hierarchy in, 766–767
importing-exporting data in, 624
mail-enabling, 779–783
modern, 20–21, 420–421, 787–790
moving, 783–785
naming conventions, 179
new concepts, 766–767
organizational forms library, 786
permissions, 775
removing, 786
transaction logs and, 420–421
Public Folders container, 345
public groups, 270
public key infrastructure (PKI), 315
PublicFolderMigrationComplete flag, 790
PublicFolderToMailboxMapGenerator.ps1 script, 789
Purges folder, 746
$pwd variable, 101

Q

QBDG suffix, 179
quarantined mailboxes, 439–441
queries
OPATH, 268, 270, 300–302
validating results, 305–306
queue lengths, 481, 517, 529
queues, MRS, 576
quorums
dynamic, 494
FSW role and, 499
QuotaNotificationSchedule property, 215
quotas
accommodating imported data, 626
dealing with I/O, 396
ensuring sufficiency of, 606
Recoverable Items, 746–747
setting, 213–218
site mailboxes, 803
sizing mailboxes, 391–392
quotation marks ("), 102–103

R

RBAC Address Lists role, 346
RBAC Discovery Management role group, 704
RBAC (role-based access control)
about, 85, 131–136
Active Directory, 35
execution policies and, 118
figuring out, 167–168
mailbox import and export, 624–626
remote Windows PowerShell, 86–92
role assignment, 133–134, 152–166
role entries and, 134
role groups and, 133, 139–141
role scope and, 134, 141–143, 409
roles and, 133, 136–139
security groups, 284
split permissions model, 166–167
testing for operational processes, 29–30
universal security groups, 73
working with, 143–152
RCA (Remote Connectivity Analyzer), 37
read-status tracking, 324
rebalancing database copies, 534–536
rebuilding databases, 448–451
Recall Message feature, 241–243
Recipient Management role group
about, 75
administrator searches and, 322
creating security groups, 284
linking to tasks, 135
remote PowerShell and, 91
Recipient Policies role, 148
Recipient read scope, 141
Recipient write scope, 142
–RecipientContainer cmdlet parameter, 307–308
RecipientContainer property, 301
–RecipientFilter cmdlet parameter, 309–311, 347
RecipientFilter property, 301
–RecipientOrganizationalUnitScope cmdlet parameter, 157
–RecipientPathFilter cmdlet parameter, 328
recipients
about, 180
filtering, 180–181, 306, 334, 338, 341–342
hidden, 357–358
moderated, 243–250
Recipients property, 713
RecipientType property, 284
RecipientTypeDetails property, 228, 284, 802
reconnecting mailboxes, 220–223
Records Management role group, 75, 136, 322
Recoverable Items structure
about, 737
asynchronous moving, 571
function of, 737–739
improvements, 739–742
managing quotas for, 746–747
retention periods and, 671
single-item recovery, 743–745
viewing items in, 745–746
RecoverableItemsQuota setting, 757
recovering failed servers, 61–63
recovering messages, 643
–Recurse cmdlet parameter, 769
recycling hardware, 23
Redirect-Message cmdlet, 564
redirecting content to public folder mailboxes, 785–786
RedistributeActiveDatabases.ps1 script, 535–536
redundancy, database, 411, 522
registry, system. *See* system registry
regulatory compliance. *See* compliance management
released to manufacturer (RTM), 13–14
relocating databases, 480, 536, 539–542, 544–546
Remote Connectivity Analyzer (RCA), 37
remote management, 492
remote procedure call (RPC)
avoiding access errors, 492
delivering messages, 28
MAPI, 8, 506
retrieving mailbox information, 128
remote Windows PowerShell
about, 86–88
connecting to, 88–91
limiting user functionality, 91–92
RemoteExchange.ps1 script, 90, 119–120
RemoteSigned execution policy, 118
Remove-ADPermission cmdlet, 232

Remove-DatabaseAvailabilityGroupNetwork cmdlet, 509
Remove-DatabaseAvailabilityGroupServer cmdlet, 494, 548–549
Remove-DistributionGroup cmdlet, 297
Remove-Mailbox cmdlet, 155, 219–220, 223
Remove-Mailbox Permission cmdlet, 205
Remove-MailboxDatabase cmdlet, 418, 548
Remove-MailboxDatabaseCopy cmdlet, 544, 547
Remove-MailboxExportRequest cmdlet, 626
Remove-MailboxImportRequest cmdlet, 625, 635
Remove-MailboxSearch cmdlet, 730–731
Remove-ManagementRoleAssignment cmdlet, 165
Remove-MigrationBatch cmdlet, 594
Remove-MigrationUser cmdlet, 591
Remove-MoveRequest cmdlet
about, 591
clearing move requests, 610
migration batches with EAC, 584–585, 605
removing databases, 418
suspending mailbox moves, 619
Remove-PublicFolder cmdlet, 786
Remove-PublicFolderMoveRequest cmdlet, 785
Remove-RetentionPolicy cmdlet, 688
Remove-RetentionPolicyTag cmdlet, 671–672
Remove-RoleGroup cmdlet, 152
Remove-RoleGroupMember cmdlet, 150
Remove-StoreMailbox cmdlet, 219–220
RemovePrivateProperty property, 258
removing
database copies, 546–548
databases, 416–418
mailboxes, 218–220, 223
migration batches, 584
move requests from migration batches, 591
public folders, 786
retention policies, 688
searches, 730–731
servers, 548–549
tags from policies, 687–688
renaming databases, 414–415
repair cmdlets, 445–448
repairing Exchange Server, 61
Replace log record type, 425
replay, transaction log. *See* transaction log replay
ReplayLagManagerNumAvailableCopies system registry setting, 531
ReplayLagPlayDownPercentDiskFreeSpace system registry setting, 531
ReplayLagStatus property, 522
ReplayLagTime property, 527–528
ReplayQueueLength property, 521
replication
block mode, 484–486
database, 390, 411, 459
public folders and, 767
queue lengths, 481, 517
transaction log replay and, 479–489
Replication service (MSExchangeRepl)
Active Manager and, 465–479
handling storage failures, 549–550
MRS versus, 465, 567
verifying transaction logs, 482
ReplicationPort property, 505
reporting licenses, 70–73
reporting mailbox moves
about, 595–596
accessing move report histories, 599–604
getting more information about moves, 597–598
troubleshooting, 599
reports
delivery, 320–329
HTML, 127–128
representational state transfer (REST), 792
REQ file, 317
RequestInPolicy property, 262–263
RequestOutofPolicy property, 262, 264
–RequireSenderAuthenticationEnabled cmdlet parameter, 280
Reread Logon Quotas interval registry setting, 217, 235
reseeding database copies, 523–525
reserved logs, 408, 431–432
Reset Password role, 137
–ResetPasswordOnNextLogon cmdlet parameter, 192
resilience equation, 558–559
Resource Booking Assistant, 454
Resource Booking Attendant, 256–262, 265, 299
resource forest design, 46
resource mailboxes
about, 188, 254–255
creating, 195
defining custom properties, 255–256
naming conventions, 178–179
resource throttling for searches, 724–726
ResourceCapacity property, 255
ResourceCustom property, 255
resources provisioning management agent, 209–213
REST (representational state transfer), 792
Restore-MailboxExportRequest cmdlet, 576
Restricted execution policy, 118
–ResultSize cmdlet parameter, 109–111, 328
Resume-ClusterNode cmdlet, 565
Resume-MailboxDatabaseCopy cmdlet, 529, 546
Resume-MailboxExportRequest cmdlet, 626
Resume-MailboxImportRequest cmdlet, 625
Resume-MoveRequest cmdlet, 605, 608, 617
Resume-PublicFolderMigrationRequest cmdlet, 790
Resume-PublicFolderMoveRequest cmdlet, 784
resynchronization
after moving mailboxes, 587
incremental, 487–488

retention date calculation, 693–696
retention hold (mailboxes), 19, 697–699
retention policies
applying to mailboxes, 677–681
compliance management and, 644, 646
creating, 663, 673–677
customizing for specifc mailboxes, 681–683
designing, 663–665
Managed Folder Assistant and, 653, 655, 665–666, 689–696
managing in hybrid environment, 681
modifying, 681
removing, 688
setting on folders, 685–687
user interaction with, 684–685
retention policy tags (RPT), 658–659, 666
retention tags
changing, 688
creating, 668–673
disabling, 687
managing in hybrid environment, 681
MRM and, 657
naming, 666–668
removing from policies, 687–688
troubleshooting, 671
types of, 657–662
RetentionAction property, 670
RetentionComment property, 680, 700
–RetentionEnabled cmdlet parameter, 672
RetentionEnabled property, 666
RetentionURL property, 678, 680, 700
RetryDelay property, 623
revoking permissions, 232
Rich Text Format (RTF), 397–398
Richard, Pat, 124
role assignment policies, 133–134, 153–155, 163–166
role assignments
about, 133, 152–163
controlling, 153
default, 141
delegated, 160
role-based access control (RBAC)
about, 85, 131–136
Active Directory, 35
execution policies and, 118
figuring out, 167–168
mailbox import and export, 624–626
remote Windows PowerShell, 86–92
role assignment, 133–134, 152–166
role entries and, 134
role groups and, 133, 139–141
role scope and, 134, 141–143, 409
roles and, 133, 136–139
security groups, 284
split permissions model, 166–167
testing for operational processes, 29–30
universal security groups, 73
working with, 143–152
role entries, 134
role groups. *See also* specific role groups
about, 133
adding roles to, 147–149
creating, 145–149
default, 141
identifying membership, 162–163
linking to tasks, 135–136
maintaining membership, 149–150
managing, 151–152
specific scopes for, 157–158
universal security groups and, 139–141
role scope, 134, 141–143, 157–158, 409
roles
about, 133, 136–139
adding to role groups, 147–149
cmdlets and, 139
creating for specific tasks, 155–157
delegating, 140, 163, 625
EAC and, 175
end-user, 164–165
scopes and, 141–143
unscoped, 160–162
–Room cmdlet parameter, 195
room lists, 270, 298–299, 349
room mailboxes
about, 188, 254–255
creating, 195
defining custom properties, 255–256
naming conventions, 178–179
viewing delegates, 263
–RoomList cmdlet parameter, 298
ROPCount metric, 453
RPC (remote procedure call)
avoiding access errors, 492
delivering messages, 28
MAPI, 8, 506
retrieving mailbox information, 128
RPT (retention policy tags), 658–659, 666
RTF (Rich Text Format), 397–398
RTM (released to manufacturer), 13–14
rule processing, 318–320, 492

S

S/MIME (Secure Multipurpose Internet Mail Extensions), 734
%s variable, 339
Safety Net, 459, 503, 531–532
SAM (Standby Active Manager), 190
SAM (Standby Active Manager) role, 466
SAN (storage area networks), 557
Sarbanes–Oxley Act, 641
SCC (single copy cluster), 29
schedules, maintenance, 434

Scheduling Assistant, 454
scheduling mailbox moves, 617
scheduling threads, 550
Schema Administrators group, 48
schemas
  Active Directory information, 35–36
  database, 398–400
  preparing, 47–48
SCOM (System Center Operations Manager), 442–443, 470, 520
SCR (standby continuous replication), 29
scripting agent, 205
ScriptingAgentConfig.xml file, 205–206
scripts. *See also* specific scripts
  calling, 117
  cmdlets in, 93, 117
  conditional checking, 103
  initialization, 90, 100
  outputting CSV files, 125–127
  pipelines and, 101–102
  writing, 93
Search-ADAccount cmdlet, 122
Search-AdminAuditLog cmdlet, 751–752
Search Foundation, 578, 701, 733–735
Search-Mailbox cmdlet, 703, 731
Search-MailboxAuditLog cmdlet, 755, 759
Search-MessageTrackingReport cmdlet, 326–328
–SearchDumpster cmdlet parameter, 731
SearchDumpster property, 713
searches
  concurrent, 733
  creating, 708–716
  for delivery reports, 321–329
  eDiscovery, 224–225, 233, 702, 726
  examining results, 720–724
  Exchange support, 733–736
  executing, 731–732
  groups with, 728–730
  mailbox content, 701–703
  memory limitations and, 726
  refining, 712–716
  removing, 730–731
  resource throttling for, 724–726
  Search Foundation processes, 578
  syntax for, 736–737
SearchFolder corruption type, 446
–SearchQuery cmdlet parameter, 731
SearchQuery property, 713
secondary indexes (views), 393, 395, 437, 578
Secure Multipurpose Internet Mail Extensions (S/MIME), 734
Secure Sockets Layer (SSL), 30, 77, 313–314
security. *See also* permissions
  administrators and, 319–320
  challenges of, 4–5
  cmdlets and, 89
  installation and, 73–77
  mailboxes and, 184
  PST files and, 646
  self-signed certificates, 78
security groups
  about, 284–285
  mail users and, 252
security support provider (SSP), 506
seeding databases, 489
Select-Object cmdlet, 98, 282
self-maintained lagging, 530–532
self-signed certificates, 77–78, 314, 317
Send As permission, 229–231, 275
Send-NewUserWelcome.ps1 script, 194
Send On Behalf Of permission, 229–231, 241, 275
SendAs action, 756
Senders property, 713
SendOnBehalf action, 756
Sent Items folder, 393, 658
separate rules functionality, 318
–Server cmdlet parameter, 86, 207
Server Management role group, 75, 135
server message block (SMB), 500
server-side filters, 110–113
server-side rules, 318
server switchovers, 536–539
ServerManager PowerShell module, 52
ServerManagerCmd cmdlet, 52
ServerManagerCmd PowerShell module, 52
servers
  edge, 48
  maintaining, 562–565
  mounted databases, 463
  provisioning, 63
  recovering failed, 61–63
  removing, 548–549
  stressed, 565–566
  virtual, 24–25
  witness, 490–491
Service (Exchange Online), 5–9
service level agreement (SLA), 9
service packs (SPs), 64
Set-AdminAuditLogConfig cmdlet, 748–749
Set-ADServerSettings cmdlet, 59, 122, 364
Set-AuthenticodeSignature cmdlet, 118
Set-CalendarProcessing cmdlet, 202, 204, 260–261, 264
Set-CASMailbox cmdlet, 84
Set-Contact cmdlet, 98, 381
Set-DatabaseAvailabilityGroup cmdlet
  adding IP addresses, 493
  DAC and, 552
  editing properties, 492, 505–506
  exerting manual control over networks, 509
  lag copy self-maintenance, 531
Set-DatabaseAvailabilityGroupNetwork cmdlet, 509

Set-DistributionGroup cmdlet
group expansion, 277–278
mail-enabled groups, 285
mailbox delegation permissions, 232
management lists for groups, 276
marking existing groups, 298
moderated groups, 245, 248–249
multilingual custom MailTips, 381
protected groups, 278–279
users maintaining groups, 294
Set-DynamicDistributionGroup cmdlet, 246, 301, 306, 313
Set-EmailAddressPolicy cmdlet, 335, 343
Set-EventLogLevel cmdlet, 331
Set-ExchangeServer cmdlet, 71, 411
Set-ExecutionPolicy cmdlet, 118–119
Set-Group cmdlet, 98, 372–373
Set-InboxRule cmdlet, 320
Set-Mailbox cmdlet
about, 84, 192, 194
address book policies, 351–352, 355
applying retention policies, 678
assigning quota values, 216
bulk updates, 115
custom MailTips, 380
defining custom properties, 256
EAC and, 175
email address policies, 333
limiting user functionality, 91–92
mailbox delegation, 231–232
managing archive properties, 651
multilingual custom MailTips, 381
OAB generation mailboxes, 367
putting mailboxes on litigation hold, 699–701
putting mailboxes on retention hold, 698–699
RBAC and, 137
read tracking, 325
remote PowerShell and, 91
removing retention policies, 680
role assignment policies, 164–165
setting language, 196–197
shared mailboxes, 241
Set-MailboxAuditByPassAssociation cmdlet, 758
Set-MailboxAutoReplyConfiguration cmdlet, 202–203
Set-MailboxCalendarConfiguration cmdlet, 202–204, 206, 261
Set-MailboxCalendarSettings cmdlet, 261
Set-MailboxDatabase cmdlet
activation performance numbers, 535
backup settings, 416
controlling checksum scans, 433
handling move request errors, 609
maintenance settings, 434
OAB assignments, 370
provisioning settings, 211
reconnecting mailboxes, 220–221
renaming databases, 414
setting values, 620
Set-MailboxDatabaseCopy cmdlet, 159, 525, 527, 531
Set-MailboxExportRequest cmdlet, 625, 637
Set-MailboxFolderPermission cmdlet, 239
Set-MailboxImportRequest cmdlet, 625
Set-MailboxJunkEMailConfiguration cmdlet, 202, 204–205
Set-MailboxMessageConfiguration cmdlet, 202, 204
Set-MailboxRegionalConfiguration cmdlet, 199, 202–203, 206
Set-MailboxSearch cmdlet, 730–731
Set-MailboxServer cmdlet
activation blocks, 542–544
building DAGs, 561
database activation restrictions, 565
maximum number of databases, 402
OAB updates, 364
server switchovers, 538–539
work cycle information, 455
Set-MailboxSpellingConfiguration cmdlet, 202
Set-MailContact cmdlet, 251
Set-MailPublicFolder cmdlet, 381
Set-MailUser cmdlet, 253
Set-ManagementRoleAssignment cmdlet, 158
Set-MigrationBatch cmdlet, 594, 639
Set-MigrationEndpoint cmdlet, 614
Set-MoveRequest cmdlet, 608, 610–611
Set-Notification cmdlet, 639
Set-OfflineAddressBook cmdlet, 366, 370
Set-OrganizationConfig cmdlet
configuring MailTips, 377–378
creating site mailboxes, 802
group creation settings, 288, 292
hierarchical address books, 372
locking down public files, 790
read-status tracking setting, 324
Set-OrganizationConfig command, 790
Set-OutlookAnywhere cmdlet, 79
Set-PublicFolder cmdlet, 784, 786
Set-ReceiveConnector cmdlet, 92
Set-ResourceConfig cmdlet, 255–256
Set-RetentionPolicyTag cmdlet, 671, 681–682, 688
Set-RoleAssignmentPolicy cmdlet, 165
Set-RoleGroup cmdlet, 151, 158
Set-ServerComponentState cmdlet, 563–566
Set-SiteMailbox cmdlet, 803
Set-SiteMailboxProvisioningPolicy cmdlet, 804
Set-ThrottlingPolicy cmdlet, 725
Set-TransportConfig cmdlet, 531, 627
Set-User cmdlet
about, 98
address book policies and, 351–352
hierarchical address books, 373
remote PowerShell and, 88
updating mailbox properties, 194

Set-WebServicesVirtualDirectory cmdlet, 613
Set-WorkloadPolicy cmdlet, 580
Setup command
about, 49–51
/ActiveDirectorySplitPermissionsparameter, 167
/ADToolsNeeded switch, 52
delegated setup, 63
/IAcceptExchangeServerLicenseTerms parameter, 49
/PrepareAD parameter, 52, 167
/PrepareSchema parameter, 52
recovering failed servers, 62
running, 53–56
split permissions model, 167
uninstalling Exchange Server, 58
Windows PowerShell and, 84
setup logs, 56–57
shared mailboxes, 188, 240–241
SharePoint
integration with, 3, 18
site mailboxes and, 791–792
SQL support, 389
synchronization with Exchange and, 795–799
sharing policies, 313
Shift+Delete key combination, 738
Shift+Tab key combination, 97
–ShowDetails cmdlet parameter, 760
Simple Mail Transfer Protocol (SMTP)
certificate management, 313
email address policies, 343–344
message routing and, 35
selective output example, 100
site mailboxes and, 794
single copy cluster (SCC), 29
single forest design, 45
single-instance storage, 393
single-item recovery, 743–745
SinglePageRestore property, 521
site mailboxes
about, 18, 791–795, 804
accessing information from Outlook, 799–800
importing-exporting data in, 624
life cycle of, 800–803
Office 365 and, 800
opening maximum number of, 800
provisioning policy, 803–804
sizing, 803
sizing
databases, 389–392
mailboxes, 391–394
public folder mailboxes, 772–773
site mailboxes, 803
transaction logs, 420
–SkipActiveCopyChecks cmdlet parameter, 541
–SkipClientExperienceChecks cmdlet parameter, 540
–SkipHealthChecks cmdlet parameter, 541
–SkipLagChecks cmdlet parameter, 541
SLA (service level agreement), 9
SMB (server message block), 500
SMTP (Simple Mail Transfer Protocol)
certificate management, 313
email address policies, 343–344
message routing and, 35
selective output example, 100
site mailboxes and, 794
SoftDelete action, 756, 759
SoftDeleted state, 581, 604
Sort-Object cmdlet, 92–93
Source property, 713
–SourceMailboxes cmdlet parameter, 731
SourceMailboxes property, 713
–SourceRootFolder cmdlet parameter, 636
split brain syndrome, 551
split permissions model, 166–167
Split-PublicFolderMailbox.ps1 script, 773
SPs (service packs), 64
SQL Server, 388–389
SSL (Secure Sockets Layer), 30, 77, 313–314
SSP (security support provider), 506
StalledDueToHA status, 481
Standard edition, 32–33, 407
Standby Active Manager (SAM), 190
Standby Active Manager (SAM) role, 466
standby continuous replication (SCR), 29
Start-MailboxSearch cmdlet, 702, 731
Start-ManagedFolderAssistant cmdlet, 678, 690
Start-MigrationBatch cmdlet, 593
Start-Transcript cmdlet, 114
StartDagServerMaintenance.ps1 script, 563
State or Province property, 305, 346, 351
statistics, database usage, 451–453
–Status cmdlet parameter, 99, 481, 493, 634
Status property, 520, 713–714
–StatusDetail cmdlet parameter, 579
–StatusMailRecipients cmdlet parameter, 731, 762
StatusRetrievedTime property, 521
Stop-MailboxSearch cmdlet, 702
StopDagServerMaintenance.ps1 script, 563
storage area networks (SAN), 557
storage management
Active Directory, 35–36
DAG design, 557
eDiscovery searches, 225
handling storage failures, 549–551
setting mailbox quotas, 213–218
single-instance, 393
upgrading, 15–22
virtualized servers, 24–25
Store (Exchange Store). *See* Exchange Store
–StoreMailboxIdentity cmdlet parameter, 220, 223
streaming backups, 29, 389

stressed servers, 565–566
submission changes in messages, 489
Super Help Desk Users USG, 140
Support Diagnostics role, 160
Suspend-ClusterNode cmdlet, 564
–Suspend cmdlet parameter, 616, 618
Suspend-MailboxDatabaseCopy cmdlet, 529, 531, 545
Suspend-MailboxExportRequest cmdlet, 626
Suspend-MailboxImportRequest cmdlet, 625, 635
Suspend-MoveRequest cmdlet, 605
suspending mailbox moves, 615–619
–SuspendWhenReadyToComplete cmdlet parameter, 616–617
SuspendWhenReadyToComplete property, 606
swelling databases, 444–445
switchovers
  database copies, 466, 533–534
  server, 536–539
  targetless, 538
synchronization
  incremental, 575, 615
  public folder hierarchy, 772
  with SharePoint and Exchange Server, 795–799
  site mailboxes, 791–792
syntax for searches, 736–737
System Attendant process, 363
System Center Operations Manager (SCOM), 442–443, 470, 520
system registry
  CrashCount entry, 438
  execution policy, 119
  LastCrashTime entry, 438
  Patches entry, 69
  Products entry, 69
  ReplayLagManagerNumAvailableCopies entry, 531
  ReplayLagPlayDownPercentDiskFreeSpace entry, 531
  ReereadLogonQuotasInterval entry, 217, 235
  version numbers, 68
system resources, MRS and, 579–580
system tags, 662–663
SystemMailbox account, 76, 224, 363

T

Tab key, 95–97
TAP (Technology Adoption Program), 14, 68
TargetArchiveDatabase property, 593
TargetDatabase property, 593
targetless switchovers, 538
–TargetMailbox cmdlet parameter, 713, 731
–TargetRootFolder cmdlet parameter, 636
task items, retention policies and, 677
task logging, 502
Tasks folder, 658
TCP (Transmission Control Protocol), 7–8, 479
Team Mailboxes role, 148
Technology Adoption Program (TAP), 14, 68
templates
  display, 382–386
  OAB, 361
temporary log files, 430–431
Test-AssistantHealth cmdlet, 454–455
Test-MigrationServerAvailability cmdlet, 613
Test-Path cmdlet, 120
Test-ReplicationHealth cmdlet, 545
testing
  ACID tests, 419
  cumulative updates, 67
  customizations, 30–31
  operational processes, 29–30
  programming, 30–31
  test plans, 28–29
  virtual machines and, 25
Thawte certificate vendor, 314
threads, scheduling, 550
Threat Management Gateway (TMG), 16
throttle-based assistants, 454
throttling
  dynamic, 580
  resource throttling for searches, 724–726
tickles (signals), 576
time formats, 199
time to live (TTL) settings, 462
time zone property, 261
TimeInCPU metric, 453
TimeInServer metric, 452–453
timestamps, 425
TLS (Transport Layer Security), 313
TMG (Threat Management Gateway), 16
tombstone maintenance, 437
TotalMailboxSize property, 597
tracking group usage, 285–286
transaction log replay
  about, 479–483
  block mode replication, 484–486
  incremental resynchronization, 487–488
  seeding databases, 489
  transaction log compression, 483–484
  transaction log truncation, 486–487
transaction logs
  about, 388, 408, 419
  block mode replication and, 484–486
  checksums, 427
  circular logging, 429–431
  compressing, 483–484
  database redundancy and, 411
  debugging swelling databases, 445
  excessive growth, 443–444
  I/O operations, 427–429
  log sets, 419–427
  moving mailboxes, 578

replaying, 479–489
reserved logs, 431–432
sizing, 420
truncating, 486–487, 511
validating, 482
transcripts, 114
transient network conditions, 483
transitions, automatic database, 467–469
Transmission Control Protocol (TCP), 7–8, 479
Transport Dumpster, 459
Transport Layer Security (TLS), 313
Transport Queues role, 141
transport service, 27–28, 247–248
Troubleshoot-DatabaseLatency.ps1 script, 442
Troubleshoot-DatabaseSpace.ps1 script, 443
troubleshooting
adding scopes, 158
archive mailboxes, 656
dark zone and, 60
delivery reports, 323
deployment, 56
exceeded message quotas, 128
excessive database or log growth, 443–444
high latency, 442
mailbox creation, 212
mailbox moves, 599
migration batch logs, 592
retention tags, 671
$True variable, 101
truncation, transaction log, 486–487, 511
TruncationLagTime property, 528
TTL (time to live) settings, 462
–Type cmdlet parameter, 671

U

UAC (User Account Control), 54
UAnrdex.oab file, 360
UBrowse.oab file, 360
UCMA (Unified Communications Managed API), 53
UDetails.oab file, 360
UM Management role group, 75, 136
unattended installation, 54–55
UNC (Universal Naming Convention), 593
underscore (_), 26
Unicode format, 623
Unified Communications Managed API (UCMA), 53
Unified Messaging server role, 314
uninstalling Exchange Server, 58–60
Universal Naming Convention (UNC), 593
universal security groups (USGs), 73–77, 139–141, 267
Unrestricted execution policy, 118–119
Unscoped Role Management role, 160
unscoped roles, 160–162
–UnscopedTopLevel cmdlet parameter, 161
Unsearchable subfolder, 721
Update action, 756
Update-EmailAddressPolicy cmdlet, 340, 343, 345
Update-MailboxDatabaseCopy cmdlet, 516, 524, 542
Update-OfflineAddressBook cmdlet, 354, 364–365
Update-PublicFolderMailbox cmdlet, 772
Update-RoleGroupMember cmdlet, 150
updates
bulk, 114–116
cumulative, 14
incremental, 578
installation and, 27, 63–73
mailbox databases, 414–416
mailbox properties, 194, 652
OAB, 363–364
testing, 67
version, 31–32
waiting for, 13–15
UPdndex.oab file, 360
upgrading Exchange Server
building business cases, 16
common scenarios, 11–12
evolving from earlier versions, 12–13
fundamental questions before, 15–17
Information Store improvements, 19–20
integration with Lync, 3, 18
integration with SharePoint, 3, 18
management interfaces, 21–22
modern public folders, 20–21
preparations for, 26–27, 63–73
waiting for updates, 13–15
upgrading from managed folders, 688–689
UPN property, 185
UPN (User Principal Name), 185
URdndex.oab file, 360
UseDatabaseQuotaDefaults property, 215, 772
User Account Control (UAC), 54
user accounts
Active Directory, 49
administrators, 163
mail-enabled, 75–76
user-defined variables, 101–103
user mailboxes, 178, 187
user-maintained groups
about, 286–287
creating group naming policy, 287, 289–295
defining default group location, 288–289
planning, 287–288
restricting users, 295–297
room lists, 298–299
User Options role, 145–146
User Principal Name (UPN), 185
–UserPrincipalName cmdlet parameter, 192
Users OU, 75, 228, 288
USGs (universal security groups), 73–77, 139–141, 267
UTmplts.oab file, 360

V

validating
checksums, 427
query results, 305–306
transaction logs, 482
variables
common and user-defined, 101–103
email address policies, 339
–Verbose cmdlet parameter, 114
verbose PowerShell, 129
VeriSign certificate vendor, 118, 314
versions and version numbers
about, 67–69
DAG servers, 494–495
evolving from earlier, 12–13
groups and, 268
installation and, 63
object, 69–70
selecting for deployment, 22–23
updating, 31–32
Versions folder, 746
View-Only Organization Management role group, 75, 135, 323
View-Only Recipients role, 146
–ViewEntireForest cmdlet parameter, 122, 364
viewing group members, 268–269, 281–283
views (secondary indexes), 393, 395, 437, 578
VIPs (virtual IPs), 558
virtual IPs (VIPs), 558
virtual servers, 24–25
virtualization, 24–25
VMware, 24–25
voice mail, default tag, 661
Volume Shadow Copy Services (VSS)
backup considerations, 458
restoring databases, 390
streaming backups, 29, 389
VSS (Volume Shadow Copy Services)
backup considerations, 458
restoring databases, 390
streaming backups, 29, 389

W

watermarks, 55–56
Watson dumps, 551
WCF (Windows Communications Foundation), 568
WebDAV API, 16, 30
Where-Object cmdlet, 92–93, 107
white space in databases, 444, 449–450, 597
wildcard characters, 94, 110, 138, 737
Windows CA (Windows Certificate Services), 314–315
Windows Certificate Services (Windows CA), 314–315
Windows Communications Foundation (WCF), 568
Windows Failover Clustering, 463–464, 492, 495–496, 553
Windows PowerShell. *See also* cmdlets
about, 83–86
Active Directory for, 120–123
automation and, 170
command editing, 95–97
deployment overview, 50–53
development priorities, 4
execution policies, 118–119
hyphens and, 103
influence of cloud service, 6
installing, 44
Integrated Scripting Environment, 103–105
memory limits, 111
profiles, 119–120
quotation marks and, 102–103
remote, 86–92
ServerManager module, 52
ServerManagerCmd module, 52
starting sessions, 53
testing programming and customizations, 30
verbose output, 129
Windows Server Backup, 389
Windows Server Manager, 51
Windows Unified Communications Managed API, 44
Windows User Account Control (UAC), 54
witness servers, 490–491
WitnessDirectory property, 505
WitnessServer property, 505
worker processes
health mailboxes, 226–227
Managed Availability, 402–407, 469–472, 522, 536
WorkerProcessId property, 402, 521
working directory, 117
Workload Management, 579
Write-AdminAuditLog cmdlet, 750
write smoothing, 395
writing scripts, 93

X

X.400 standard, 343–344
X.509 certificates, 26, 313
%xg variable, 339
XML configuration files, 51–52, 205–206
%xs variable, 339
XSO API, 568

Z

zeroing, page, 433, 436

About the Author

Before retiring in March 2010, **Tony Redmond** spent 26 years at Digital Equipment Corporation, Compaq, and HP, ending up as Vice President and Chief Technology Officer for HP Services. Throughout that time, he maintained a close interest in enterprise email and worked on products such as ALL-IN-1, MailWorks, and TeamLinks at Digital and then formed a group to work on Exchange Server 4.0 in 1996. Since then, Tony has remained interested in Exchange and writes extensively about the technology, including its Office 365 cloud variant. This is his eleventh book covering Exchange.

Although Tony now runs his own consulting company, his range of other interests limits the time he spends doing real work. You can track his activities on his blog at *http://thoughtsofanidlemind.wordpress.com/* or read his more formal discussions about technology on his "Exchange Unwashed" blog at *http://windowsitpro.com/*. You can also follow Tony's Twitter feed by subscribing to @12Knocksinna.

Now that you've read the book...

Tell us what you think!

Was it useful?
Did it teach you what you wanted to learn?
Was there room for improvement?

Let us know at http://aka.ms/tellpress

Your feedback goes directly to the staff at Microsoft Press, and we read every one of your responses. Thanks in advance!

Microsoft