

Tony Redmond

Well-known author and Microsoft MVP for Exchange Server

Technical review by Paul Robichaux

Microsoft

Microsoft®
**Exchange
Server 2010**

**INSIDE
OUT**

- The ultimate, in-depth reference
- Hundreds of timesaving solutions
- Supremely organized, packed with expert advice

Includes
YOUR BOOK—ONLINE!

See back

Microsoft®

Microsoft® Exchange Server 2010 Inside Out

Tony Redmond

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

Copyright © 2010 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: 2010935972
ISBN: 978-0-7356-4061-0

3 4 5 6 7 8 9 10 11 QGT 7 6 5 4 3 2

Printed and bound in the United States of America.

Microsoft Press books are available through booksellers and distributors worldwide. For further information about international editions, contact your local Microsoft Corporation office or contact Microsoft Press International directly at fax (425) 936-7329. Visit our Web site at www.microsoft.com/mspress. Send comments to mspinput@microsoft.com.

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, e-mail addresses, logos, people, places, and events depicted herein are fictitious. No association with any real company, organization, product, domain name, e-mail 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: Martin DelRe

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.

Cover: Tom Draper Design

Body Part No. X17-21593



Contents at a Glance

Chapter 1	Chapter 15
Introducing Microsoft Exchange 2010 1	Compliance 973
Chapter 2	Chapter 16
Installing Microsoft Exchange 2010 39	Rules and Journals 1083
Chapter 3	Chapter 17
The Exchange Management Shell 75	The Exchange Toolbox 1129
Chapter 4	
Role-Based Access Control 147	
Chapter 5	
Exchange Management Console and Control Panel 181	
Chapter 6	
Managing Mail-Enabled Recipients 255	
Chapter 7	
The Exchange 2010 Store 357	
Chapter 8	
Exchange's Search for High Availability 425	
Chapter 9	
Backups and Restores 527	
Chapter 10	
Clients 555	
Chapter 11	
Client Access Server 651	
Chapter 12	
Mailbox Support Services 707	
Chapter 13	
The Exchange Transport System 801	
Chapter 14	
Message Hygiene 907	

Table of Contents

Foreword	xix
Introduction	xxi
Service Pack 1	xxii
Writing style and general approach to content	xxii
Examples used in the book	xxiii
Thanks	xxiv
In conclusion	xxvi
Support for this book	xxvi
We want to hear from you	xxvii
Your Companion eBook	xxvii
 Chapter 1: Introducing Microsoft Exchange 2010	1
The motivation to upgrade	3
Moving from Exchange 2003 or Exchange 2007	4
Testing and beta versions	6
Fundamental questions before you upgrade	7
No in-place upgrades	8
What version of Windows?	10
Preparing for Exchange 2010	11
The test plan	12
Testing for operational processes	14
Testing for programming and customizations	14
Bringing Exchange 2007 up to speed	16
Deploying earlier versions of Exchange servers alongside Exchange 2010	17
Web-based Deployment Assistant	18
Exchange 2010 editions	18
Active Directory	19
The strong link between Exchange and Active Directory	20
ADSIEdit	22

 **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:

www.microsoft.com/learning/booksurvey/

Types of Active Directory deployments that support Exchange	23
The role of ADAccess	25
Planning for global catalogs	29
Preparing Active Directory for Exchange	31
The joys of a customizable schema	34
Ready-to-go custom attributes	35
Let's install	37
Chapter 2: Installing Microsoft Exchange 2010	39
Approaching the installation	39
Running /PrepareAD	41
Installing prerequisite system components	42
Installing the Microsoft Filter Pack	46
Running Setup	46
Setup logs	49
Uninstalling Exchange	51
Repairing Exchange	53
Installing an edge server	54
Language packs	54
Recovering a failed server	55
Customer Experience Improvement Program	58
The services of Exchange	60
Versions, roll-up updates, and service packs	63
Exchange 2010 Service Pack 1	65
Version numbers	66
Object versions	68
Reporting licenses	69
Security groups and accounts created by Exchange	71
Contemplating management	74
Chapter 3: The Exchange Management Shell	75
How Exchange leverages Windows PowerShell	76
Remote PowerShell	79
Flowing remotely	81
Connecting to remote PowerShell	84
Be careful where you execute	86
A more complex environment to manage	86
Advantages of remote PowerShell	91
EMS basics	93
Command editing	96
Handling information returned by EMS	99
Selective output	100
Using common and user-defined variables	103
Identities	106
Piping	109
Adding recipient photos	111
OPATH filters	113

Server-side and client-side filters	114
Transcripts	117
Bulk updates	118
Code changes required by remote PowerShell	120
Command line versus Integrated Scripting Environment	122
Calling scripts	123
Profiles	124
Script initialization	125
Active Directory for PowerShell	126
Setting the right scope for objects in a multidomain forest	127
Some useful EMS snippets	129
Looking for large folders	129
Outputting a CSV file	130
Creating a report in HTML	131
Finding disconnected mailboxes	132
Creating and sending messages from the shell	132
Reporting database size and mailbox count via email	134
Verbose PowerShell	136
Setting language values	136
Execution policies	137
Testing cmdlets	139
Test-SystemHealth	139
Test-ServiceHealth	140
Test-MAPIConnectivity	141
Test-ReplicationHealth	141
Test-ExchangeSearch	142
Test-OWAConnectivity	143
Test-ECPCConnectivity	143
Test-MRSHealth	144
Testing POP3 and IMAP4 Connectivity	144
Testing mail flow	145
But we need some control	146
Chapter 4: Role-Based Access Control	147
RBAC basics	148
Roles	151
Using role assignment policy to limit access	152
Creating roles for specific tasks	154
Scopes	155
Role groups	156
Creating a new role group	159
Role assignment	160
Specific scopes for role groups	162
Special roles	164
Unscoped roles	165
What role groups do I belong to?	166
Assignment policies	168

RBAC enhancements in SP1	170
Managing role groups through ECP	170
Database scoping	174
Implementing a split permissions model	175
RBAC reports in ExBPA	178
RBAC validation rules	179
Exchange Control Panel and roles	179
Figuring out RBAC	179
On to management	180
Chapter 5: Exchange Management Console and Control Panel	181
Exchange Management Console	182
Changes to EMC in Exchange 2010	182
A different console philosophy from Exchange 2003	185
Managing objects across Exchange 2010 and Exchange 2007	187
EMC startup	188
How EMC accesses Exchange data	190
Changing EMC columns	194
Auto-generated PowerShell commands	195
Using EMS command logs	197
Naming conventions	199
Organizational health data	201
Managing multiple organizations	204
Sharing policies	205
Certificate management	208
Exchange Control Panel	213
SP1 updates for ECP	215
An overview of the ECP application	215
Basic ECP user options	216
Inbox rules	220
Delivery reports	224
ECP administrator options	227
Administrator searches for delivery reports	228
Running ECP without an Exchange mailbox	235
Managing groups with ECP	237
Defining a default group location and group naming policy	238
Creating new groups	242
Creating security groups with ECP	243
Users and groups	244
Allowing users to create new groups through ECP	247
Planning for user-created groups	248
Maintain groups but don't create!	249
Setting diagnostics for Exchange servers	251
But what will we manage?	253
Chapter 6: Managing Mail-Enabled Recipients	255
Stop and think	255
Mailbox naming conventions	257

Creating new mailboxes	259
Completing the new mailbox setup	264
Creating new room and resource mailboxes	265
Mailbox provisioning agent and database allocation	265
Languages and folders	269
Manipulating mailbox settings	273
Bulk mailbox creation	277
Setting quotas	279
What's in a mailbox?	284
Removing or disabling mailboxes	285
Reconnecting mailboxes	286
Email address policies	290
Email policy priority	292
Creating a new email address policy	293
Creating email address policies with custom filters	297
Setting priority for an email address policy	297
Virtual list view (VLV) for Exchange address lists	299
Discovery mailboxes	299
Creating additional discovery mailboxes	301
Setting mailbox permissions	303
Mail flow settings	303
The difference between Send on Behalf and Send As	304
Managing full access permission	306
Sending messages on behalf of other users	309
Opening another user's mailbox	310
Distribution groups	312
Room lists	314
Group owners	316
Group expansion	318
Protected groups	319
Self-maintaining groups	321
Viewing group members	322
Tracking group usage	324
Dynamic distribution groups	324
OPATH queries	325
Creating new dynamic distribution groups	326
Creating dynamic groups using custom filters	329
Moderated recipients	334
Moderation requests	337
Moderated mailboxes	340
Mail-enabled contacts	341
Mail users	342
Resource mailboxes	343
Defining custom properties for resource mailboxes	345
Providing policy direction to the Resource Booking Attendant	347
Processing meeting requests according to policy	352
Equipment mailboxes	355
Data, data, everywhere	355

Chapter 7: The Exchange 2010 Store	357
Long live Jet!	358
Maximum database size	359
Database limits for the standard edition	361
Mailboxes per database (or per server)	362
Dealing with I/O	364
Maintaining contiguity	370
A new database schema	372
Database management	374
Creating new mailbox databases	377
Updating mailbox databases after installation	381
Background maintenance	383
Scheduling background maintenance	387
Content maintenance tasks	388
Tracking background maintenance	390
Corrupt item detection and isolation	391
Backups and permanent removal	394
Protection against high latency	395
Protection against excessive database or log growth	396
Store driver fault isolation	397
The death of ISINTEG	398
Controlling named properties	401
Database defragmentation	404
Using ESEUTIL	406
Database usage statistics	407
Transaction logs	409
Log sets	410
Transactions, buffers, and commitment	413
Transaction log checksum	417
Transaction log I/O	418
The question of circular logging	419
Noncircular logging	421
Reserved logs	422
And now for something completely different	423
Chapter 8: Exchange's Search for High Availability	425
Breaking the link between database and server	426
Introducing Database Availability Groups	428
The dependency on Windows clustering	431
Active Manager	433
Automatic database transitions	435
Best copy selection	437
ACLL: Attempt copy last logs	439
Transaction log replay: The foundation for DAG replication	440
Transaction log compression	445
Block replication	446
Transaction log truncation	448

Incremental resynchronization	449
Seeding a database	451
Unique database names	451
Changes in message submission within a DAG	455
Day-to-day DAG management and operations	455
Building the DAG	462
Investigating DAG problems	468
Managing DAG properties	469
DAG networks	471
Using circular logging with database copies	475
Adding new database copies to a DAG	477
Handling initial seeding errors	479
Monitoring database copies	480
Reseeding a database copy	481
Adding database copies with EMS	482
Using a lagged database	484
Activating a mailbox database copy	488
Applying updates to DAG servers	492
Dealing with a failed server	493
AutoDatabaseMountDial and potential issues moving databases	495
Activation blocks	499
Moving database locations within a DAG	500
Removing database copies	502
Removing servers from a DAG	506
Handling storage hangs	507
Upgrading servers in a DAG	508
Datacenter Activation Coordination	510
Planning for datacenter resilience	511
Managing cross-site connections	513
Crimson events	514
Approaching DAG designs	515
Scripts to help with DAG management	520
On to protecting data	525
Chapter 9: Backups and Restores	527
An interesting philosophical question	527
The Windows Server Backup plug-in for Exchange	530
Exchange and Volume ShadowCopy Services	531
Making an Exchange 2010 backup	533
The backup complexities posed by passive database copies	537
Restoring to a recovery database	538
Performing a restore	540
Validating the recovered database	543
Mounting a recovery database	544
Restoring mailbox data	547
Complete server backups	552
Clients	553

Chapter 10: Clients	555
The Outlook question	557
Missing functionality when using earlier versions of Outlook	559
Why new mail notifications seem slower on Outlook	561
Forcing faster Outlook Anywhere connections	562
Conversation views	563
Conflict resolution	567
Listing client connections	569
Blocking client connections to a mailbox	570
Blocking client access to a mailbox server	573
Outlook Web App	574
A refresh for OWA provided by Exchange 2010 SP1	575
OWA functionality deprecated in Exchange 2010	578
Different browsers, different experiences	579
OWA configuration file	583
Missing favorites	584
Forwarding meeting requests	585
OWA Web parts	586
Long signatures	587
Sharing calendars	588
Sharing calendars with Internet users	590
Mailbox quota exceeded	594
Handling attachments	595
OWA themes and customizations	597
OWA mailbox policies and feature segmentation	600
More than just segmentation	604
Attachment processing	608
Applying an OWA mailbox policy	609
POP3 and IMAP4 clients	610
Configuring the IMAP4 server	612
Configuring IMAP4 client access	615
Exchange ActiveSync	618
Setting ActiveSync policies	620
Generating ActiveSync reports	622
Reporting synchronized devices	623
Blocking types of mobile devices	626
Blocking devices on a per-user basis	631
Wiping lost devices	632
Debugging ActiveSync	635
Testing mobile connectivity	636
ActiveSync for BlackBerry	636
Client throttling	637
Unified Messaging	641
Voice mail preview	642
Fax integration	647
Exchange 2010 APIs	647
Exchange Web Services	648
A common connection point	650

Chapter 11: Client Access Server	651
The CAS role	652
Benefits of relocating the MAPI endpoint	653
CAS installation priority	655
The RPC Client Access layer	657
Linking CAS to mailbox databases	659
Supporting Outlook 2003 clients	661
CAS access to directory information	662
The Autodiscover service	663
Accessing a Service Connection Point	663
CAS settings	666
Site scope	668
AutoConfiguration	668
Logging Autodiscover actions	670
Static Autodiscover	673
SRV pointers to Autodiscover	675
Client Access Server arrays	676
Creating a CAS array	678
Managing cross-site connections with the RPC Client Access service	679
Load balancing and CAS arrays	681
Upgrading a Client Access Server in an array	682
CAS and perimeter networks	684
RPC Client Access logging	685
Certificates	688
Outlook Anywhere	691
An increased load for the CAS	692
Load balancing the CAS	693
The importance of affinity	696
Assigning static ports to the CAS	698
Web services URLs and load balancing	701
Changes to facilitate SSL offloading	702
Domain controllers	702
Preparing for transition and interoperability	703
A matter of manipulation	705
Chapter 12: Mailbox Support Services	707
The Mailbox Replication Service	707
MRS configuration file	708
Moving mailboxes	709
Asynchronous moving	711
Mailbox Replication Service processing	713
Preventing loss of data	716
Moving mailboxes	717
Clearing move requests	722
Managing mailbox moves with EMS	723
Preserving the mailbox signature	726
Moving mailboxes between versions of Exchange	727
Moving mailboxes with personal archives	729

Checking move request status	731
Planning mailbox moves	732
Ensuring high availability	736
Reporting mailbox moves	738
Accessing move report log data	740
Moves and mailbox provisioning	743
Handling move request errors	744
Mailbox import and export	747
Gaining permission through RBAC to execute mailbox import and export	749
Planning the import of PST data	750
Exporting mailbox data	758
Limiting user access to PSTs	760
MailTips and group metrics	762
Configuring MailTips	766
User experience	768
Custom MailTips	770
Multilingual custom MailTips	771
The Offline Address Book	772
OAB download	773
OAB generation	776
Updating OAB files	781
Moving the OAB generation server	782
Web-based distribution	783
Creating and using customized OABs	785
OAB support for MailTips	790
OABInteg and Dave Goldman's Blog	791
Hierarchical address book	791
Mailbox assistants	793
Calendar Repair Assistant (CRA)	794
Work cycles	797
Time to transport	799
Chapter 13: The Exchange Transport System	801
Overview of the transport architecture	802
Active Directory and routing	806
Overriding Active Directory site link costs	808
Delayed fan-out	810
The critical role of hub transport servers	811
Version-based routing	813
Transport configuration settings	816
Limits on user mailboxes	822
Transport configuration file	823
Caching the results of group expansion	825
Routing tables	826
TLS security	830
Receive connectors	831
Creating a receive connector	835

Send connectors	841
Creating a send connector	845
Selecting a send connector	851
Linked connectors	853
Throttling	854
Back pressure	857
Transport queues	859
How messages enter the submission queue	861
Moving messages to delivery queues	861
Viewing queues	862
Problem queues	865
Exchange Queue Viewer	867
Submitting messages through the pickup directory	869
Replay directory	871
Customizable system messages	871
Exchange DSNs	871
Customizing NDRs	875
Customizing quota messages	878
Logging	880
Controlling connectivity logging	881
Interpreting a connectivity log	883
Protocol logging	884
Accepted domains	886
Creating a new accepted domain	888
Updating accepted domains	889
Remote domains	889
Transport pipeline	891
Foreign and delivery connectors	893
Shadow redundancy	894
Linking Exchange 2003 to Exchange 2010	898
Decommissioning Exchange 2003 routing groups	900
Handling Exchange 2003 link state updates	900
Changes in Exchange 2010 SP1	901
Better SMTP load balancing	902
Monitoring the submission queue	903
Mailbox delivery prioritization	904
Upgraded shadow redundancy	906
Squeaky-clean email	906
Chapter 14: Message Hygiene	907
To Edge or not to Edge, that's the question	908
Edge servers	909
Edge synchronization	911
Validating Edge synchronization	915
Ongoing synchronization	919
Exchange anti-spam agents	923
Installing the anti-spam agents on a hub transport server	924

Order of anti-spam agent processing	925
X-headers added by anti-spam agents	926
Header firewalls	929
Connection filtering	931
Sender filtering	934
Backscattering	935
Sender reputation	936
Recipient filtering	939
Tarpits	940
Sender ID	940
Content filtering	946
Attachment filtering	953
Address rewriting	955
Agent logs	957
Safelist aggregation	961
Choosing an antivirus product	964
Client defense	965
Outlook's junk mail filter	966
Cleansed email, but compliant?	972
Chapter 15: Compliance	973
The joy of legal discovery	974
Personal archives	976
Enabling a personal archive	979
Default archive policy	985
Disabling a personal archive	987
Using a personal archive	987
Messaging records management	989
The new approach to messaging records management in Exchange 2010	990
System tags	994
Designing a retention policy	995
Naming retention tags	997
Creating retention tags	998
Creating a retention policy	1004
Applying a retention policy to mailboxes	1007
Modifying a retention policy	1009
Customizing retention policies for specific mailboxes	1010
User interaction with retention policies	1012
Removing a retention policy	1017
Upgrading from managed folders	1018
How the Managed Folder Assistant implements retention policies	1018
Putting a mailbox on retention hold	1021
Putting a mailbox on litigation hold	1022
The very valuable dumpster	1025
Dumpster basics	1025
Dumpster 2.0 arrives	1027
Single item recovery	1029

Knowing what's in the dumpster	1031
Managing dumpster parameters	1032
Discovery searches	1033
Unsearchable items	1035
Creating and executing a multimailbox search	1037
Accessing search results	1040
Deduplication of search results	1043
Search logging	1045
Search annotation	1046
Executing searches with EMS	1047
Auditing administrator actions	1049
The audit mailbox	1052
How administrator auditing happens	1052
Auditing mailbox access	1057
Enabling mailboxes for auditing	1059
Accessing mailbox audit data	1061
Message classifications	1064
Creating a message classification	1065
Localized message classifications	1067
Client access to message classifications	1067
Protecting content	1070
Active Directory Rights Management Services	1072
Installing Active Directory Rights Management	1073
Using AD RMS to protect content	1076
Rights management enhancements in Exchange 2010 SP1	1080
Outlook Protection Rules	1080
Rules help compliance, too	1082
Chapter 16: Rules and Journals	1083
Transport rules	1083
Examples of transport rules	1085
Rules and ECP	1087
Basic structure of transport rules	1088
Edge versus hub rules	1088
Setting transport rule priority	1089
Creating a corporate disclaimer	1091
Basic moderated workflow	1097
Evaluating Active Directory attributes in transport rules	1099
Ethical firewalls	1101
Blocking certain users from sending external email	1102
Scanning attachments with transport rules	1105
Using message classifications and rights management templates in transport rules	1108
Caching transport rules	1110
Transferring rules between Exchange versions	1111
Transport rule actions	1112
Developing custom transport agents	1113

Transport rule priority	1114
Journaling	1114
When journaling happens	1115
Journaling options	1116
Journal reports	1116
Alternate journal recipient	1120
Standard journaling	1121
Journal rules	1122
Creating a journal rule	1123
Assessing journal load	1125
Securing a mailbox used as a journal recipient	1126
Intervention and interorganization journaling	1127
To the toolbox	1127
Chapter 17: The Exchange Toolbox	1129
Display or Details Templates Editor	1130
Message tracking	1135
Message tracking log files generated on servers	1139
Interpreting entries in message tracking logs	1142
Measuring message latency	1151
Using the Tracking Log Explorer	1153
Other options for analyzing messaging tracking logs	1158
Performance Monitor	1159
Exchange Performance Troubleshooter	1162
ExPerfWiz	1162
ExPerfWiz limitations	1164
Exchange Load Generator 2010	1165
Remote Connectivity Analyzer	1167
Searching for more information	1170
 Index to Troubleshooting Topics	1171
 Index	1173



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:

www.microsoft.com/learning/booksurvey/

Foreword

I took over the Exchange team in 2008 after 18 years in various roles at Microsoft, and was welcomed to the team appropriately via a post on the “You Had Me at EHLO” Exchange team blog. In November 2009, not too long after taking on this new mantle, I was in Las Vegas preparing to keynote the Exchange Connections conference to celebrate the launch of Exchange Server 2010. Knowing that I’d have some time to get to know members of the much-lauded Exchange community at the event, I reached out to my team for suggestions on which customers to seek out, which partner booths to visit, and any other advice they had. If one thing was universally clear it was that I had to—simply must—meet Tony Redmond.

Tony has been a fixture in the Exchange community for over a decade. Whether it is his advocacy for the Exchange customers or his critical feedback toward improving the product, Tony has played a significant role in the Exchange ecosystem since before the first Exchange Server ever shipped out of Redmond. He is one of the most popular speakers and authors on Exchange and an important voice for every one of the executives who preceded me as leader of Exchange at Microsoft.

It is appropriate that, after 14 years, Tony is publishing a book on Exchange Server 2010 SP1, a release that has so much to do with responding to customer and community feedback through early adopter and beta programs in which he has been so active over the years. Although an important milestone for the server, Exchange 2010 SP1 is also a significant milestone in our cloud strategy. This is the version of Exchange that we carry to our own datacenter as we bring the capabilities of Exchange 2010 to the cloud. It is unprecedented in the industry to provide a product that so comprehensively increases the operational efficiency of IT, makes users more productive in their daily workflow, reduces the risk profile of an organization, and brings this value to customers whether they choose to deploy servers on premises or migrate mailboxes into the cloud.

It is this unprecedented choice and flexibility that makes Exchange so unique and SP1 so important. It is with this focus that my team marches forward as we plan for the next updates to Exchange Online and the next versions of Exchange Server. Exchange 2010 SP1 makes me excited for the future of Exchange and I hope it does the same for you.

On behalf of my entire team, thank you for being part of our Exchange family and enjoy the book!

Rajesh Jha
Corporate Vice-President, Exchange
Microsoft Corporation

Introduction

Welcome to my tenth book covering the technology in Microsoft Exchange Server and its surrounding ecosystem. I seem to have been constantly writing about Exchange since before its introduction as version 4.0 in March 1996 in books and many articles printed in the redoubtable *Windows IT Pro* magazine (<http://www.windowsitpro.com>). All my previous books were published under the Digital Press imprint, which has now disappeared as a result of corporate upheavals. This is my first book working with Microsoft Press and it's been an interesting and productive experience for me to work with the publishing arm of the company that engineers Exchange. I look forward to future collaboration.

No book can cover every aspect of a huge product such as Exchange. To attempt to do so would require a multivolume set spanning many thousands of pages and create something that would probably be too expensive for most administrators to buy. This book covers the topics in Exchange that are most interesting to me and those that I think are most useful to the majority of administrators who need to understand how to manage an Exchange organization. There are some notable omissions, such as Unified Messaging and Exchange's connection to other Microsoft products such as Office Communications Server, which you might find surprising. However, the truth is that there are other books available that do a good job of covering these topics, so I feel able to concentrate on the areas that I think deserve the most investigation (or are most interesting to me). There's also an incredible amount of information posted in blogs and other commentaries available on the Web, so if your interest is piqued by a topic and you want to find more information, plug the topic into your search engine of choice and you're likely to find additional insights and observations. Apart from anything else, you'll discover information that is up to date and reflects advances due to software updates (I predict that Microsoft will continue to upgrade Exchange 2010 after Service Pack 1!) and the knowledge that accumulates over time about any product that's used in production environments.

Exchange 2010 has been an interesting journey because it provoked more new thoughts and ideas for me than any other version released by Microsoft. Although Exchange 2007 laid down much of the architecture that Exchange 2010 exploits, there is a mass of detail in the changes between the two versions. Two of the three big changes effected in Exchange 2007—Windows PowerShell, the transition to a pure SMTP-based transport system, and the introduction of transaction log shipping as the basis for database replication—have been expanded and enhanced in a very effective manner. Remote Windows PowerShell and the Database Availability Group might be what people remember as the big new things that appeared in Exchange 2010, but their foundation was laid many years previously and what we see today is simply the result of a lot of development and hard work since Microsoft finished the development of Exchange 2003. Maybe that's why there is so much to discuss and comment on.

Service Pack 1

I decided to base the book on Exchange 2010 Service Pack 1 (SP1), which Microsoft released to customers in August 2010, because I felt that there were a number of important areas that were incomplete in the original (RTM) version of Exchange 2010 released in October 2009. The fact that Microsoft needed additional time and effort to fully build out the features in Exchange 2010 should come as no surprise to anyone who has any experience with Exchange.

Don't get me wrong: The software that Microsoft shipped to customers in late 2009 was highly functional and had many strong points. However, the nature of software development is that a little extra time always helps to smooth rough edges and deliver the complete plan that the engineers wanted to build when they started to work on Exchange 2010. SP1 had the benefit of nine months' more development, testing, and documentation, plus the invaluable exposure that software receives when customers deploy it into production environments. The experience gained from this time, the feedback from customers and the Microsoft Most Valuable Professional (MVP) community, the insight shared in blogs and forums, and the bugs that were uncovered and fixed made SP1 a better target about which to write. Besides, I hate books that are rushed to market to meet an arbitrary date so that they can be first to market to cover new technology, because I know the dangers involved in writing about beta versions of technology. You can hope that the developers won't change the software between the beta and final version, but all too often a diversion appears between the description on the printed page and what the software actually does. It's safer to wait and see what the experience is with the software after it is released rather than rush to market to sell a few more books.

Many major and minor differences exist between the original version of Exchange 2010 and SP1 and I have attempted to indicate clearly where these differences exist.

Writing style and general approach to content

My writing approach to books is best described as chaotic and not very organized. I don't start with a list of topics and work through from A to Z until the book is done. I think I would find such an approach tiresome and would lose interest halfway through. Instead, I build the book from page to page and split content into chapters as the form of the book develops. Text is added as I discover new and interesting facts about the technology. I make no apologies for this approach, because it has served me well in my previous 12 books. However, I acknowledge that topics that don't interest me are omitted or receive short shrift. Ah well, you can't be brilliant at everything and you have to leave room for others to prosper.

Along these lines, I made some deliberate decisions about topics that I would not cover in this book to concentrate on what I believe are the most important technical and functional

changes in Exchange 2010. In effect, I used the 80–20 rule to select topics that I believe are of interest to the vast majority of the Exchange community and omitted others. So, to take two examples, if you are interested in the finer technical points of Unified Messaging or Active Directory Rights Management Services, you'll probably want to buy a different book. I think that these topics deserve specialized in-depth attention that cannot be justified in a book that attempts to cover the fundamental principles of Exchange. The same is true about connecting Exchange organizations with federation or integrating Exchange with various other products such as Microsoft Office Communications Server. These tasks can be done and are often done, and the subjects are explained better elsewhere. Be warned, therefore, that you might be disappointed if your favorite topic is not covered. On the other hand, you might be delighted that a topic of special importance to you is covered across many pages because we share a joint interest in it. In passing, I note that three chapters were removed from the original draft of the book to get the book down to a reasonable size. These chapters covered public folders, hardware planning, and cloud-based Exchange.

I doubt that many will read the book from beginning to end. At least, I have not written the text to flow from one chapter to another in the way that a novel or a history does. I expect most people to plunge into a part that interests them and then discover new topics as the need arises or curiosity takes over. I like technical books where chapters are self-sufficient and I hope that this book meets that goal.

In terms of other sources of technical information about Exchange 2010, I recommend that you download the latest version of the compiled help file (CHM) and keep it close at hand, because it will provide you with an invaluable guide to Exchange 2010 that you can use offline. TechNet provides an online copy, but Microsoft is quite good at updating the downloadable CHM regularly and did an excellent job for SP1. They've also gotten a lot better in terms of the breadth and depth of the content, even if it sometimes leaves gaps in the explanation. Of course, those gaps are exploited by the mass of Exchange 2010 books, magazine articles, and blogs. I particularly like the blogs of my fellow Exchange MVPs, even if it took me a long time before I got around to writing a blog myself. I now blog regularly at <http://thoughtsofanidleminde.wordpress.com/> and hope that I add some value to the Exchange community there.

Examples used in the book

I've tried to include as many examples as I can to illustrate points and show you exactly what you can expect to see when you execute a command. There are three kinds of examples:

- A simple screen shot. Hopefully these are self-evident. I've tried hard to avoid liberally scattering screen shots across the pages of the book because I hate big, thick, technical books that are half composed of screen shots. However, given the graphic nature of Windows and the Exchange management GUI, some screen shots are inevitable! In

most cases I have opted to use Outlook Web App to illustrate client functionality simply because you can be guaranteed that Outlook Web App is available within every Exchange 2010 deployment. In addition, huge variation exists in the feature set available in the Outlook versions supported by Exchange 2010; Outlook 2010 is therefore only used to illustrate unique features that are only exposed in this version.

- Illustrative Windows PowerShell (Exchange Management Shell) commands. If only because it is the foundation of Exchange 2010, there are many EMS examples throughout the book. If you don't know how to use EMS to manage Exchange, you miss out on so much of the potential that the product offers. All references to cmdlets in the body of the text plus example commands are shown like this:

```
Get-User -Identity Redmond
```

- Windows PowerShell commands and example output. In some cases I want to show you exactly what you will see when a command is executed. Windows PowerShell output can be pretty ugly and reading it from a screen shot is not always easy, so I show the Windows PowerShell command before and then the output in a separate block immediately following the command:

```
Get-StoreUsageStatistics -Database 'VIP Data'
```

DigestCategory	SampleId	DisplayName	TimeInServer
-----	-----	-----	-----
TimeInServer	0	Mailbox -Redmond, Tony	485

I'm certainly not an elegant or skilled Windows PowerShell coder. Rather, I like to think of myself as a contented hacker who fully buys into the concept that the charm of Windows PowerShell is that you can quickly stitch together snippets of code to do remarkable things. I apologize if I offend any purists with the examples presented in the book. I just do my best to make things happen with the best code I can.

The examples are based on an Exchange organization called contoso.com that runs on my notebook. It's really amazing how notebook technology has evolved to a point where a portable computer can happily support several servers while still allowing you to run client applications such as Word and Outlook that are available through a simple Alt+Tab keystroke.

Thanks

David Espinoza, Senior Product Manager in the Exchange "ship team," has been a delight to work with for many years. The ship team does what its title indicates: It is responsible for the complex choreography involved in shepherding a product from development to a point where it can be used by customers. David's team runs the Technology Adaption Program

(TAP), which puts beta versions of Exchange into customer hands early so that code can be exercised in real-life environments. The ship team organizes regular calls to inform people about new features and development progress, makes new builds available on a regular basis, and acts as the interface for bugs and feature requests that flow back from customers into Microsoft. All in all, the team does a standout job and David and his faithful assistants Robbie Roberts and Kern Hardman deserve my full thanks.

I've also received great advice and information from many individual contributors in the Exchange development group, including Dmitri Gavrillov, Jim Edelen, Kumar Venkateswar, Crystal Flores, William Rall, Julian Zbogor-Smith, Derek Tan, Kamal Janardhan, David Los, Sanjay Ramaswamy, Frank Byrum, Robin Thomas, Yesim Koman, Todd Luttinen, Linden Goffar, Ed Banti, Jim Knibb, Mayerber Carvalho Neto, Greg Taylor, Paul Bowden, and Siddhartha Mathur. I also acknowledge the help that I received from Bill Long to make the ExFolders utility work with beta builds of SP1.

Martin DelRe first contacted me in August 2008 to explore the possibilities of writing a book for Microsoft Press. Exchange 14, which is what Exchange 2010 was known as then, seemed like a good target and there was certainly plenty of new stuff to write about, but I was busy and didn't become enthused about the project until June 2009. Martin guided me through the process of writing for a new publisher (they are all different) and made sure that I didn't have to cope with too much bureaucracy, for which I am very thankful. Karen Szall directed the production of the book from submission through copyediting, technical review, and eventual publication, and did a wonderful job of making everything come together on time, including the ability to manage a constant flow of changes from me as I attempted to update the technical content of the book to match new experiences, insights, and reports of problems and workarounds discovered in the field.

A group of willing volunteers within HP who have enormous experience with enterprise messaging systems reviewed the material as it was under development. I'd like to thank Marc Van Hooste, Kevin Laahs, Andreas Zit, and Mike Ireland for their advice.

HP has one of the largest deployments of Exchange, with more than 350,000 mailboxes supported by a highly centralized datacenter structure centered in the United States. Kathy Pollert, Mike Ireland, and Stan Foster have contributed greatly to my understanding of how Exchange functions in very large environments and I truly appreciate the insight that they have shared with me over the last decade. I'd also like to thank Aric Bernard and Guido Grillenmeier for allowing me to deploy new software into the sandbox environment used by HP consultants on a regular basis. Aric and Guido are Active Directory gurus and would probably prefer that Exchange was kept well away from their nice, clean domain controllers. Into every life a little rain must fall and every Active Directory administrator has to learn that the directory is there to serve applications rather than to explore the wonders of replication. We had fun getting Exchange 2010 to even install into HPQBOX, mostly because a previous deployment of Exchange 2007 had been removed through brutal surgery applied

to the directory, leaving many lingering objects that just got in the way. Guido cleaned everything up and Aric made the servers run nicely, and I appreciate their efforts.

Finally, the dedicated effort of Paul Robichaux as technical editor must be acknowledged. Paul read every page, challenged places where I didn't seem to make sense or had misstated something, and ferreted out errors, all to improve the overall quality and content of the book. I owe him a lot.

In conclusion

I hope that you enjoy the book and its contents prove valuable in your understanding of Exchange 2010 as you approach the assessment, design, deployment, operations, and management of the software in production. At the end of the day, Exchange is only software and it's only as good as the people who work with it. To do a good job, you need knowledge about the product and wisdom to know when that knowledge runs out and it's time to look elsewhere for help, whether that's seeking out a Web site or asking someone for an opinion. The key is to realize that best practice is never stagnant and evolves all the time. Assuming that knowledge gained two or three years ago represents the current best advice and guidance is a fool's position. Always be prepared to learn.

Tony Redmond
September 2010

The author can be contacted through his blog at
<http://thoughtsofanidlemind.wordpress.com/>.

Support for this book

Every effort has been made to ensure the accuracy of this book. If you do find an error, please report it on our Microsoft Press site

1. Go to *www.microsoftpressstore.com*.
2. In the Search box, type the ISBN for the book, and click Search.
3. Select the book from the search results, which will take you to the book's catalog page.
4. On the book's catalog page, find the Errata & Updates tab

If you have questions regarding the book or the companion content that are not answered by visiting the book's catalog page, please send them to Microsoft Press by sending an email message to msspinput@microsoft.com.

We want to hear from you

We welcome your feedback about this book. Please share your comments and ideas via the following short survey:

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

Your participation will help Microsoft Press create books that better meet your needs and your standards.

Note

We hope that you will give us detailed feedback via our survey. If you have questions about our publishing program, upcoming titles, or Microsoft Press in general, we encourage you to interact with us via Twitter at <http://twitter.com/MicrosoftPress>. For support issues, use only the email address shown above.

Your Companion eBook

The eBook edition of this book allows you to:

- Search the full text
- Print
- Copy and paste

To download your eBook, please see the instruction page at the back of this book.



CHAPTER 1

Introducing Microsoft Exchange 2010

The motivation to upgrade	3	Exchange 2010 editions.....	18
What version of Windows?.....	10	Active Directory	19
Preparing for Exchange 2010	11	Let's install.....	37

FOR more than 30 years, I have worked with email software. The advent of a new version of a popular software product such as Microsoft Exchange Server 2010 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 seeks to explain the impact of the release of Exchange 2010 and the subsequent Service Pack 1 update 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 presented as you go through the different topics presented in this book. We should begin, though, by presenting the case for Exchange 2010 and exploring just what Microsoft wanted to accomplish in this release of the product.

Microsoft hopes that the quality of Exchange 2010 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 their competitive situation, something that is especially important in the new era of cloud-based services where Google has taken a lot of mindshare and IBM shows new signs of life with an online version of Domino. 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 the 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 and assess, then decide how best to use the new software. All of these things are true for Exchange 2010, which sits at the center of a large ecosystem spanning well over 100 million deployed mailboxes that has been growing since 1996.

It's tough to drive innovation into a product that has been around for so long, and it's tough to satisfy all of 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, they have to include enough "new stuff" in the product to create a compelling case for an upgrade. The evolution to the cloud makes this release even more interesting because Microsoft now has to build a product that works equally well for on-premise and hosted deployments. Companies have offered hosted Exchange services for years, so that's not what is different here. The critical changes are the nature of competition and Microsoft's decision to enter the hosted services market in a much more emphatic way than they have in the past.

When briefing customers about the development priorities for Exchange 2010, Microsoft stresses three areas:

1. Increase operational flexibility through easier deployment, higher availability, and simpler administration. Much of the development work occurs in the Store and Client Access Server (CAS) to increase availability through data replication and break the connection between databases and servers. A new ability to delegate administrative tasks through the Exchange Control Panel coupled with the expansion of Windows PowerShell to support remote tasks allows customers to automate common processes.
2. Streamline communications by supporting larger, better-organized mailboxes; investing more into unified communications; and allowing users to work more easily together no matter what device or client they use. The focus here is to support 10 GB mailboxes with the same performance that Microsoft Exchange Server 2007 uses to support 1 GB mailboxes. The user experience is further enhanced with new functionality in Microsoft Outlook 2010, Outlook Web App, and mobile clients. In passing, it's worth noting the name change for OWA (which I will use throughout the book, if only to stop calling the application by its old name). OWA was originally named Outlook Web Access in Microsoft Exchange 5.0. This name reflected the provision of access to a mailbox from a browser (initially only Microsoft Internet Explorer was supported) rather than the full-fledged application into which OWA gradually evolved. Microsoft now regards OWA as a client that delivers functionality comparable with Outlook in most respects. The name change to Outlook Web App reflects this stance and also aligns the name with other Web-based versions of Microsoft Office applications.
3. Deliver greater visibility and control with protected communications, built-in compliance and archiving functionality, and better reporting and management alerts. Exchange has supported message journaling since Microsoft Exchange Server 2003 and Microsoft Exchange Server 2007 introduced features such as managed folders

and transport rules. Exchange 2010 takes a much more comprehensive approach to the problem posed by legal requirements to meet regulations, comply with discovery actions, and handle situations such as sexual harassment.

Microsoft makes a fair point that these areas of investment have to work as well for hosted environments as they do when deployed onsite. Security is obviously a big challenge for hosted environments, as all communications have to be routed from a customer's own network across the Internet to a datacenter hosted by Microsoft or another provider. 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 as well as the multitenant shared environments that are becoming more common because of their cost efficiencies.

At the time of writing, Exchange has been under development for more than 16 years, and its source code encompasses some 21 million lines of code. No engineering group stays constant over such an extended period. Different engineering managers, internal Microsoft politics, and competitive pressure have all contributed to different priorities for the product over the years. The initial thrust in 1996 through 1998 to provide a migration path for Microsoft Mail and to take market share from other email systems evolved into a head-on fight with Lotus Notes, from which Microsoft emerged triumphant at the start of the 21st century. Since then, the focus has been on making Exchange easier to manage, cheaper to deploy, and better resistant to failure. Much of this work can be seen in Exchange 2007 and 2010 in features such as the use of Windows PowerShell as the basis for administration, a steady reduction in I/O demands, and the introduction of different flavors of continuous log replication. Cloud-based services represent the latest competitive threat through offerings such as Gmail.

Microsoft now has a somewhat bifurcated set of development priorities that must continue to satisfy the requirements of customers who deploy "on premise" while also serving the needs of Microsoft's own hosted service that operates in mammoth multitenant datacenters. Exchange 2010 is the first version to be developed under this regime, and it will be interesting to see how Microsoft's focus will move between the hosted and on-premise worlds over the next few years.

The motivation to upgrade

The first point in a deployment project is to understand why you want to deploy Exchange 2010. Different circumstances dictate the ability and willingness of companies to move forward with the deployment of a new version of Exchange, including these common scenarios:

- They currently run a very old version of Exchange, including Exchange 5.5 (released in 1997).

- They might have declined to upgrade from Exchange 2003 to Exchange 2007 because their current infrastructure meets their needs. They might have wanted to avoid the need to buy new hardware to deploy Exchange 2007 or did not want to grapple with the need to understand the new architecture, perhaps because of other priorities within their overall IT infrastructure. Like all new software versions, another reason for not upgrading is that Exchange 2007 did not deliver sufficient or relevant new functionality to justify the cost.
- They might run another mail system and now want to move to Exchange. The vast bulk of these migrations are from Lotus Notes, which continues to lose market share to Exchange. Some migrations from Novell GroupWise from a very small installed base are still seen.

Believe it or not, there are companies that still operate very old Exchange servers. Because it is relatively simple when compared to today's software, Exchange 5.5 is very stable. Although its use has declined over the last few years, there are still some companies that aren't interested in running the latest version and continue to use servers commissioned between 1999 and 2002. Their logic is impeccable and follows the old adage that you shouldn't attempt to fix something that isn't broken. However, although software bits don't degrade over time, hardware does, and the older servers that support versions like Exchange 5.5 or Exchange 2000 are becoming obsolete, as replacement parts become harder to source and replacement servers are so cheap that it's more cost effective to throw the old hardware away if it fails. Hardware is actually a small part of the overall upgrade cost, as new software licenses and the time required to migrate data to a newly installed Exchange 2010 organization will be far more expensive.

Moving from Exchange 2003 or Exchange 2007

Exchange 2003 is another stable platform that has served customers well. Like Exchange 5.5, it has benefited from the work done in previous versions to fix bugs and complete functionality. Faced with the need to buy new hardware and to deploy new 64-bit versions of Windows and other associated applications before they could move to Exchange 2007, many companies opted to stay with Exchange 2003. Although server hardware has been 64-bit capable for a long time, the move to use a 64-bit platform for an operating system and applications introduces some instability and "newness" into the infrastructure. If the infrastructure is reliable, the servers are not due to be replaced, and there is no good business reason to upgrade, then it's easy to understand why people chose to leave things alone. In addition to the hardware refresh, the need to upgrade administrator knowledge to cope with the Exchange 2007 architecture, change operational procedures, and perhaps rewrite some code to use Windows PowerShell instead of Windows Management Instrumentation (WMI) scripts all contributed to the disruption and cost of the migration.

If you run Exchange 2007 today, you may experience less fear of the unknown elements of a new version because much of the Exchange 2010 architecture is an enhancement of Exchange 2007 and is therefore not as new and unknown as it would be if you approach Exchange 2010 from a deployment based on Exchange 2003. Features that made their debut in Exchange 2007—such as continuous log replication—are in their second iteration, and there's a mass of published information from Microsoft and third parties covering topics from basic design approaches to Windows PowerShell code examples that help bridge the knowledge gap.

Some observers referred to the original release of Exchange 2010 as “Exchange 2007 finished,” a comment that is underlined by the completion of the management user interface to support the deployment of features such as retention tags in Exchange 2010 SP1. There's some truth in this view insofar as it is the nature of server software used by a huge variety of companies to constantly evolve and there's no doubt that some of the features introduced in Exchange 2007 have matured further in Exchange 2010. The best example is high availability, but there are others, such as Unified Messaging, where features such as voice mail transcription make Exchange a much more user-friendly platform for voice mail, and the changes made to allow organizations to deploy policy-driven compliance for messaging. Some of these changes rely on additional Microsoft components such as Active Directory Rights Management and won't be as valuable to companies that operate in a heterogeneous IT environment, but they are all signs of building out functionality to meet different needs.

Of course, you now have the choice between running Exchange 2010 on premises or in the cloud, or even in a hybrid configuration where some users are hosted internally and some have their mailboxes in the cloud. The option to adopt an “evergreen” approach to messaging and have Microsoft take care of running Exchange for you will be attractive to some companies and less so to others, but at least the choice now exists.

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 that Microsoft will provide mainstream support for Exchange 2007 (assuming the latest service pack is deployed) until at least November 2012, so there's plenty of time available to deploy and use what is now well-understood technology.

A move to Exchange is usually combined with a deployment of Microsoft Office on the desktop, and the combination of the latest versions of Exchange 2007 and Office 2007 delivers solid results in most cases. The same is true of Exchange 2003, as this product has been around so long that all of its original flaws have now been eradicated or at least

brought to the point where they are well understood and can be avoided in production environments. Depending on how long after the product is released the decision is made and the availability of fixes included in a roll-up release or a service pack, the deployment team might worry that they have to face unknown product limitations or bugs similar to the CAS scalability issues that some early Exchange 2007 deployments faced.

Testing and beta versions

Microsoft goes to great lengths to run beta versions of Exchange internally to validate that it works in enterprise environments. However, someone once observed that running code inside Microsoft isn't really a fair test because users are supported by the massed ranks of the Windows, Exchange, Outlook, and other associated engineering groups. On the other hand, Microsoft will say that their users are among the most demanding on the planet and will find problems where no one else will. To cover the world outside Microsoft, they also have an extensive Technology Adoption Program (TAP) that allows customers early access to code for testing. The companies that participate in the TAP are committed to dedicating considerable resources to installing and testing successive beta versions of Exchange and to using the test software to host real-life production mailboxes. However, no matter how extensive the tests that are performed through these programs, it is unreasonable to expect that Microsoft will discover all of the potential issues that customers will face when software is deployed across a base that spans well over 100 million mailboxes in circumstances from small 50-user systems serving a single office to massive hosting environments.

The problem gets even larger when you consider that Exchange 2010 introduces some major new code, such as the components that support the Database Availability Group, role-based access control, and compliance features. The difficulty of testing new functionality for major products underpins the mantra that you should never deploy Microsoft software until the first service pack is available: It's best to leave others to endure the horror stories experienced in early deployments. Although better testing and programs such as the TAP have improved the situation dramatically in terms of finding bugs and usability issues much sooner in the development process, Microsoft can't shake this perception among the customer base.

From a user perspective, the most obvious gain in moving to a new version of Exchange is the availability of a more functional user interface for Outlook Web App. The last major change that fundamentally improved the Outlook user experience came in Exchange 2003 with cached Exchange mode because it removed a lot of hassle that users experienced in previous versions waiting for messages to synchronize over patchy network connections. Exchange 2010 offers the promise of huge mailboxes and better Outlook performance (available from Outlook 2007 SP2 onward) together with features such as MailTips and archive mailboxes.

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 2010 before they can proceed:

- Will Exchange 2010 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 storage might replace storage area network (SAN) technology.
 - Add-on software might be eliminated because the desired features are now included in Exchange 2010. For example, third-party data replication products can be replaced with Database Availability Groups.
 - Clusters can be replaced with standard servers to remove complexity from the operational environment.
 - Other reasons might also exist.
- What new costs will the company take on to move to Exchange 2010?
 - New servers might be needed.
 - New or upgraded software licenses for Windows Server 2008 or Windows Server 2008 R2, Exchange 2010, and any associated products (third party and Microsoft) are required. To access specific functionality, you might have to purchase enterprise Client Access Licenses (CALs).
 - Replacement of code that depends on deprecated application programming interfaces (APIs) is necessary.
 - Client upgrades (Windows Mobile devices, Outlook 2010, and so on) need to be made.
 - Training for administrators, help desk personnel, and users must be provided.
 - Consulting will be advisable to help to make the transition.

- Apart from basic email functionality, what features in Exchange 2010 does the business need?
 - Will you use Unified Messaging (including integration with other Microsoft products such as Office Communications Server)?
 - Is better high availability required?
 - Will you use archiving and compliance?
- What are the major roadblocks to deployment?
 - The need to upgrade other applications, including rewriting code that depends on now unsupported APIs such as Web Distributed Authoring and Versioning (WebDAV), could cause difficulty.
 - 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 2010.
 - A new version of Outlook must be deployed to take full advantage of the features of Exchange 2010.
- Can I get the same functionality at the same price point elsewhere?
 - Microsoft's Business Productivity Online Suite (BPOS) includes the option to run a hybrid model, where some mailboxes are supported on classic on-premise servers and some run in the cloud. Moving to the cloud seems like a simple decision, but considerable complexity lurks under the surface.
 - A different email platform might be selected, although this introduces additional work items 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 2010, you can proceed to the planning phase.

No in-place upgrades

Microsoft chose not to engineer the code to allow administrators to upgrade a server from Exchange Server 2003 to Exchange Server 2007, and they have gone along the same route for Exchange 2010. The logic was that it is just too difficult to create software that can perform a reliable upgrade from a 32-bit platform of Windows Server 2003 and Exchange Server 2003 to a 64-bit platform of Windows Server 2003 and Exchange Server 2007, even if Windows and Exchange run the latest service pack. There are just too many edge cases that Microsoft won't know about until they are encountered in the field. All of the

Exchange code base and associated products such as Microsoft ForeFront run on the 64-bit platform today, so that's not the major engineering concern.

The problem now is how to accomplish a dual in-place upgrade of operating system and mail server to get to the desired Windows 2008/Exchange 2010 configuration. This is far less of a problem than when the underlying platform changes, as in the case of going from a 32-bit to a 64-bit platform, but it still would require substantial engineering effort to write and then test the code to perform a complete upgrade.

Microsoft's view is that the experience of Exchange Server 2007 deployments proved that it is far easier to introduce new servers and move mailboxes to those servers when you are ready. Such an approach avoids the need to perform in-place database upgrades that would otherwise be required to support the database schema changes such as the major upgrade applied in Exchange 2010. It also eliminates the need to test the installation (setup) program to make sure that it can accommodate the multitude of scenarios that Exchange is deployed into for production.

CAUTION !

The problem with in-place database upgrades is that they are usually slow because every page in the database has to be processed to upgrade it to a new version. The need to process databases introduces a period of vulnerability during the installation process. For example, if your server supports a mailbox database of 100 GB and the data can be upgraded at the rate of 10 GB/hour, you can look forward to a 10-hour period during the installation when the server is fully occupied with the database upgrade. Not only must this processing occur when all users are blocked from using their mailboxes, but if anything happens during the upgrade, you'll have to restart after you fix the problem. Building this kind of data upgrade into upgrades introduces too much risk. From an engineering perspective, it is far better to require customers to install new servers with clean databases and then gradually move users over to the new platform. Although the "no upgrade" approach means that new servers are required for Exchange 2010, it might be possible to align the upgrade with a hardware refresh cycle or to reuse some older servers.

Although customers might incur some extra cost to achieve the upgrade, Microsoft will argue that the time they save from not having to figure out how to make in-place upgrades work (even partially) allows their engineering teams to dedicate time to solving other problems, such as making mailbox moves work more efficiently (which occurs in Exchange 2010), improving the quality and features of the installation program, and upgrading tools such as the Exchange Best Practice Analyzer to help administrators understand any issues that might exist in their infrastructure that must be resolved before an Exchange 2010

upgrade can proceed. In addition, users typically experience less downtime when new servers are introduced because they don't depend on complex upgrade processes. On balance, it's hard to argue against Microsoft's decision, and in any case, we can't do anything about it except order the new hardware to allow the upgrade to Exchange 2010 to proceed.

What version of Windows?

Microsoft supports the deployment of Exchange 2010 on either Windows Server 2008 SP2 or Windows Server 2008 R2 Standard or Enterprise editions. Exchange 2010 is not certified for deployment on Windows Server 2008 Datacenter edition (see <http://www.windowsservercert.com>), and Microsoft initially would not support Exchange 2010 on this platform. Lacking certification doesn't mean that software won't function on a specific version of Windows; instead, it means that the software has not been put through the certification process.

Microsoft reversed their position in early 2010, and you can safely use Windows Server 2008 Datacenter even if Exchange 2010 doesn't boast the official certified logo. It remains doubtful whether the additional features of the Windows Server 2008 Datacenter edition make it an attractive platform for Exchange 2010 because few companies will need to exploit 256 processor cores or hot-add or hot-replace CPUs, especially when these features come with a hefty increase in the cost of the software license. By comparison, Windows Server 2008 R2 Enterprise supports a maximum of 32 processor cores and won't allow a CPU to be replaced or added while the server is running. However, there are bound to be a few companies that will want to explore the Datacenter edition, and it's good that Microsoft will support the deployment of Exchange 2010 on the platform.

The Windows Server 2008 Core, Web, or Foundation server editions remain unsupported and are unlikely to ever be supported given that they are essentially cut-down versions of Windows designed to be deployed to meet specific needs. (It's possible to make some Exchange 2010 roles install on Server Core, but they don't work once installed, so it's not just a matter of Microsoft arbitrarily deciding to block those versions.) No support exists for Exchange 2010 to run on the Itanium (IA64) version of Windows.

Selecting the version of Windows Server 2008 for deployment is a critical decision, as Microsoft does not support in-place server upgrades (with Exchange 2010) from Windows Server 2008 SP2 to Windows Server 2008 R2. Given the relative age of the operating systems, you are likely to use Windows Server 2008 R2 sometime in the next couple of years. Therefore, it is an excellent idea to consider using Windows Server 2008 R2 as the basic operating system for your Exchange 2010 deployment. This is much better than creating a situation in which the only way that you can upgrade to Windows Server 2008 R2 is by deploying a set of new Exchange servers on new Windows Server 2008 R2 and moving mailboxes over to them and then decommissioning the old Windows Server 2008 SP2

servers. It also makes sense to run the same version of the operating system and Exchange on every server in the organization, as this makes support and administration much easier.

Another point to take into consideration is that Windows engineering has made improvements in some of the critical components affecting Exchange that make Windows Server 2008 R2 the best choice for specific servers. For example, testing done by the Exchange development group demonstrates that Remote Procedure Call (RPC) over HTTP performance is better in Windows 2008 R2 than in Windows Server 2008 SP2. This has a direct influence on the ability of a CAS server to handle Outlook Anywhere connections and means that Windows Server 2008 R2 is a better platform for Internet-facing CAS servers. See <http://msexchange team.com/archive/2010/04/30/454805.aspx> for details of the performance tests that make this point.

INSIDE OUT

Upgrading workstations used for management

The Exchange 2010 administration tools can run on either Vista SP2 (x64) or Windows 7 (x64) workstations, so you might need to upgrade workstations that you want to use for management. You can run the Exchange 2007 SP2 administration tools on the same workstation, provided that you install the Exchange 2007 tools first and then install the Exchange 2010 administration tools. Alternatively, you can simply use Windows terminal services to connect to the servers that you want to manage from Vista or Windows 7 workstations.

Preparing for Exchange 2010

Apart from deciding on the operating system, what actions can you take to prepare for an eventual deployment of Exchange 2010, assuming that you run an earlier version of Exchange today? The following is a non-exhaustive list that should be supplemented with details of your particular environment, including items such as applications that depend on Exchange.

- If you already operate an earlier version of Exchange, you should run the Exchange Best Practice Analyzer (ExBPA) tool regularly to identify any problems that can be found by validating the details of your infrastructure against Microsoft's best practice database.
- Be sure to check for required upgrades and hot fixes before you install servers. Exchange affects many parts of the operating system and has a track record of exposing weaknesses. Microsoft IT discovered a problem with NTFS deadlocks on heavily

loaded mailbox servers soon after they deployed Exchange 2010 internally. This problem is specific to Windows 2008 SP2 and required administrators to kill Store.exe to free the deadlock condition, so it was pretty serious. Microsoft fixed the problem quickly (see KB974646 for details), but it's a good example of the kind of problem that comes to light when new combinations of operating system and applications go into production.

- If you haven't already done so, you should move your Active Directory to Windows 2003 forest functional mode (or higher). Exchange 2007 shares the same requirement and there is no good reason to keep Active Directory at a lower functional level. Deploy Active Directory domain controllers and global catalog servers on 64-bit Windows Server 2003 SP2 or, even better, on Windows 2008 SP2 or R2. Note that Exchange does not support domains that have an underscore in their name because of an internal dependency on X.509 certificates, which cannot contain this character.
- Remove any Exchange server that runs Exchange 2000 or earlier versions as they cannot be installed in a forest that supports Exchange 2010. If you still run Exchange 2003, make sure that these servers run SP2 as this is the version that can coexist with Exchange 2010 inside an organization.
- Exchange 2007 servers must be upgraded to SP2 (or later releases). We'll discuss this topic in more detail in just a little while.
- Decide on the version of Exchange 2010 you will use. The choice is between the standard edition and the enterprise edition. See "Exchange 2010 Editions" later in this chapter for more information on the features supported by each version. Note that you can upgrade from the standard to enterprise edition but you can't downgrade from enterprise to standard. If you intend to use the new Database Availability Group high availability feature, you need to run the enterprise edition of either Windows Server 2008 SP2 or R2; bear in mind that you can't upgrade an existing Windows installation from the standard to the enterprise edition without performing an upgrade installation that might or might not cause problems for other applications.
- CALs are also required for every user who connects to Exchange 2010. Standard and enterprise versions are available. The enterprise version is additive, meaning that you also have to buy a standard CAL for each user. You need the enterprise CAL to be able to use features such as Unified Messaging, advanced journaling, and archive mailboxes.

The test plan

Successful preparation always involves dedicated effort to test and test again to ensure that new software works well in the environment run by a specific company. Microsoft cannot be expected to test for every possible condition that software encounters in the wild, so

you have to protect yourself by designing and executing a comprehensive test plan. The plan should address these points:

- All clients used by your company (in all versions) have to be verified against Exchange 2010. The list might include:
 - All versions of Outlook that you currently use. Note that no version prior to Outlook 2003 SP2 is supported by Exchange 2010.
 - The features and functionality available in Outlook Web App 2010 for the browsers that you use (Internet Explorer, Chrome, Opera, Firefox, Safari), including the various platforms that these browsers run on, such as Windows, Linux, UNIX, and Apple Mac.
 - Internet Messaging Access Protocol 4 (IMAP4) and Post Office Protocol 3 (POP3) clients (Eudora, Thunderbird, and so on) on whatever operating system platforms you use.
 - Entourage and other Mac solutions. If you are using Office 2008, you need the Exchange Web Services version of Entourage 2008 to connect Entourage to Exchange 2010. In late 2010, Microsoft shipped a new client, Outlook for Mac, as part of Office 2011. It is a worthwhile upgrade if you have Mac users currently running Entourage.
 - Mobile clients (Windows Mobile, other ActiveSync clients, Apple iPhone, Palm Pre, Android devices, and so on).
- The outcome of the client test plan might result in a number of steps that you have to take before or during the Exchange 2010 deployment, including:
 - Consider the deployment of Outlook 2007 SP2 (or later) as soon as possible to benefit from better support for large mailboxes and improved overall performance. Exchange 2010 does not support versions before Outlook 2003, and it's really best to upgrade to Outlook 2007 to get other features such as Autodiscover.
 - Some Exchange 2010 features (such as MailTips) do not work with Outlook unless you deploy Outlook 2010, so consider how your plans (if any) to deploy Office 2010 might influence your plans to introduce Exchange 2010.
 - Opt for Windows Mobile devices that run at least version 6.0 (Windows Mobile 6.5 or Windows Phone 7 devices are preferred). If you don't use Windows Mobile, select devices that support ActiveSync rather than depend on the IMAP or POP3 protocols to support mobile access to mailboxes.

Testing for operational processes

Apart from the purely functional testing, you should also test how well Exchange 2010 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 as both features are deprecated in Exchange 2010. Use cluster continuous replication (CCR) or standby continuous replication (SCR) instead, as these are closer to the technology used in the new Database Availability Group that replaces both CCR and SCR in Exchange 2010.
- If you use a third-party data replication solution to protect mailbox data, consider whether the new replication features of Exchange 2010 will replace or complement your existing solution.
- If you use tape-based backup solutions for Exchange, you need to consider how to use a solution based on Volume ShadowCopy Services (VSS) instead. Exchange 2010 no longer supports backups made with the streaming backup APIs that have been around since Exchange 5.0, and that means no tape backups. Do not underestimate the work required to move from tape-based backups to VSS-based backups, especially in terms of complying with auditing requirements, off-site storage, and so on.
- If you use a third-party archiving and compliance solution, have a discussion with the vendor to understand their go-forward plan to work with or move to the archiving and compliance functionality that is in Exchange 2010. The ideal situation is that the third-party solution will interoperate seamlessly with the base features built into Exchange. If you don't use archiving today, you might want to consider increasing mailbox quotas so that users can keep more information in their mailboxes that is eventually archived by Exchange 2010. Note that this approach has consequences for storage and backup operations.
- Discuss the permissions model used in your company to control access to Windows resources and applications to ensure that the role-based access control model introduced by Exchange 2010 meets the company's security and organizational needs. Exchange 2010 SP1 includes support for a split permissions model (see Chapter 4, "Role-Based Access Control") that will interest companies that like to keep a clear and 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. You should test to ensure that you will be able to continue to deliver the same degree of service to end users after Exchange 2010 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 Exchange 2007 to ensure that everything is transferred over to Exchange 2010. Microsoft makes sure that customizations such as transport and journal rules are transferred on the deployment of the first Exchange 2010 server in an organization, but you should know what you have so that you know what to check is still working for Exchange 2010. You will have to reapply customizations such as changes to OWA 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, some settings and data objects, such as Secure Sockets Layers (SSL) certificates and Microsoft Internet Information Services (IIS) settings, are stored outside Active Directory and must be manually updated for Exchange 2010.
- Understand what use is made of older Exchange APIs such as WebDAV, which Exchange 2010 replaces with Exchange Web Services (EWS). You will probably have to rewrite any custom code using EWS or look for another solution that can be used with Exchange 2010, such as a Windows PowerShell script. Alternatively, you can consider keeping an Exchange 2007 server around 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 tasks such as system monitoring or account maintenance to ensure that it will continue to work with remote PowerShell and Exchange 2010. For example, Microsoft Identity Lifecycle Management (ILM) V1 includes the option to “enable Exchange 2007 provisioning” to allow administrators to update Exchange data as part of the provisioning process.

Note

Behind the scenes, ILM requires administrators to install the Exchange Management Shell on the same server to be able to call the Update-Recipient cmdlet. However, ILM V1 is a 32-bit product that depends on the ability to use 32-bit Windows PowerShell for its integration with Exchange. Microsoft has updated Forefront Identity Manager 2010 (the successor to ILM) to allow it to call remote PowerShell on an Exchange 2010 server, so if you’re using ILM, you should move to FIM 2010 as part of your Exchange 2010 migration.

- Prepare a deployment plan to introduce Exchange 2010 servers into the organization, taking into account restrictions such as the inability of OWA clients to open public folder replicas that reside on Exchange 2007 servers (OWA 2010 can only open public folder replicas on Exchange 2010 servers). The suggested order of deployment is CAS servers, then Edge and hub transport servers, and finally mailbox servers (and Unified Messaging servers, if you use Unified Messaging). Smaller deployments might find that their needs are best met by running Exchange 2010 on one or more multirole servers.

Of course, 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 totally different approach such as outsourcing Exchange to a third party using either a traditional on-premise deployment or a cloud-based approach through Microsoft BPOS.

Bringing Exchange 2007 up to speed

If you run Exchange 2007 today, it's likely that you have deployed SP2 or SP3. Microsoft released SP2 in August 2009 and SP3 in mid-2010. These releases are part of Microsoft's regular release schedule to provide bug fixes and enhancements to customers. SP2 is also an important stepping stone to Exchange 2010 because it upgrades the Active Directory schema to the revision level required by Exchange 2010. The schema upgrade allows Exchange 2010 and Exchange 2007 servers to coexist within the same organization. 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 2007 SP2 or SP3 is that these versions support Windows PowerShell 2.0, which means that you can use the same version of Windows PowerShell to work with both Exchange Server 2007 and Exchange Server 2010. If you want to deploy Exchange 2007 on Windows 2008 R2, you need to deploy Exchange 2007 SP3. It is not a requirement to deploy Exchange 2007 SP3 to support Exchange 2010 SP1, as Exchange 2010 is quite happy to work alongside Exchange 2007 SP2 or SP3. This being said, Exchange 2007 SP3 supports Windows 2008 R2, and is therefore the natural partner to deploy alongside Exchange 2010 SP1.

Moving to a new service pack for Exchange 2007 might also require you to upgrade other products. For example, if you use Microsoft Forefront Security for Exchange Server for antivirus and anti-spam protection, you need to deploy its SP2 release alongside Exchange 2007 SP2. The need to coordinate upgrades for different products adds some time and complexity to the upgrade project. Additional downtime should also be factored into your plans if you run clustered Exchange 2007 servers (SCC or CCR) that need to be upgraded.

INSIDE OUT

A general rule

The general rule is that all Exchange 2007 servers in the Active Directory site in which you first introduce Exchange 2010 must run SP2 (or later), as must all CAS and Unified Messaging servers in the entire forest. You also need to install SP2 or SP3 on Exchange 2007 mailbox servers from which you intend to move mailboxes to Exchange 2010.

Deploying earlier versions of Exchange servers alongside Exchange 2010

The possibility exists that you might want to deploy some Exchange 2007 servers inside a brand new Exchange 2010 organization. On the surface, this shouldn't be a problem because Exchange 2010 can coexist with Exchange 2007 in the same organization. However, the order in which you deploy servers is important. If you begin an organization with an Exchange 2010 server, you will only be able to install Exchange 2010 servers in that organization because beginning with Exchange 2010 will mark the organization as incapable of supporting earlier versions. Therefore, if you think that you will need to use Exchange 2007, you should start by installing an Exchange 2007 server first, then adding Exchange 2010 servers as required. The Exchange 2007 server doesn't have to be active. It simply has to exist in the organization to create the conditions to permit support of a hybrid organization.

The same situation exists if you are running Exchange 2003 today and want to deploy Exchange 2010. If you go ahead and deploy an Exchange 2010 server into the organization, you will never be able to deploy an Exchange 2007 server into that organization; a side effect of installing Exchange 2010 server is a block on the deployment of anything else but an Exchange 2003 or Exchange 2010 server into the organization.

If you think that you might need to use Exchange 2007 in the future (perhaps to support an application that has not yet been upgraded to support Exchange 2010), you need to install an Exchange 2007 multirole server (everything except Unified Messaging). However, you don't need to ever use the Exchange 2007 server, and it can exist as a virtual server that does nothing except provide evidence to the Exchange 2010 setup program that Exchange 2007 is present.

Once you reach the point when Exchange 2007 will never be required again, you can remove the server, but be sure before you proceed, because if you remove the Exchange 2007 server from the organization and then install a new Exchange 2010 server, the setup program will block future deployment of Exchange 2007.

Web-based Deployment Assistant

There's a lot of new technology in Exchange 2010 and system administrators sometimes need help to know how best to approach a deployment. You can employ consultants to help, and this is certainly a recommended option for large or complex deployments. Microsoft has been working on tools to help customers manage and optimize Exchange. The latest is the Exchange 2010 Deployment Assistant, which is available online at <http://technet.microsoft.com/exdeploy2010>.

The Deployment Assistant is based on a lot of real-world feedback from experts and customers. It allows you to explore the recommended steps to deploy Exchange 2010 in different scenarios, including upgrades from Exchange 2003 and Exchange 2007 and a green-field deployment. The idea is that you answer a few questions to set the basic context for the deployment and the assistant then generates a recommended set of steps in a PDF file that you can download.

Note

The output generated by the Deployment Assistant contains a lot of good information about the work activities that have to be performed to deploy Exchange 2010, but it should be regarded as no more than a first draft. Automated tools can't be expected to know everything that might influence your organization, so you have to be prepared to add detail to provide the necessary depth for a fully developed deployment plan.

Exchange 2010 editions

Exchange 2010 is available in three editions. The standard edition is the most common, as it contains all of the necessary functionality required by an email server. It is limited to five databases (including a public folder database if deployed). If you attempt to mount a sixth database, Exchange 2010 Standard Edition will gracefully refuse and issue event 9591: Exceeded the max numbers of 5 MDBs on this server. You also have to operate under the same restriction if you install a trial version of Exchange 2010. The standard version of Exchange 2010 supports the new Database Availability Group (DAG) high availability feature. Microsoft made the decision to include DAG support in Exchange 2010 standard edition very late in the development cycle to allow all sizes of customers to deploy high availability solutions. However, to use a DAG, you need to deploy Exchange on the Enterprise edition of Windows Server 2008 SP2 or R2 to provide the Windows Failover Clustering feature that underpins the DAG.

Exchange 2010 Enterprise edition is designed to meet the needs of large enterprises. You can mount up to 100 databases on a server running the Enterprise edition, so it is the

obvious choice for large-scale deployments. The Coexistence edition (also known as the gateway edition) is only used to facilitate connectivity and migration from previous versions of Exchange to the Microsoft BPOS-managed version of Exchange 2010; you aren't supposed to deploy it for normal production use even though there are no technical blocks that prevent you from doing so.

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 requires an enterprise CAL per mailbox, as does voice mail integration, the discovery and retention features, extended Exchange ActiveSync policies, and use of Microsoft's Forefront anti-spam protection for Exchange 2010. The enterprise CAL is additive in that you first buy a standard CAL to license the standard features and then purchase an enterprise CAL if you need to use the extended features. U.S. list prices at product announcement were \$55 for the standard CAL and \$35 (additive) for the enterprise CAL, \$550 for Exchange 2010 Standard edition, and \$3,200 for Exchange 2010 Enterprise edition. However, these are very much guide prices and are subject to volume discounts and other negotiations; you should refer to the Exchange 2010 licensing page at <http://www.microsoft.com/exchange/2010/en/us/licensing.aspx> to get an idea of what licenses are required for the specific features you want to use.

INSIDE OUT

How many enterprise CALs do you need?

The Exchange 2010 version of Exchange Management Console (EMC) has been updated to report the number of standard and enterprise CALs that you need to support users, and it now highlights features that trigger the need for an enterprise CAL. For example, if you create a new mailbox that has an associated archive, EMC flags that the archive requires an enterprise CAL. The RTM version sometimes produced CAL counts that were just plain wrong. The code used by SP1 is better, but you are still advised to check its CAL count to make sure that it is right.

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 older Exchange directory services, and there is no doubt that Microsoft learned a lot about how to design and operate directories from the experience gained with Exchange 4.0 to Exchange 5.5.

The strong link between Exchange and Active Directory

By any measure, Exchange makes the most extensive use of Active Directory of any Microsoft application. The Exchange 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 to 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 allow them to work with Exchange. Other attributes are created for new objects and are only used for those objects. For example, the ability to define a maximum number of active databases that a server can support is dependent on the *msExchMaxActiveMailboxDatabases* (almost all of the objects that belong to Exchange are prefixed with “msExch”) attribute that Exchange 2010 adds when preparing Active Directory for deployment.

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 added by Exchange, as 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, to identify objects.

- Creates a container called Microsoft Exchange under the Services root in the Configuration Naming Context. All the configuration data about objects managed by Exchange are located here (Figure 1-1). 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, which means they will normally be found in the organizational units in the default naming context. Mail-enabled objects use the extended schema to populate attributes such as email addresses. EMC relies heavily on data fetched from the Exchange container. The data are replicated around the forest to ensure that a common and consistent view of the Exchange organization is available everywhere. This fact underlines how Active Directory replication supports Exchange operations.
- 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 are available to every domain within the forest. For example, Exchange adds many mailbox attributes, such as the database and server that currently hosts 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 is added to the PAS so that it is available everywhere.

- Adds a set of extended rights (permissions) to allow administrators to manage objects such as databases.
- 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 having to deal with each of the individual properties that make up a set. For example, the Exchange Personal Information property set contains all of the attributes that Exchange holds about individual users, such as their phone numbers.

The following table summarizes the position by showing where Exchange stores the data that it relies on 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 once a domain controller learns that a schema update 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, a feeling that can be compounded when the scale of the upgrade required for Exchange 2010 is revealed.

This fear can be addressed by understanding that schema updates will replicate quickly throughout a forest if Active Directory replication is healthy. The risk to other applications can be reduced by scheduling the schema upgrade for a time of low user demand. For this reason, many companies schedule Active Directory upgrades over holiday weekends. It's a good idea for you to validate that Active Directory replication is healthy with the RepAdmin utility before you commence the upgrade, just in case some problems have crept into the forest unnoticed.

ADSIEdit

ADSIEdit has proven its worth to Exchange administrators time after time to fix problems with Active Directory objects or simply to help them understand the complex relationship between Exchange and Active Directory. Figure 1-1 illustrates the utility in use to browse the Microsoft 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 15, “Compliance,” 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 Windows 2008 servers along with the Active Directory Domain Services role, after which you can access ADSIEdit through the Start menu Administrative Tools option. Once you start ADSIEdit, you need to connect it to the Configuration Naming Context to be able to access Exchange data.

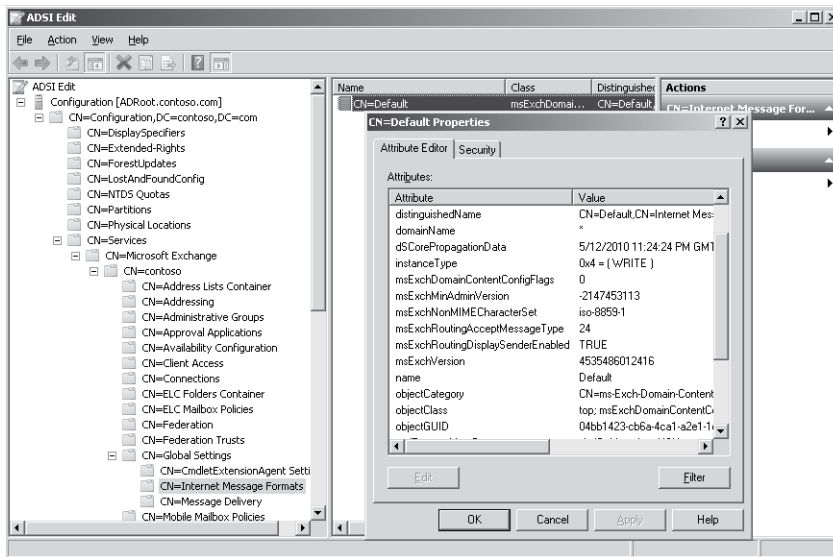


Figure 1-1 Exchange 2010 objects in Active Directory.

ADSIEdit started life in Windows 2000 as an add-in for Microsoft Management Console (MMC) that Microsoft didn't really support, probably because they didn't want administrators to have the ability to mess up Active Directory by editing objects and properties. If you've used ADSIEdit with previous versions of Windows, you'll see that work has been done in Windows 2008 to polish the program. However, you still work with raw data and a user interface that doesn't attempt to do much to protect you with warnings about the potential consequences of certain actions, such as editing an attribute.

CAUTION !

There is no doubt that 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 working of applications such as Exchange that depend on Active Directory to hold their configuration data.

Types of Active Directory deployments that support Exchange

Active Directory implementations in use today extend from a single forest to multiple forests with many variations in between. Some companies deployed Active Directory early and have an implementation that features a root domain which 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. Others have deployed multiple Active Directory forests because separate security contexts are required by different businesses. Others have acquired multiple forests through buying companies and are in the process of 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 that Active Directory is under the control of a separate team from the group that manages Exchange. It's obvious from the previous discussion about how Exchange leverages Active Directory that close coordination is required from all concerned to achieve a smooth deployment.

Exchange supports three basic Active Directory designs. Apart from understanding that a forest can only support a single Exchange organization (there is always a one-to-one mapping between a forest and an Exchange organization), the biggest thing to remember is that a forest (and not a domain) is the security boundary for Active Directory. Thus, once you deploy multiple forests, you have to make sure servers and user accounts are able to access the resources that they need to do work, no matter in what forest the resource is located.

- **Single forest:** This is the simplest design and the one that is normally most appropriate for small to medium deployments as 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 no difficulties are usually encountered with security and permissions, as all objects operate within a single security context.
- **Multiforest:** 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 each forest so that users can access a common directory that includes everyone. Exchange can be deployed in one or more forests.

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. Multiforest deployments usually occur as the result of company acquisitions. Less commonly, companies have deployed Exchange in multiple forests because of the need to satisfy particular requirements for specific parts of the company that were inconsistent with the needs of other parts. Exchange 2010 supports the ability to move mailboxes between organizations, so this is not an obstacle. It's also possible to use the Exchange 2010 management console 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 where 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, as 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 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 sound advice to start simple and deploy a single forest to support Exchange. The old KISS ("Keep it simple, Stupid") adage comes to mind here! Later on, if you find that this model does not meet business requirements, you can evolve it by adding additional forests or even additional 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.

The role of ADAccess

Operational Exchange servers make constant connections with Active Directory to retrieve directory information that is then loaded into on-server caches and used for many purposes that we explore throughout this book. A component called ADAccess, which runs as part of the System Attendant service, is responsible for providing all other Exchange services with a consistent, common view of the Active Directory topology as well as the population and maintenance of a cache that the services can consult when they need to retrieve information about the Active Directory topology.

ADAccess also performs tests to ensure that servers that are indicated to be domain controllers and global catalog servers are functional enough to be used by Exchange, as there is no point in Exchange attempting to read configuration data or to make a routing decision based on data from a barely functional or uncontactable Active Directory server. ADAccess monitors the availability of Active Directory servers within the local site and will instruct Exchange to failover from one server to another if an Active Directory server goes offline, including situations when all of the Active Directory servers in the local site are unavailable and Exchange needs to contact a domain controller in a remote site. In this scenario, ADAccess will make the decision about what server to contact based on Windows site link connection costs. ADAccess detects when the local Active Directory servers come online again and will redirect Exchange to a local server as soon as one is available.

If you examine the properties of a server and open the System Settings tab (Figure 1-2), you'll see the set of domain controllers and global catalog servers used by Exchange. Domain controllers are used to provide configuration data to Exchange, and global catalog servers are the source of directory information about mail-enabled objects from anywhere in the forest. As you can see from Figure 1-2, the same server can perform both roles.

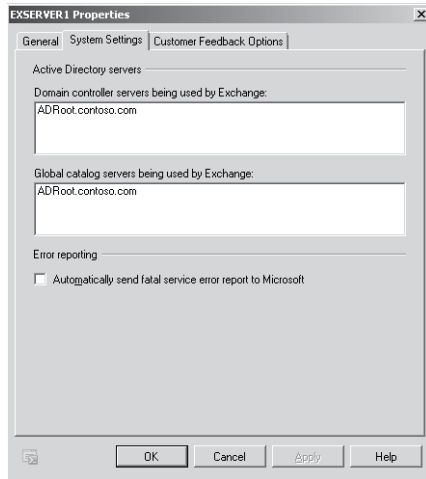


Figure 1-2 Active Directory servers used by Exchange.

Every 15 minutes, ADAccess runs the Topology Discovery process on Exchange servers to discover the domain controllers and global catalog servers that are available. Topology Discovery checks what Active Directory servers are available within the local site and those that are available outside the site (local servers are always used whenever possible). Event 2080 is logged after each discovery, and you can discover some additional detail about the servers by examining the data reported in the event log. By comparison, Exchange logs event 2070 when ADAccess cannot locate any suitable global catalogs during its attempts to discover the Active Directory topology.

The results reported in Figure 1-3 show that only one in-site server has been found. Apart from the fully qualified domain name (FQDN) of the server, ADAccess reports a cryptic set of letters and numbers for each server that it discovers. The letters indicate what role the server can play for Exchange. D indicates that the server is a domain controller; G means that it is a global catalog; C means that the server is acting as the Configuration domain controller.

In Figure 1-3, we see that Topology Discovery reports “CDG” for both servers, meaning that each is able to act as the Configuration domain controller, a regular domain controller, and a global catalog server. Exchange inserts a hyphen in the position when a server cannot be used for a particular purpose. For example, a server listed as “CD-” is a domain controller but not a global catalog. Table 1-2 shows how you can interpret the numeric values reported by ADAccess. Note that Exchange 2010 does not support read-only domain controllers (RODCs). These servers can exist in your Active Directory, but Exchange needs to connect to a writable domain controller.

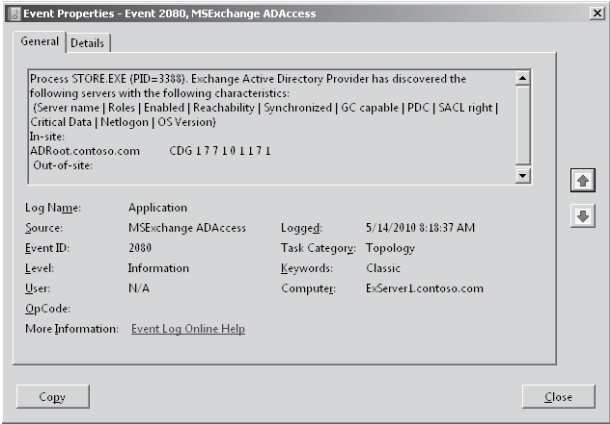


Figure 1-3 ADAccess discovers Active Directory servers.

Table 1-2 Interpreting ADAccess flags

Position	Use	Meaning
1	Availability	1 indicates that the server is available.
2	Port access	Bit mask indicating what ports are available for LDAP access via a Transmission Control Protocol (TCP) connection. 1 means that LDAP access for global catalog requests is possible through port 3268; 2 means that the server is reachable for domain controller requests through port 389; and 4 means that the server can act as the configuration domain controller. 7 indicates that the server is available for all necessary ports. 0 means that the server is unreachable and not usable by ADAccess.
3	Synchronization status	Bit mask indicating the Active Directory synchronization status of the server as indicated by the isSynchronized flag on the rootDSE object. 1 indicates that the global catalog is synchronized; 2 that the domain controller is synchronized; and 4 that the configuration domain controller is synchronized. 7 means that the server is completely synchronized in terms of Active Directory.
4	Global catalog flag	1 indicates that the server is a global catalog server; 0 means that it is a domain controller.
5	PDC flag	1 indicates that the server is the primary domain controller (PDC) for the domain; 0 means that it is not.

6	SACL test flag	1 indicates that ADAccess has the necessary security permission (via the system access control list [SACL]) to read Exchange information from the directory; 0 means that it does not.
7	Critical data flag	1 indicates that ADAccess located the Exchange server that it is running on in the configuration naming context of the domain controller. This container stores critical data such as the names of Exchange servers, the roles installed on each server, routing data, and so on. ADAccess only selects a domain controller if it hosts this container.
8	Netlogon	Bit mask indicating the success of ADAccess in connecting to the NetLogon service running on the domain controller using RPC. Similar to the results reported for LDAP access, 7 means that all attempts to connect were successful.
9	OS Version flag	1 indicates that the domain controller runs a version of Windows that supports Exchange 2010 (Windows Server 2003 SP1 or higher, SP2 recommended); 0 indicates that it does not.

INSIDE OUT

Monitoring SACL

The SACL flag is worthy of further comment. If the SACL for the Exchange Servers group is removed from a domain controller, many Exchange services, including the Information Store, will refuse to start. Many support calls were generated as a result of overeager administrators cleaning up Active Directory and removing the SACL with unfortunate results the next time Exchange started. Microsoft added code called the SACL Watcher to Exchange 2010 SP1 to monitor the presence of the SACL. Every 10 minutes, the watcher code runs to check the SACL and if it is missing, the watcher adds the SACL back.

Exchange 2010 caches configuration data from Active Directory in memory and shares the data with all the services that need to use it. However, unlike some previous versions of Exchange, it does not cache recipient data, as the belief is that it is better to fetch the definitive version of data from the Active Directory rather than run the risk that cached data might be outdated.

TROUBLESHOOTING

EMC can't contact a domain controller

Errors occur with computers and systems sometimes go offline. Unless it is the only domain controller or global catalog in a site, it shouldn't matter too much if you need to take an Active Directory server offline to apply a hot fix or other upgrade, as Exchange will quickly detect that the server is offline and switch its attention to another server. In some circumstances, the point at which a server goes offline could interfere with EMC and cause a slight hiccup. For example, Figure 1-4 shows a problem that occurred when EMC attempted to create a new mailbox. The domain controller to which EMC was connected when the administrator began to run the New Mailbox Wizard was taken offline, and the operation failed after a timeout (which accounts for the 21 seconds reported for the attempt). The usual solution is to exit EMC to force it to connect to a new global catalog server and try again.

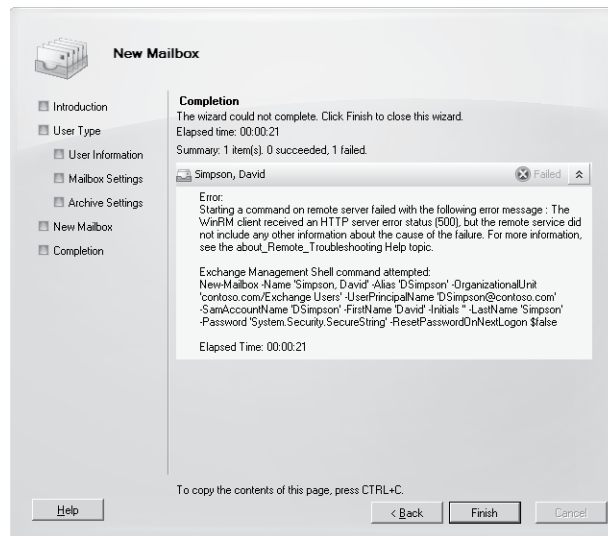


Figure 1-4 An Active Directory error occurs when creating a new mailbox.

Planning for global catalogs

The basic rule is to place at least one global catalog server in each site that hosts an Exchange server. This is probably sufficient for small sites that host one or two Exchange servers, but as the number of Exchange servers grows and the need to ensure availability increases, you will need to position more global catalog servers to handle the load generated by users and Exchange and to ensure that service continues even if one of the global

catalog servers fails. This then raises the question of just how many global catalog servers are needed within a site.

Global catalog servers are critical to Exchange. For example, the transport system cannot expand a distribution group's membership or execute a query to determine the membership of a dynamic distribution group if a global catalog is unavailable. Thus, if you don't provide sufficient global catalog capacity within a site, your Exchange servers won't function smoothly, no matter how powerful and well configured they are. The classic trap that people fall into is to assume that the relatively light load exerted on global catalogs by standard Windows processing (client logins and so on) persists after Exchange comes on the scene. This could not be further from the truth, as demand on a global catalog expands dramatically once Exchange is active. The general rule of thumb is therefore to follow these recommendations:

- Servers running a 32-bit operating system: Provide one global catalog processor core for every four processors dedicated to Exchange.
- Servers running a 64-bit operating system: Provide one global catalog processor core for every eight processors dedicated to Exchange.

In other words, in a site supporting four Exchange servers, each equipped with four-way processor cores, you have 16 processors dedicated to Exchange and therefore need to deploy two 64-bit processor cores (or four 32-bit cores) for global catalogs. Two servers equipped with two-way processor cores would be a good solution in this case. Dividing the load across two servers means that Exchange will be able to continue working even if one global catalog is unavailable, and most servers have at least two processor cores today. An investment in global catalogs is a good thing, as there is nothing quite as good as stopping Exchange dead as the unavailability of a global catalog.

Servers running Windows 2008 are a better choice for global catalogs and should be the basic choice for deployment. Apart from being a more modern platform than the 32-bit Windows Server 2003 alternative, the servers will be able to address more memory and load more of the Active Directory into memory for speedier access. In all but the largest deployments, you can easily equip the server with sufficient memory to cache the entire Active Directory, which is the ideal situation.

Another issue that you need to consider is how to position global catalogs alongside Exchange servers. Again, we have a simple rule of thumb to follow: You need a global catalog on every wide area network (WAN) segment that supports an Exchange server. In other words, you must ensure that you position global catalogs so that any network outage has a minimal effect on Exchange. Satisfying the rule might not always be possible, especially if you feel that you have to place an Exchange server in a branch office (to speed up email for local users) and cannot afford to add a global catalog.

CAUTION !

Exchange does not support Windows 2008 RODCs that are often recommended for branch deployments. In this case, you might consider directing Active Directory look-ups to a global catalog server across a WAN link. However, although it seems an attractive option to use the network in this way, this approach inevitably leads to outages and user dissatisfaction. The better option is either to follow the rule of thumb and deploy a local global catalog in each site or to centralize Exchange and global catalog servers together in a single datacenter and use clients such as Outlook running in cached Exchange mode to connect over the WAN.

Exchange 2010 does not support Windows 2008 RODCs. You can still deploy RODCs in your infrastructure, but you have to make writeable domain controllers and global catalog servers within the Active Directory sites that host Exchange servers. Any RODC will be ignored by Exchange.

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. The basic framework for preparation is as follows:

1. Make sure that the forest runs in Windows Server 2003 functional mode. The computers used for domain controllers and global catalog servers should run Windows Server 2003 SP2 at a minimum, with Windows Server 2008 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 of 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 /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 to allow them to 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. You can also run setup with the /PrepareAllDomains switch to prepare all of 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 be able 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.

INSIDE OUT

Schema upgrades and replication activity

Schema upgrades might take some time to perform, and you probably don't want a lot of replication activity going on while you upgrade the schema master. You can disable replication on the schema master while the upgrade progresses and then start replication after all the upgrades have been applied. This is done with the RepAdmin utility. For example, to disable replication:

```
RepAdmin /Options Schema-Master.contoso.com +Disable_Outbound_Rep1
+Disable_Inbound_Rep1
```

If the upgrade is successful, you can then restart replication with:

```
RepAdmin /Options Schema-Master.contoso.com -Disable_Outbound_Rep1
-Disable_Inbound_Rep1
```

The joys of command-line utilities and their wonderful syntax!

Although the most common need for schema extensions arises when Exchange 2010 is first deployed for an organization, it's also important to recognize that schema updates might be required before a build-to-build upgrade can be performed. For example, Exchange 2010 SP1 introduces some new schema upgrades that must be applied to Active Directory before you can deploy the first Exchange 2010 SP1 server.

If you are installing Exchange from scratch to build a brand 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 then proceed to install the new server. Subsequent server installations add their own objects to the organization data. Software upgrades such as the deployment of a service pack or set of roll-up fixes can also affect the information held in Active Directory.

One way to check the current version is to examine the value of the *objectVersion* property of the Microsoft Exchange System Objects (MESO) object. This is a container that holds objects that Exchange uses for internal processing. The value of the *objectVersion* property is usually increased following a successful software upgrade. The value of the *objectVersion* property for the Exchange 2010 RTM version is 12639 (build 639.21), while Exchange 2010 SP1 (build 218.15) displays 13214.

Another check that is often performed is to look at the *rangeUpper* property of the *Ms-Exch-Schema-Version-Pt* object. The value of this property for the RTM release of Exchange 2010 is 14622. Table 1-3 lists the relevant values for each version of Exchange that has required an Active Directory schema upgrade since the original extension occurred for Exchange 2000. See Chapter 2, “Installing Exchange 2010,” for more information about how you can interpret Exchange version numbers.

Table 1-3 Exchange versions and schema upgrades

Value of rangeUpper	Exchange Version
14726	Exchange 2010 SP1
14622	Exchange 2010 RTM or Exchange 2007 SP2
11116	Exchange 2007 SP1
10628	Exchange 2007 RTM
6870	Exchange 2003 RTM
4406	Exchange 2000 SP3
4397	Exchange 2000 RTM

Microsoft recommends that you do not install Exchange on a domain controller. This is good advice and it is wise to maintain as much separation as possible between Exchange and Active Directory, even in small installations where extra servers impose a significant cost in terms of the overall deployment. The logic is that having Active Directory on the same server as Exchange creates additional complications in case the server experiences an outage and has to be rebuilt. At worst, if one server supports Active Directory and Exchange and is the only server in use, any outage creates an immediate disaster, where you can expect that the server will be unavailable to users for a significant time and some data are likely to be lost. It can take significant time and effort to recover from situations like this, so it is best to be safe and maintain separation if at all possible.

As is obvious from Table 1-3, Exchange 2007 SP2 uses the same schema version as the RTM version of Exchange 2010. However, a caveat exists that governs the existence of

down-level (Exchange 2007 and Exchange 2003) servers in an organization where Exchange 2010 is present. The general rule is that the installation of Exchange 2010 server as the first server in the organization blocks the installation of any down-level server thereafter. Therefore, if you want to install an Exchange 2003 SP2 or Exchange 2007 SP2 server in the organization to support something like an old email connector, you have to make sure that the down-level version is present before you deploy the first Exchange 2010 server. Once Exchange 2010 recognizes the existence of the down-level servers, it will “tolerate” their existence and you can continue to deploy more instances of these servers thereafter for as long as they are needed.

Tip

Deploying legacy servers is not the ideal situation because it is preferable to concentrate on the newer technology, but it is possible that you'll need additional Exchange 2007 servers to meet the needs of applications that might not have been upgraded for Exchange 2010.

The joys of a customizable schema

Active Directory boasts a customizable schema that is used by applications like Exchange to extend the underpinning database for their own purposes. Exchange adds just over 3,000 attributes to Active Directory as part of its installation process before you can install the first server in an organization. This number has increased ever since Exchange first began to use Active Directory in Exchange 2000, and every new version of Exchange continues to tweak the Active Directory schema to accommodate new features. For example, Exchange 2007 was the first version to support unified messaging, so it created a batch of new attributes to sort information about things like phone dialing plans. In its turn, an example of how Exchange 2010 extends the schema is the addition of new attributes to support the DAG. The total impact of the extensions is a growth of approximately 6 MB in the Active Directory database (.dit file).

Ideally, you should run process schema updates before you start to deploy any Exchange 2010 servers by running the Exchange installation procedure with the `/preparead` or `/prepare-schema` switch (depending on whether you are installing a new organization or updating an existing organization). The best approach is to run the installation procedure “close” (in network terms) to the schema master for the forest. The computer used must be a member of the same domain and Active Directory site as the schema master. The procedure unpacks the set of .ldf files included in the Exchange kit and then calls the `Ldifde.exe` utility to process the schema updates that these files contain. Afterward, Active Directory replicates the updated schema to all domain controllers in the forest.

TROUBLESHOOTING

I changed the schema and now Exchange isn't synchronized with Active Directory

Sometimes Exchange and Active Directory become unsynchronized as the result of a change made to the schema for other purposes. For example, KB951710 (fixed in Exchange 2007 SP1 roll-up update 5) describes the problem that occurred when an organization extended the "Company" attribute in the schema to accommodate more than 64 characters. All versions up to and including Exchange 2007 SP1 have code that expects the company attribute to be no more than 64 characters, so any attempt to write more than this amount through EMC or Exchange Management Shell (EMS) results in an error. You can use other tools (like ADSIEdit) to update Active Directory through LDAP and write more than 64 characters into the attribute only to risk a potential corruption when Exchange subsequently attempts to retrieve or set the value.

The same situation is known to have occurred with changes to other attributes such as "Title" and "Initials," and although the Exchange engineering team is willing to consider changing their code to accommodate customer requirements to store more information in Active Directory, there will always be a time lag before Microsoft can issue new code. The moral of the story is that you need to carefully consider the impact of any schema update you make to Active Directory and test it thoroughly against all of the applications that use Active Directory before you apply the change in production.

Finally, if you are moving from Exchange 2003 to Exchange 2010, you must run `Setup.com /PrepareLegacyExchangePermissions` in every domain that supports Exchange 2003 servers. This action makes sure that the Exchange 2003 Recipient Update Service (RUS) continues to function after the Active Directory schema is updated for Exchange 2010. The new schema introduces a new property set called the Exchange-Information property set that the role-based access control model in Exchange 2010 uses to grant permission to users to manage recipient information. Running `/PrepareLegacyExchangePermissions` in the domain allows the RUS to continue to have permissions to update properties on user objects that it needs to stamp new email addresses and other attributes.

Ready-to-go custom attributes

Even when the product included its own directory, Exchange has provided a set of custom attributes that you can use 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 of the attributes required by every company that might use the directory and that it's easier to provide a set of custom attributes than to go into the potential support nightmare that might occur if everyone attempted to extend the schema to add their own set.

To access the custom properties in Exchange 2007 and Exchange 2010, select an object in EMC, view its properties, and click Custom Properties. Figure 1-5 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 whether the mailbox is used by a permanent employee.

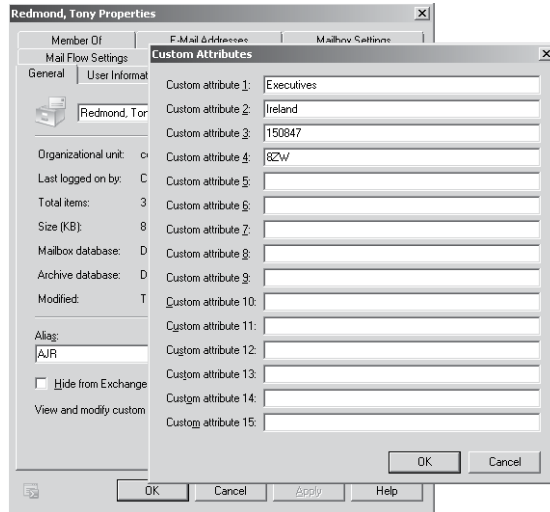


Figure 1-5 Custom attributes.

CAUTION

Be careful to ensure that the data that 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 '82W' -CustomAttribute12 'Temporary
Assignment'
Set-DistributionGroup -Identity 'Sales' -CustomAttribute10 'Synchronized 10-Dec-2010'
Set-DynamicDistributionGroup -Identity 'Texas Users' -CustomAttribute3 'Dallas'
Set-MailContact -Identity 'Ruth, Andy' -CustomAttribute1 'Lotus Notes User'
```

Based on operational experience, you can write values of up to at least 600 characters into the custom attributes.

Let's install

We now know the landscape for Exchange 2010, 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 actually install the software on a server, which is where we go in Chapter 2.

Installing Microsoft Exchange 2010

Approaching the installation	39	Security groups and accounts created by Exchange. . .	71
The services of Exchange	60	Contemplating management	74
Versions, roll-up updates, and service packs	63		

INSTALLING Exchange 2010 is not a particularly difficult operation, provided you do the necessary work to prepare. A test server can be installed and configured in well under an hour; production-class servers take longer because of the additional care that administrators must exert when they work in an environment where a mistake can impact the service delivered to users. Whatever the circumstances, the fundamental steps to install Exchange 2010 are very similar. We 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 a service pack or upgrade comes along.

Approaching the installation

If you have done the work to prepare the environment by installing the various prerequisites on the servers that will host Exchange (see Chapter 1, "Introducing Microsoft Exchange 2010"), the Exchange installation process is straightforward and painless. The steps that you'll take include the following:

- If you are deploying Exchange for the first time, you have to select a name for the organization. An organization name can be up to 64 characters and obviously cannot be blank. You can include spaces in the name; if this is the case and you 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 that it's best to choose a nonspecific name such as "Email" or "Exchange."
- Identify the target servers on which to install Exchange and the roles that each server will support.
- Decide on a name for each server. A good naming convention is one that conveys the purpose and use of a server without forcing the administrator to examine

server properties to discover its purpose. For example, Ex-HT-Dublin01 might be a good name if you know that “Ex” prefixes all Exchange servers, “HT” indicates a hub transport server, and “Dublin01” shows that the server is the first server installed in the Dublin Active Directory site or datacenter. On the other hand, the name “XYZ-Server-1234” poses a comprehension challenge for administrators unless they can learn all the server names by heart. Table 2-1 provides a set of codes that you could use in a server naming convention.

- Install prerequisite components such as Microsoft .NET Framework 3.5, Windows PowerShell 2.0, Windows Remote Management (WinRM), and Active Directory remote management tools on the servers on which you want to install Exchange 2010. Windows 2008 R2 servers include Windows PowerShell 2.0 and Windows Remote Management, but it is always a good idea to consult TechNet to validate the current list of prerequisite features. For this purpose, <http://www.microsoft.com/exchange/2010/en/us/system-requirements.aspx> is a good starting point. As discussed later in this chapter, Exchange 2010 SP1 is able to install prerequisite software when you install a server.
- Remove or upgrade any servers that run unsupported versions of Exchange, as these will stop the Exchange 2010 installation program. Exchange 2010 will not install if it detects an Exchange 2000 server or an Exchange 2003 server that is not running SP2 or later. Exchange 2007 servers must run SP2 or later.

Table 2-1 Codes for server naming

Server role designator	Server role
DC	Domain controller
GC	Global catalog server
MB	Mailbox server
HT	Hub transport server
ED	Edge transport server
CAS	Client Access server
ESV	Exchange multirole server
UM	Unified Messaging server

Unless it’s for a test environment, it is not a good idea to install Exchange 2010 on a domain controller and the Exchange setup program will issue a warning to this effect if it detects that it is running on a domain controller. The presence of Exchange 2010 on a domain controller has the side effect of elevating the privileges of domain administrators through the Exchange Trusted Subsystem. In effect, domain administrators will then have full read-write control over every object in Active Directory. This may be the situation anyway, but it’s not good to distribute permissions without reason.

Running /PrepareAD

The final step is to prepare Active Directory by installing the necessary schema extensions to support Exchange 2010. To do this, run the Setup program with the */PrepareAD* switch (Figure 2-1). Setup reads the instructions contained in an LDF file to extend the Active Directory schema and creates the security groups that the Exchange installation process will use to create the organization and install servers.

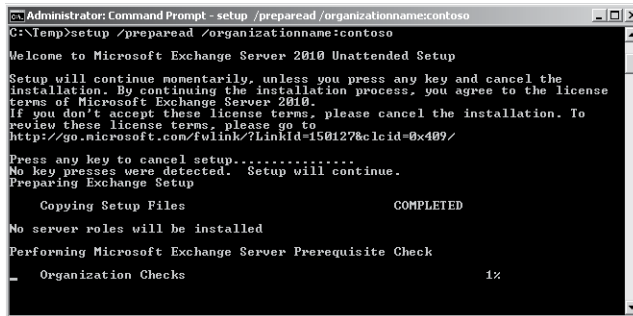


Figure 2-1 Running command-line setup to prepare Active Directory.

The Exchange installation kit contains a number of different sets of LDF files to cater for the various conditions that occur on a server. You can explore the LDF files in the Setup \ServerRoles\Common\Setup\Data and Setup\Data folders in the installation kit to see how Exchange extends the schema in different situations. For example, the set of files named PostWindows2003_schema0.ldf to PostWindows2003_schema99.ldf are used to update a forest where Windows 2003 has been installed but Exchange has never been deployed.

See the section “Preparing Active Directory for Exchange” in Chapter 1 for a more comprehensive discussion about the steps required to prepare Active Directory for Exchange 2010.

Exchange 2010 SP1 builds on the schema extensions implemented by Exchange 2010 and extends the schema further to support the introduction of new features such as the hierarchical address book and the group naming policy. You therefore need to include the schema extension as part of the planning for the deployment of SP1. Of course, if you plan to use Exchange 2010 SP1 for the initial deployment, you only need to extend the schema once as the SP1 version includes all of the previous extensions.

It’s not a disaster if you forget to run Setup /PrepareAD and then proceed to run Setup to install the first Exchange 2010 server in the organization. Setup will detect that the organization has not been prepared beforehand and is able to execute /PrepareAD as part of the steps it takes to prepare and then install the first Exchange server as long as you use an account that is part of the Schema Admins group to run Setup.

Figure 2-2 shows what happens when Setup detects that /PrepareAD has not been run. You can also see that Setup is warning that additional Exchange 2007 servers cannot be installed into the organization once Setup prepares Active Directory. This is because no Exchange 2007 server is present in the brand new organization. However, even if Setup is smart enough to run /PrepareAD for you, the nature of schema extensions and the requirement for the new schema to replicate throughout the forest means that it's still a much better idea to schedule this task separately well in advance of the deployment of the new organization. This approach allows you to ensure that the schema has replicated correctly and that Active Directory is truly ready for Exchange.

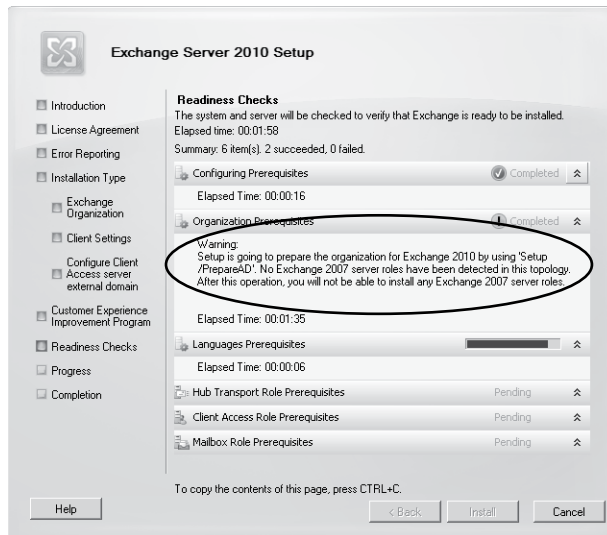


Figure 2-2 Setup runs /PrepareAD when installing the first Exchange server in an organization.

Installing prerequisite system components

A set of prerequisite software must be installed on a server before you can deploy Exchange 2010. Windows 2008 R2 servers are easier to prepare for Exchange 2010 because the .NET Framework 3.51, Windows Remote Management, and Windows PowerShell 2.0 are included in the operating system. To prepare Windows 2008 SP2 servers, you have to copy and install these components as the first step in the deployment process.

If the .NET Framework, Windows Remote Management, and Windows PowerShell 2.0 are installed on a server, both the GUI and command-line versions of the Exchange 2010 SP1 installation program offer the option to automatically configure all the remaining prerequisites thereafter. Figure 2-3 shows how the GUI version of Setup offers the option to install any required Windows roles and features. As you can see, you'll be prompted if any of the roles and features that are installed by Setup require a server reboot. (In particular, the

Windows Desktop Experience feature, which is required on Exchange Unified Messaging [UM] servers, will force a reboot.) In case 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 where you left off.

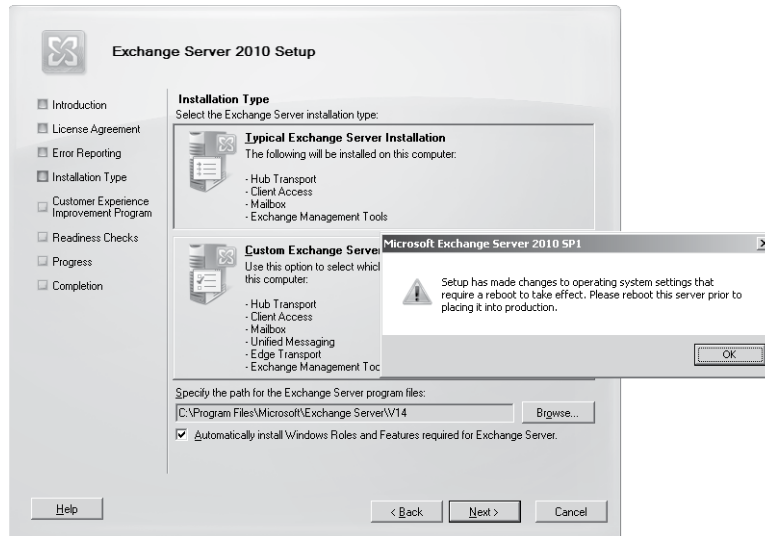


Figure 2-3 Exchange 2010 SP1 setup program offers to install required roles and features.

By comparison, to run `setup.com` to install a typical Exchange 2010 SP1 server including any required Windows server roles and features, type this command into a CMD session:

```
Setup.com /Mode:Install /roles:"C,H,T" /InstallWindowsComponents
```

The Exchange 2010 installation procedure requires that you configure the features manually. You can add all the necessary features by:

- Selecting and installing each feature with Windows Server Manager
- Running one of the installation configuration files provided with Exchange
- Installing each feature with the `Add-WindowsFeature` cmdlet in Windows PowerShell 2.0

Microsoft includes a set of Extensible Markup Language (XML) configuration files in the `\Scripts` directory under the Exchange installation directory that you can run to configure a server. These files can be used on both Windows 2008 R2 and SP2 servers, and the commands that they contain are directives for the `ServerManagerCmd` command that install the required Windows features. Although the `ServerManagerCmd` command is still present

in Windows 2008 R2, Microsoft has deprecated ServerManagerCmd, and the long-term direction is to use Windows PowerShell to configure software instead. However, the command works today.

To configure the required features with one of the provided XML files, run PowerShell as the Administrator and use the Import-Module ServerManager cmdlet to add the *ServerManager* module to the set of cmdlets available to the session. You can then select the most appropriate XML file from those provided in the \Scripts directory and input that to the ServerManagerCmd cmdlet. Table 2-2 lists the available files. For example, the command to configure a typical Exchange server (one that hosts the Client Access Server [CAS], hub transport, and mailbox server roles) is:

```
ServerManagerCmd -ip \Scripts\Exchange-Typical.xml
```

Table 2-2 XML configuration files to prepare Exchange 2010 servers

Server roles	Script
CAS, HT, MBX (typical server)	ServerManagerCmd -ip \Scripts\Exchange-Typical.xml
CAS	ServerManagerCmd -ip \Scripts\Exchange-CAS.xml
HT	ServerManagerCmd -ip \Scripts\Exchange-Hub.xml
MBX	ServerManagerCmd -ip \Scripts\Exchange-MBX.xml
UM	ServerManagerCmd -ip \Scripts\Exchange-UM.xml
Edge	ServerManagerCmd -ip \Scripts\Exchange-Edge.xml

ServerManagerCmd installs the features specified in the configuration file. Any features that are already configured will be ignored. Table 2-3 lists the features that are configured for each role. If you want, you can create your own scripts to automate the deployment process and use the Add-WindowsFeature cmdlet to add the different components for the server roles that you require. Some idea of the close relationship that exists between Exchange 2010 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 Control Panel application.

Table 2-3 Windows PowerShell commands to add Windows features

Server roles	Windows PowerShell command
CAS, HT, MBX	Add-WindowsFeature NET-Framework,RSAT-ADDS,Web-Server,Web-Basic-Auth,Web-Windows-Auth,Web-Metabase,Web-Net-Ext,Web-Lgcy-Mgmt-Console,WAS-Process-Model,RSAT-Web-Server,Web-ISAPI-Ext,Web-Digest-Auth,Web-Dyn-Compression,NET-HTTP-Activation,RPC-Over-HTTP-Proxy

Server roles	Windows PowerShell command
CAS	Add-WindowsFeature NET-Framework,RSAT-ADDS,Web-Server,Web-Basic-Auth,Web-Windows-Auth,Web-Metabase,Web-Net-Ext,Web-Lgcy-Mgmt-Console,WAS-Process-Model,RSAT-Web-Server,Web-ISAPI-Ext,Web-Digest-Auth,Web-Dyn-Compression,NET-HTTP-Activation,RPC-Over-HTTP-Proxy
HT or MBX	Add-WindowsFeature NET-Framework,RSAT-ADDS,Web-Server,Web-Basic-Auth,Web-Windows-Auth,Web-Metabase,Web-Net-Ext,Web-Lgcy-Mgmt-Console,WAS-Process-Model,RSAT-Web-Server
UM	Add-WindowsFeature NET-Framework,RSAT-ADDS,Web-Server,Web-Basic-Auth,Web-Windows-Auth,Web-Metabase,Web-Net-Ext,Web-Lgcy-Mgmt-Console,WAS-Process-Model,RSAT-Web-Server,Desktop-Experience
Edge	Add-WindowsFeature NET-Framework,RSAT-ADDS,ADLDS

Although it's easy to fire up a Windows PowerShell session, import the *ServerManager* module, and install the necessary components (Figure 2-4), there are several good Windows PowerShell scripts available on the Internet that can help you automate the installation of prerequisite components.

For example, some of the Exchange Most Valuable Professionals (MVPs) have collaborated to create the *Set-Exchange2010Prereqs.ps1* script that can be found and downloaded from a number of sites such as <http://www.ucblogs.net/files/folders/powershell/entry125.aspx>. The code in the script presents a menu to allow you to select various server configurations and then installs the necessary features. Its authors have done an excellent job of keeping the script updated to take account of new software such as Exchange roll-up releases and SP1. Of course, before you run any code downloaded from the Internet, you should run it on a test server to validate that the code doesn't do anything inappropriate and then make whatever changes are necessary to meet your requirements. For example, you might check that there's sufficient disk space available to install Exchange. To run the script, open a standard Windows PowerShell session and then set the script execution policy to be unrestricted, as otherwise Windows PowerShell will block the execution of any unsigned scripts. Remember to set the execution policy back to its original value after you are finished.

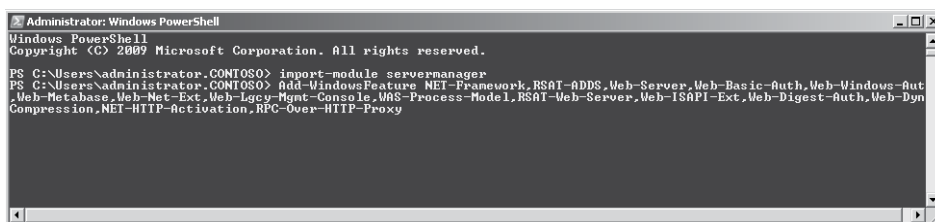


Figure 2-4 Installing Windows components with Windows PowerShell.

In addition to installing prerequisite software, before you run the installation program for a new Exchange CAS server, you have to update the properties of the .NET TCP Port Sharing service so that it starts automatically. CAS servers use the .NET TCP Port Sharing service to allow Exchange to multiplex communications through a single TCP port instead of requiring separate ports to be assigned to each service, specifically by the Mailbox Replication Service when it processes requests to move mailboxes or import and export data from mailboxes. The setup process checks that the .NET TCP Port Sharing service is started and its startup state is Automatic rather than the default (Manual). You can configure the service by executing this command from the CMD prompt:

```
Set-Service NetTcpPortSharing -StartupType Automatic
```

Installing the Microsoft Filter Pack

Before you can install the mailbox and hub transport roles, you must first install the Microsoft filter pack. Exchange uses the filters in the pack (referred to as IFilters) to be able to include attachments of various formats in its content indexes and to allow the hub transport service to apply transport rules that search attachments for specific words or phrases. The filter pack is available from the Microsoft Web site. If you deployed Exchange 2007 or the RTM version of Exchange 2010, you probably installed version 1.0 of the filter pack. Microsoft has updated the filter pack for Office 2010, and this version is the prerequisite for Exchange 2010 SP1. You can upgrade a server with the new filter pack and Exchange will update its content indexes as necessary.

The installation of the filter pack makes its filters available to Windows Search. After you install Exchange 2010, you need to take the steps described in <http://technet.microsoft.com/en-us/library/ee732397.aspx> to enable Exchange Search to use the filters. This should be done on all mailbox and hub transport servers. Because of the widespread use of Adobe's PDF format, you should also download and install the Adobe 64-bit PDF IFilter from <http://www.adobe.com/support/downloads/detail.jsp?ftpID=4025> to allow Exchange Search to include PDF documents in its content indexes. (Note that Adobe's filter has problems with some types of PDF files, so don't be surprised if your PDF searches don't find every possible match!) If you are unsure about how to enable these filters for Exchange Search, you can investigate Pat Richard's script at <http://www.ucblogs.net/files/folders/powershell/entry122.aspx>, which provides a very nice way to install the Microsoft Filter Pack, the Adobe PDF filter, or both.

Running Setup

Once Active Directory is prepared and you have installed the necessary prerequisite software and features, you can proceed to install Exchange. This can be done using the GUI version of the setup program, or you can use the command-line version.

Exchange 2010 setup supports four distinct modes:

- Installation of a new server, including the option to install a new role on an existing server.

Note

If you use a service pack to install Exchange on a new server, setup performs a complete installation from scratch. You do not have to apply the service pack separately after the base installation finishes.

You can install the language pack (see next section) if it is available or go ahead with just English, the default language. Make sure that you install your preferred language to manage Exchange on Windows before you install Exchange so that Exchange Management Console (EMC) runs in that language rather than English.

- Uninstall, including the option to remove a specific role from a server.
- B2B (build to build), the installation of a new build of Exchange on top of an existing server. This mode is used to apply service packs and roll-up 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 between 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.com /M:Upgrade`.
- Disaster recovery, which rebuilds a server based on its configuration data stored in Active Directory. It does not recover any of the mailbox or public folder databases that might have been present on the server: These have to be restored separately. The command-line switch for this mode is `Setup /m:RecoverServer`.

INSIDE OUT

Command-line vs. a GUI version setup

The command-line version is most often used to execute an unattended installation, but experienced administrators who have installed Exchange many times usually prefer the brevity 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. Another tip is to always perform installations (initial and subsequent upgrades) with elevated permissions, as this avoids running into any potential problems with Windows 2008 User Account Control (UAC).

As an example of using the command-line version of setup, this command installs a new CAS server:

```
Setup /Mode:install /Roles:ClientAccess
```

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 that Microsoft refers to as the “Can Opener” internally, perhaps because it opens all the possibilities that exist when Exchange is deployed correctly. Some might argue that it also opens up a can of worms, but that’s another day’s work! This version of the setup program depends on the .NET Framework and Windows PowerShell, which is the reason 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 of 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. You can get some idea of the amount of prerequisite checking that the setup program performs by browsing the contents of `ExBPA.PrePreqs.xml`. As the name suggests, setup leverages the framework used by the Exchange Best Practice Analyzer (ExBPA) to perform these checks. As mentioned earlier, the SP1 version of Setup will install any missing Windows server roles and features required by Exchange if you allow it to do so.

The GUI used by the setup program is adaptable in that it will ask for different information depending on the options that you choose. For example, if you install a CAS server that supports email connections from outside your network, you’ll be asked for the names of the external Uniform Resource Locators (URLs) that clients will use to access the service. If you install a hub transport server that will connect to Exchange 2003, you’ll be asked to nominate the Exchange 2003 bridgehead server that will link that infrastructure to Exchange 2010.

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 only install a single role on a server. All of the steps to deploy prerequisite software and to then 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 as per the recommendations published by the virtualization platform vendor.

Setup captures details of its progress in a watermark stored in the system registry to allow it to restart from the point of failure. The SP1 version also captures state information such as the options that you selected for the installation, meaning that you can restart Setup and won't have to input all your options again (Figure 2-5).

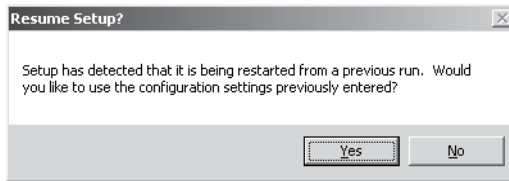


Figure 2-5 Setup offers to resume at the place it left off.

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).

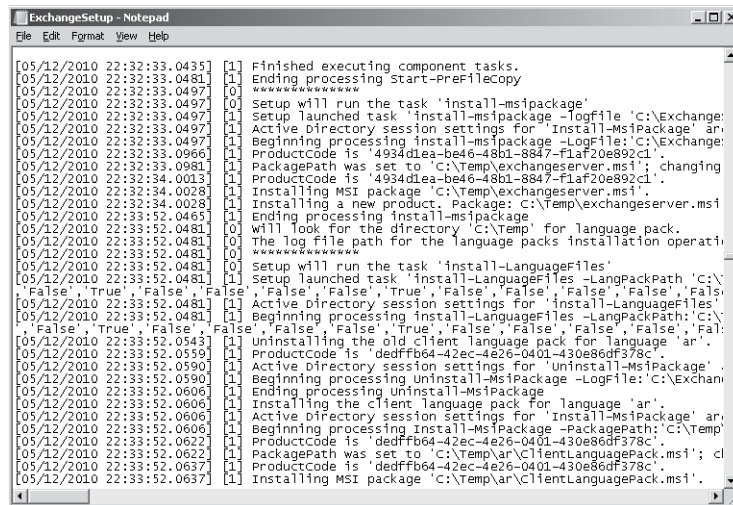
Following a successful installation, you can then apply the latest roll-up updates and any hot fixes that Microsoft has made available for Exchange 2010 to bring the server up to the latest software revision. Installing a roll-up update is straightforward (but should always be tested first) and should proceed smoothly as long as you run the update procedure with elevated permissions.

Setup logs

If any problems occur during the installation, you can normally find the reason why in the files that 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 of 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 (SP1 is a little less verbose as it removes some of

the messages output by the RTM version). The log is broken into major activities, which are indicated by a long line of asterisks, and minor activities, 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 2-6 reveals, there's a mass of data within the setup log. Most of the data captured in the logs are of no use unless you encounter a problem during an installation and need to find out why the problem occurred. Even so, most of the information that you'll need is right at the end of the log, where you'll see details of the problem experienced by the setup program and the actions that it took to unwind the installation. The remainder of the data in the log is interesting but is unlikely to be looked at by anyone except a Microsoft support specialist who's trying to track down an elusive problem.



```

[05/12/2010 22:32:33.0435] [1] Finished executing component tasks.
[05/12/2010 22:32:33.0481] [1] Ending processing Start-PreFileCopy
[05/12/2010 22:32:33.0497] [0] *****
[05/12/2010 22:32:33.0497] [0] Setup will run the task 'install-msipackage'
[05/12/2010 22:32:33.0497] [1] Setup launched task 'install-msipackage -logfile 'C:\Exchange
[05/12/2010 22:32:33.0497] [1] Active directory session settings for 'install-MsiPackage' ar
[05/12/2010 22:32:33.0497] [1] Beginning processing install-msipackage -logfile:'C:\Exchange
[05/12/2010 22:32:33.0966] [1] Productcode is '4934d1ea-be46-48b1-8847-f1af20e892c1'
[05/12/2010 22:32:33.0981] [1] PackagePath was set to 'C:\Temp\exchangeserver.ms1'; changing
[05/12/2010 22:32:34.0013] [1] Productcode is '4934d1ea-be46-48b1-8847-f1af20e892c1'.
[05/12/2010 22:32:34.0028] [1] Installing MSI package 'C:\Temp\exchangeserver.ms1'
[05/12/2010 22:32:34.0028] [1] Installing a new product. Package: C:\Temp\exchangeserver.ms1
[05/12/2010 22:33:52.0465] [1] Ending processing install-msipackage
[05/12/2010 22:33:52.0481] [0] will look for the directory 'C:\Temp' for language pack.
[05/12/2010 22:33:52.0481] [0] The log file path for the language packs installation operati
[05/12/2010 22:33:52.0481] [0] *****
[05/12/2010 22:33:52.0481] [1] Setup will run the task 'install-LanguageFiles'
[05/12/2010 22:33:52.0481] [1] Setup launched task 'install-LanguageFiles -LangPackPath 'C:\
[05/12/2010 22:33:52.0481] [1] Active directory session settings for 'install-LanguageFiles'
[05/12/2010 22:33:52.0481] [1] Beginning processing install-LanguageFiles -LangPackPath:'C:\
[05/12/2010 22:33:52.0543] [1] Uninstalling the old client language pack for language 'ar'.
[05/12/2010 22:33:52.0559] [1] Productcode is 'dedffb64-42ec-4e26-0401-430e86df378c'.
[05/12/2010 22:33:52.0590] [1] Active directory session settings for 'uninstall-MsiPackage'
[05/12/2010 22:33:52.0590] [1] Beginning processing uninstall-MsiPackage -logfile:'C:\Exchan
[05/12/2010 22:33:52.0606] [1] Ending processing Uninstall-MsiPackage
[05/12/2010 22:33:52.0606] [1] Installing the client language pack for language 'ar'.
[05/12/2010 22:33:52.0606] [1] Active directory session settings for 'install-MsiPackage' ar
[05/12/2010 22:33:52.0606] [1] Beginning processing install-MsiPackage -packagePath:'C:\Temp
[05/12/2010 22:33:52.0622] [1] Productcode is 'dedffb64-42ec-4e26-0401-430e86df378c'.
[05/12/2010 22:33:52.0622] [1] PackagePath was set to 'C:\Temp\ar\clientLanguagePack.ms1'; cl
[05/12/2010 22:33:52.0637] [1] Productcode is 'dedffb64-42ec-4e26-0401-430e86df378c'.
[05/12/2010 22:33:52.0637] [1] Installing MSI package 'C:\Temp\ar\clientLanguagePack.ms1'.

```

Figure 2-6 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 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 application event log to enable the support team to understand what happened and how to set about fixing any problems. They 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 minor event 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 play around with file locations when they want to run Exchange utilities such as Eseutil.exe.

Uninstalling Exchange

Of course, no one would ever remove Exchange from a computer unless it was a test box or it was time to decommission the server after you upgrade 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 the hub transport role from a server, the command `Setup.com /Mode:Uninstall /Roles:HT` will do the trick.
- Go to Control Panel, click Uninstall A Program, and then select Exchange 2010 from the list of programs. This invokes the GUI setup program in uninstall mode. You can then select the roles that you want to remove from the computer.

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 that 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 Database Availability Group [DAG]).
- 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 6, “Managing Mail-Enabled Recipients”).
- Dismount and remove databases.
- Remove the server from its DAG, if it is a member.
- Clear any outstanding move requests for databases hosted by the server.

- Move public folder replicas from a public folder database, if hosted by the server.
- Move the responsibility for Offline Address Book (OAB) generation and distribution, if 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 and move responsibilities such as OAB generation, 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 the servers to which you have transferred work to ensure that no problems are flagged. You could then take the server offline for another period to see if 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. As an added protection, Setup will check that all prerequisites are satisfied before it will proceed. Figure 2-7 shows what happens when Setup encounters issues that an administrator needs to resolve.

Usually, issues reported can be resolved quickly with the steps suggested by Setup.

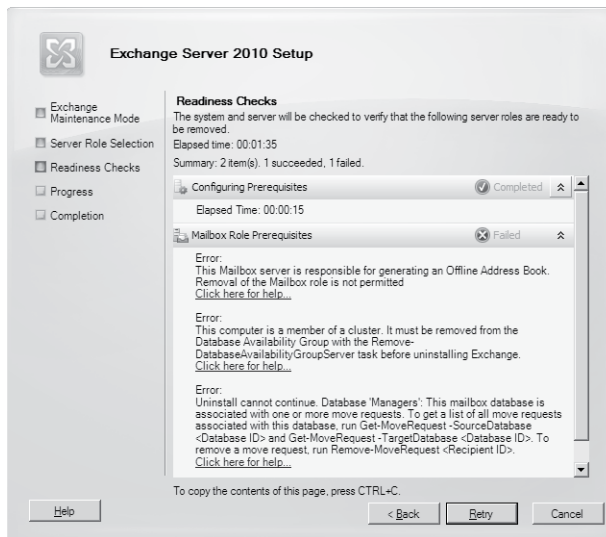


Figure 2-7 Setup reports errors during its prerequisite checks before a server can be uninstalled.

INSIDE OUT

Entering the dark zone

We enter Exchange's dark zone if the suggested resolution does not work. The dark zone is where a problem exists that cannot be resolved with an option available through EMC, 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 Exchange's 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 to be in when Setup won't proceed and you've tried all suggestions, including those found on the Internet.

It's at this point that some advocate using the ADSIEdit utility (see Chapter 1, "Introducing Exchange 2010"), 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. Of course, this is a radical step and certainly not something to attempt without a clear and comprehensive understanding of what you're about to do, but it is a method that has often been used to remove a stubborn server.

Perhaps the Exchange development group will help by providing a new Setup mode in the future, giving us a command something like this:

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

Repairing Exchange

The Setup program for Exchange 2010 does not support a repair mode. In other words, 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 you get into a situation where a server is not functioning correctly and nothing you do seems to make any difference (and there's no help to be gained by searching the Internet), you might be forced to deinstall all of the roles from the server to remove Exchange and then reinstall the server from the beginning. This is not so bad, as 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 which then only appear once you repair the apparent problem.

An alternative approach is to wait for Microsoft to provide a service pack or roll-up 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 which it detects as the server is upgraded.

Installing an edge server

An edge server is a specialized version of Exchange designed to reside in a perimeter network to handle incoming messages from the Internet. The server is not part of the Exchange organization, nor does it belong to the same Active Directory forest into which you install Exchange. However, an edge server can be installed in a separate Active Directory forest, perhaps one that you use to control all of the servers installed into the perimeter network. The isolation of the edge server from the other Exchange servers and Active Directory ensures that they remain protected even if the edge server is compromised by a hacker or other attack on security.

An edge server can only support that role. You cannot install the mailbox, CAS, or hub transport roles on the same server. Running setup proceeds in much the same way as a normal server except that you select the option for a custom server installation. Setup will only allow you to select the edge server role, and the management tools (EMC and Exchange Management Shell [EMS]) will be included automatically. Before installing Exchange you need to install the components that the setup program depends on, including the .NET Framework 3.51 and Windows PowerShell 2.0 (this is included in the operating system if you deploy the edge server with Windows 2008 R2).

You also need to install Active Directory Lightweight Directory Services (AD LDS) and the Remote Server Administrative Tools Active Directory Management Tools (RSAT-ADDS) before you install the edge role on a computer. AD LDS is a special version of Active Directory designed to provide directory storage and access for applications. It is a replacement for Active Directory Application Mode (ADAM) for Windows 2003 and used by Exchange 2007 edge servers. When you install the Exchange 2010 edge server, the setup program configures AD LDS to support Exchange. Somewhat confusingly, the service that manages the interaction between Exchange and AD LDS is still called Microsoft Exchange ADAM. The contents of the AD LDS directory are populated through a process called edge synchronization that we discuss in Chapter 14, "Message Hygiene."

Language packs

The first corporate email systems were determinedly monolingual. Systems such as IBM PROFS and Digital's ALL-IN-1 were splendid in U.S. English, but struggled to deal with anything else. The first email system to accommodate multiple languages was the ALL-IN-1 Basic European Edition (BEV), introduced by Digital in 1986 to support languages such as French, German, Hebrew, and Dutch. Great strides have been made to make software more

international in the quarter-century since based on the principle of separating language-dependent strings from the underlying code, even if disagreement occurs between applications about some of the more esoteric details.

Exchange and its clients have always supported multiple languages. Exchange 2010 uses a language-neutral model for the product. This approach is also used by Windows 2008 and Vista and means that Microsoft provides an installation kit for the base product and a separate language pack that contains all of the elements required for every supported language. The separation of the language pack from the product distribution allows Microsoft to update languages without impacting Exchange. They can add new languages or simply improve the quality of the translation to address local idioms or nomenclature identified by users. You can find the current list of languages supported by Exchange at <http://technet.microsoft.com/en-us/library/dd298152.aspx>.

The language pack can be installed during the setup of an Exchange server or afterward (you need to download the language pack from Microsoft's Web site first). You can install it with the GUI or using the */LanguagePack* parameter with the command-line setup program. The command-line program supports fresh installations, installations after initial server deployment, and language pack updates.

Tip

If you install Exchange without the language pack, the Exchange management tools and clients will only run in English, no matter what language is used to install Windows on the server. This is why it's a good idea to install Windows in the language with which you want to administer Exchange before you apply the language pack.

A separate language pack is available for UM. This language pack includes elements such as the prerecorded prompts used by UM, automatic speech recognition files, and support for voice preview and text-to-speech translation to allow content to be read to users in different languages.

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 first have to 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 in terms of its ability to support user load as the failed computer. Given that new hardware usually has improved performance, it shouldn't be a problem to replace the failed computer with new

hardware, but it might be problematic to replace it with older hardware if the configuration of that computer isn't as capable and is therefore less likely to be able to support the same load.

Once the new hardware is installed, you will need to do the following:

- Reset the Active Directory computer account for the failed server.
- Install and configure Windows with the prerequisite roles and features including any hot fixes required by Exchange; the Setup program does not have the ability to install any required software when it runs in recovery mode.
- Ensure that network connectivity works properly and that the new computer is configured with the correct IP addresses and other settings.
- Join the replacement computer to the domain hosting Exchange with the same name as the failed server.
- Authenticate the Windows server license.

You can then run Setup.com with the **/m:recoverserver** option. 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. Figure 2-8 shows a recovery operation in progress.

```

Administrator: Command Prompt - setup.com /m:recoverserver

Copying Setup Files                                COMPLETED

The following server roles will be recovered
Languages
Hub Transport Role
Client Access Role
Mailbox Role
Management Tools

Performing Microsoft Exchange Server Prerequisite Check
Configuring Prerequisites                          COMPLETED
Language Pack Checks                              COMPLETED
Hub Transport Role Checks                         COMPLETED
Client Access Role Checks                        COMPLETED
Mailbox Role Checks                             COMPLETED

Configuring Microsoft Exchange Server
Preparing Setup                                    COMPLETED
Stopping Services                                COMPLETED
Copying Exchange Files                           COMPLETED
Language Files                                    COMPLETED
Restoring Services                               COMPLETED
Languages                                         COMPLETED
Hub Transport Server Role                        COMPLETED
Client Access Role                               14%
  
```

Figure 2-8 Running Setup.com in recovery mode.

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. In other words, you cannot recover and upgrade at the same time; for instance, it's not possible to recover a failed Exchange 2010 server with an Exchange 2010 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 of the server roles defined in Active Directory will be operational on the replacement server. In the case of a failed mailbox server, you will still have to restore databases to their location on disk. The fully up-to-date database and transaction logs might be accessible from the disks used by the failed server. 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 nonmailbox servers, you might also have to restore configuration files that are not included in Active Directory. For example, if you update the transport configuration file for some reason, you will have to either:

- Copy the updated configuration file from another hub transport 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 first have to 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.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, will need to be restored separately.

INSIDE OUT

Delegated setup

Exchange 2010 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, this is done 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, and he 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
```

The physical server must exist and be properly prepared for Exchange to be installed. After a successful Setup run, you can check Active Directory, and you'll see that the new server is listed along with the other Exchange servers.

To complete server provisioning, the user's account must be a member of the Delegated Setup role group. We'll get to discuss Exchange's role-based access control system and the various role groups that are defined in later chapters. For now, it's sufficient to know that you have to run an EMS command to add the mailbox to which you want to delegate the server completion task. You can only delegate roles to mailboxes or universal security groups, so the user has to 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, we open EMS and type this command:

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

Afterward, this user will be able to run Setup on the provisioned server to complete the installation, provided of course that she holds local administrator permission for the server.

Customer Experience Improvement Program

The Exchange 2010 Setup program offers you the chance to join Microsoft's Customer Experience Improvement Program (CEIP). This is a program that gathers data from operational Exchange servers to allow Microsoft to understand how Exchange is used in production. Microsoft then uses the data to decide how to develop Exchange over the next few releases. In short, this ensures that their engineering effort is business-driven and focused in areas that deliver a real customer impact rather than being based on interesting technology challenges. The kind of questions that Microsoft hopes will be answered by the CEIP data includes the following:

- How many users turn on Short Message Service (SMS) notification for their mailboxes?
- How many non-delivery reports (NDRs) are generated by the transport system, and for what reason?

- How much latency exists between the time a user sends an SMS and the time the mobile device picks up the SMS message from Outbox?
- What are the most frequently observed cmdlet errors?
- What Exchange server roles are most often installed in virtualized (VMware, Hyper-V) environments?

Enabling a server to provide CEIP data requires a deliberate opt in. You can opt in by selecting the option when you run the GUI version of Setup to install an Exchange server. Alternatively, you can enable CEIP reporting by doing either of the following:

- Going to the Microsoft Exchange On-Premise node in EMC and accessing the Customer Feedback page (Figure 2-9).
- Running two EMS commands to enable CEIP reporting for the organization and then for the servers that you'd like to provide data. For example, these commands enable the organization to provide CEIP data from server ExServer1. The *-Industry* parameter is optional and is used to categorize the results by industry. If set, it must be one of the values defined by Microsoft such as Manufacturing, Transportation, Finance, or Agriculture. If you are unsure, enter Unspecified:

```
Set-OrganizationConfig -CustomerFeedbackEnabled $True -Industry "NonProfit"
```

```
Set-ExchangeServer -Identity ExServer1 -CustomerFeedbackEnabled $True
```

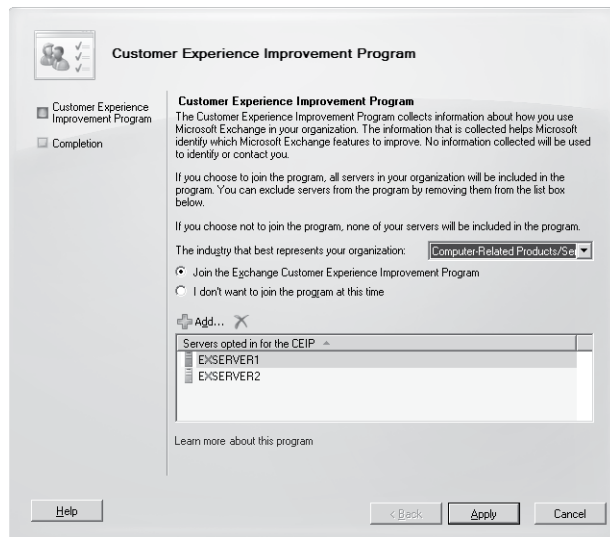


Figure 2-9 Enabling CEIP for the organization.

The effect of enabling CEIP is to create a registry value called `HKEY_Local_Machine\Software\Microsoft\SQMClient\Windows\CEIPEnable` and set it to 1 (one). The presence of this registry value instructs a Windows component called Software Quality Metrics (SQM) to record and then upload CEIP data on a regular basis. SQM gathers information from applications that register for CEIP, such as Exchange, on a per-session basis and stores these data in memory to a threshold of approximately 15 KB. According to the SQM engineers, 15 KB is sufficient to hold roughly 4,000 datapoints gathered about how an application functions. When the threshold is reached or the session terminates, SQM writes the data to a file on disk and uploads the file to Microsoft the next time that a new session is initialized. This approach is taken to ensure that an application shutdown is not delayed by the need to transmit data to Microsoft.

It's natural to worry that data about operational environments might be reused for purposes that you don't approve. In this case, the CEIP data gathered by Microsoft is governed by the privacy guidelines described at <http://www.microsoft.com/products/ceip/en-us/privacypolicy.msp>. Microsoft does not store information that can be identifiable to a specific customer. However, before you enable CEIP for Exchange, it's a good idea to verify that your company's privacy and security teams review the Microsoft privacy guidelines and support the transmission of CEIP data from your organization to Microsoft.

The services of Exchange

Once upon a time, email servers were engineered and delivered as a single monolithic process. The Information Store process captures the imagination as the single most obvious embodiment of Exchange with its seeming ability to occupy huge amounts of memory, but in reality, the Exchange code base is spread across a series of interconnecting services that collectively deliver all of the product's functionality. Not all services run on all servers, as some are only required for specific activities. For example, you don't need to run the Internet Messaging Access Protocol 4 (IMAP4) or Post Office Protocol 3 (POP3) services unless you need to support clients that connect to Exchange using these protocols. Some services have dependencies on other Exchange services or base Windows services (such as the Net TCP Port Sharing service).

Table 2-4 lists the details of the sets of services that you can expect to see on a multirole server together with their internal names and any dependency that exists. You can see that the majority of Exchange services depend on the Active Directory Topology service. This is not surprising because Exchange stores so much information about its own configuration in Active Directory and makes use of other data such as users and groups. If this service isn't running, then Exchange doesn't know about the Active Directory site topology that exists or the domain controllers that are available, so it can't fetch the data necessary to start up services such as Transport, nor does it know what user accounts are mail-enabled, what database holds their mailboxes, or even what their email addresses are.

Table 2-4 Exchange 2010 services

Service Name	Use	Dependencies	New in Exchange 2010
Active Directory Topology (MSExchangeADTopology)	Computes the current Active Directory site topology and makes these data available to other services		
Address Book (MSExchangeAB)	Generates the OAB and other address book views	AD Topology	
Anti-spam update (MSExchangeAnti-spamUpdate)	Applies updates to Microsoft Forefront anti-spam services	AD Topology	
Edgesync (MSExchangeEdgeSync)	Synchronizes information between the Exchange organization and standalone edge servers	AD Topology	
File Distribution (MSExchangeFDS)	Distributes files such as OAB updates to servers within the organization	AD Topology Workstation	
Forms-based Authentication (MSExchangeFBA)	Manages forms-based authentication process executed on CAS servers to allow access to OWA and ECP		
IMAP4 (MSExchangeIMAP4)	Provides access to mailboxes for IMAP4 clients	AD Topology	
Information Store (MSExchangeIS)	Manages the internal operations (including memory management) for mailbox and public folder databases	RPC Server Event Log Workstation	
Mail Submission (MSExchangeMailSubmission)	Submits messages from the Information Store to the Transport service	AD Topology	
Mailbox Assistants (MSExchangeMailboxAssistants)	Performs background maintenance operations on mailboxes such as auctioning calendar requests and automatic message archiving	AD Topology	
Mailbox Replication (MSExchangeMailboxReplication)	Replicates mailbox contents to target servers during mailbox moves performed by the Mailbox Replication Service (MRS)	AD Topology Net .TCP Port Sharing	Yes
Monitoring (MSExchangeMonitoring)	Allows applications to call the Exchange diagnostic commands		
POP3 (MSExchangePop3)	Provides access to mailboxes for POP3 clients	AD Topology	
Protected Service Host (MSExchangeProtectedServiceHost)	Provides a secure host for internal Exchange services that need to be protected from other services	AD Topology	Yes
Replication (MSExchangeRepl)	Manages the replication of transaction logs to target mailbox servers within a DAG	AD Topology	

Service Name	Use	Dependencies	New in Exchange 2010
RPC Client Access (MSExchangeRPC)	Manages incoming RPC client (Outlook) connection requests and directs clients to server holding target mailbox	AD Topology	Yes
Search Indexer (MSExchangeSearch)	Generates content indexes of data held in mailbox and public folder databases	AD Topology Microsoft Search (Exchange)	
Server Extension for Windows Backup (WsbExchange)	Permits Windows 2008 Backup utility to take online backups of Exchange databases		
Service Host (MSExchangeServiceHost)	Provides an internal host for Exchange services; for example, this service generates group metrics data for mail tips	AD Topology	
System Attendant (MSExchangeSA)	Performs background monitoring and maintenance functions such as the application of email address policies to objects; some of the functions served by the System Attendant process in previous versions of Exchange have moved to the Service Host process	RPC Server Event Log Workstation	
Throttling (MSExchangeThrottling)	Provides throttling function to limit the ability of user actions to swamp system resources	AD Topology	Yes
Transport (MSExchangeTransport)	Manages Transport system to move messages between Exchange servers and connectors	AD Topology	
Transport Log Search (MSExchangeTransportLogSearch)	Executes search requests through message tracking logs on mailbox and hub transport servers	AD Topology	
Microsoft Search (Exchange) (MSFTESQL-Exchange)	A version of the SQL-based search process tailored to produce content indexes from Exchange mailbox and public folder data	RPC	
Microsoft Exchange ADAM (ADAM_MSExchange) (only on edge servers)	Controls the interaction between an Exchange 2010 edge server and the AD LDS instance that stores configuration and other data used to process incoming messages	COM+	
Microsoft Exchange Credential (MSExchangeEdgeCredential)	Manages the shared credentials required to synchronize data between an edge server and hub transport servers in a connected Active Directory site	ADAM	

Note

On a side note, as Windows PowerShell is a big part of the future for Windows management, you can examine details of a service through Windows PowerShell. For example, to see what service dependencies exist for the Information Store service, you can use a command like this:

```
Get-service MExchangeIS | Select Name, ServicesDependedOn|Format-List
```

The output is:

```
Name                : MExchangeIS
ServicesDependedOn  : {RPCSS, EventLog, LanmanServer, LanmanWorkstation}
```

Other services run on servers that host the UM role, and others are present if you deploy Microsoft Forefront antivirus and anti-spam services (at additional cost). Of course, if you choose to deploy third-party technology for functions such as anti-spam, monitoring, backup, or archiving, those services will be present and have to be managed.

Versions, roll-up 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 brand new version of the product. The current approach divides updates into two different kinds of releases that you can plan for:

- **Service packs (SP):** Microsoft releases a new service pack for Exchange on average once a year, so it came as no surprise to see Exchange 2010 SP1 appear in June 2010. A service pack typically includes new functionality as well as fixes. It is common to find that the first service pack completes functionality that could not be finished in time to allow the product to meet its planned release date. This factor, plus a feeling that no one really wants to live on the bleeding edge of new technology, is usually cited as the reason why companies delay upgrades until Microsoft has released the first service pack for a new product.
- **Roll-up updates (RU):** Roll-up updates appear roughly every three months and are cumulative in that RU3 includes all of the fixes issued in RU1 and RU2. Therefore, you can apply the latest RU to update a product with all of the current released fixes. Roll-up fixes are issued on the Web and different versions are available for each version of Exchange (including service packs) that Microsoft supports, so you have to be sure that you download and install the right RU for the version of Exchange that you use. Some hot fixes that pertain to specific situations might also be available separately,

and it's worth checking with Microsoft whether any important hot fixes exist for functionality that are important to your deployment before you plan to deploy a roll-up update.

Testing roll-up updates

Make sure that you test roll-up updates before you deploy into production as there have been cases for both Exchange 2007 and Exchange 2010 where problems occurred during or after the installation of the update. For example, Microsoft published the first roll-up update for Exchange 2010 on December 9, 2009. After deploying RU1, some customers reported that their users couldn't access the OWA logon page because of a JavaScript syntax error. The problem was easily fixed by removing and recreating the OWA virtual directory, but it serves to demonstrate that your servers might present a different environment to those used by Microsoft to test new releases, so predeployment testing is an absolute necessity. It's also important to install roll-up updates with accounts that have full administrator permissions to ensure that all files are properly updated.

It is sometimes difficult to understand the difference between a roll-up release, a service pack, and a new version of a Microsoft server product. My view is that a roll-up release helps you keep servers running the latest possible code so that you run a lower risk of encountering a known bug. Of course, roll-up releases have been known to introduce their own problems in the past, and the regular cadence of releases means that you have yet another task to fit into a maintenance schedule. Nevertheless, Microsoft's approach of delivering a flow of fixes through roll-up releases has worked well for Exchange 2007 as it allows administrators to plan scheduled software maintenance on a consistent basis. The same approach applies for Exchange 2010, and I anticipate that it will be equally valuable.

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 to allow servers to 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 used by Exchange. 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.

A service pack is a more definitive point in a product's development cycle. It includes fixes and will bring software to a patch level to at least the same degree as the most recent roll-up release, and it usually adds some value in terms of improvements to product features. The improvements might be in the form of completing functionality that couldn't be finished or were simply too buggy before Microsoft shipped the previous version of the software or simply refining features in the light of customer feedback. For example, the release to manufacturing (RTM) version of Exchange 2007 didn't include Standby Cluster Replication because the work required to ensure that log replication would work across two datacenters was incomplete when the cut-off came to build the RTM software, so it appeared in Exchange 2007 SP1. Similarly, ActiveSync made its appearance in Exchange 2003 SP2. Microsoft used to have a rule that no new feature could be added to a service pack that caused a user interface change. Thankfully, this rule has long been ignored, as all it did was introduce badly wanted features that had no administrative interface except registry changes or command-line utilities. Microsoft introduced a lot of new functionality in Exchange 2010 and made many changes in SP1 to improve the functionality of features such as personal archives, move requests, and retention policies.

A new version of Exchange can be a more refined release of an existing architecture as the result of a couple of additional years in development, or it can be a brand new architecture that seeks to meet different customer challenges. Exchange 2003 was a development of Exchange 2000. Exchange 2007 was a brand new architecture. Some will argue that Exchange 2010 is a refinement and expansion of the architecture laid down in Exchange 2007, but I consider that the amount of change to the Store in terms of the new schema, the requirement to move to Windows 2008, and the introduction of native high availability features within the product mark Exchange 2010 as a new architecture. This means that the deployment of Exchange 2010 requires the same degree of attention and planning as the move from Exchange 2003 to Exchange 2007 (or even from Exchange 5.5 to Exchange 2000).

Exchange 2010 Service Pack 1

It's fair to say that the first release of Exchange 2010 was feature-rich, but had some rough edges that required further work to bring it up to the standard required to convince large enterprises that the software was ready to deploy. As we will see throughout the remainder of this book, Exchange 2010 introduced fundamental change in areas such as authorization and control (role-based access control), compliance (personal archives, retention tags and policies, and discovery searches), high availability (DAGs), and a brand new management application (Exchange Control Panel). Unsurprisingly, although these features worked, the engineers could do far more to round out the product once they were given more time to refine and smooth the functionality, which is exactly what happened in Exchange 2010 SP1.

Table 2-5 lists the most important areas of improvement in Exchange 2010 SP1, together with the chapter in which the improvements are discussed in some detail.

Table 2-5 Areas of improvement in Exchange 2010 SP1

Improvement in Exchange 2010 SP1	Chapter
Installation of required Windows features by Setup (Windows 2008 R2 servers only)	1
More comprehensive user interface provided in ECP to manage RBAC roles, role groups, and assignments	4
Increased range of management options available in ECP	5
Store driver fault isolation	7
New mailbox repair requests	7
Granular transaction replication for DAGs	8
New scripts to help manage DAGs	8
Upgrades to the Public Folder management console	9
Outlook Web App user interface improvements and new features	12
Mailbox import and export requests	14
Hierarchical address book	14
Flexibility to place personal archive in a different database than the primary mailbox	17
User interface in EMC to define retention tags and policies	17
Discovery searches initial scans and deduplication of results	17

Version numbers

Exchange's 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 14.00.0639.021 means:

- Product version 14: Exchange 2010 is a component of the "Office 14" wave of products. Exchange 2007 was version 12 and there is no version 13.
- Service pack 00: No service packs have been applied, so you know that this is the original version of the product.
- Major build 639: Microsoft builds new versions of the product on an almost daily basis and the build number increments sequentially, so this is build 639, the build used for the RTM version of Exchange 2010.
- Minor build 021: Twenty-one minor updates have been applied to major build 639. The number of minor updates is accounted by the way that Microsoft tweaked the last major build of Exchange 2010 with a series of minor fixes as they drove to complete the product.

You can find the version information in the “Help About Exchange Server 2010” option of the EMC. Another way is to open the Server Configuration node of EMC where 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"
"EXCH-SVR1", "Enterprise", "Version 14.0 (Build 639.21)", "Mailbox, ClientAccess, HubTransport"
"EXCH-DUBLIN", "Enterprise", "Version 14.0 (Build 639.21)", "Mailbox, ClientAccess, HubTransport"
```

Version 639.21 is the RTM release of Exchange 2010. The development team released this version to manufacturing on October 8, 2009. The first production customer deployments were already in progress before this time as part of Microsoft’s Technology Adoption Program. General deployments in the wider customer base started in November 2009. Exchange 2010 SP1 is version 218.15 and was released to manufacturing on Build 218.15 and August 23, 2010.

Exchange also writes information about version numbers into the system registry as shown in the following example. You can retrieve this information with some Windows PowerShell code (see Figure 2-10 Installation information in the system registry) to identify the major and minor builds that are installed on a server. In this instance, MsiBuildMajor is 639 and MsiBuildMinor is 21, so this server runs version 639.21 or the RTM release of Exchange 2010. The following example shows Windows PowerShell code you can use to fetch the Exchange version information from the system registry.

```
$RegExSetup = 'Software\\Microsoft\\ExchangeServer\\v14\\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)) -as [String]
$BuildMinor = ($RegKey.GetValue($V2)) -as [String]
$ExVersion = $BuildMajor + ";" + $BuildMinor
Write-Host $Server "runs version" $ExVersion
```

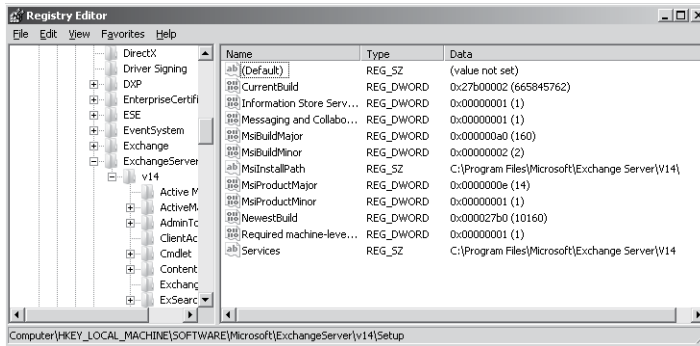


Figure 2-10 Installation information in the system registry.

Exchange also stores information in the system registry about roll-up 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 patch) 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 EMS 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
0.10 (14.0.100.0)
```

We can see that version 14 is listed, so we know that this object should be managed through Exchange 2010 management tools. We can also see that the minor version is 100 and we know that the minor build number for the RTM release is 639, so any build of Exchange 2010 can manage mailbox objects. Mailboxes certainly exist in previous versions of Exchange, and the fact that the version number specifies that Exchange 2010 must be used indicates that mailbox objects have been upgraded for Exchange 2010. We know this

is true because new attributes have been added to mailbox objects for features such as archive mailboxes. If you look at the version numbers for other objects that are introduced in Exchange 2010, like management roles, you'll see even higher version numbers such as 14.0.451.0. However, because the minor build number (451) is under the RTM build number, these objects can be managed by any release of Exchange 2010. By comparison, if we look at a send connector and do the same thing, we 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 like Exchange 14 became Exchange 2010). 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.

An excellent article on the Exchange development team blog (<http://msexchange team.com/archive/2010/01/08/453722.aspx>) gives guidance as to what version of objects can be managed by the different versions of EMC and EMS.

Reporting licenses

Just like Exchange 2007, EMC also reports any unlicensed servers in the organization (Figure 2-11) that it detects when it starts up and reads in the Exchange configuration data. Microsoft regards these servers as being in a trial status. 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.

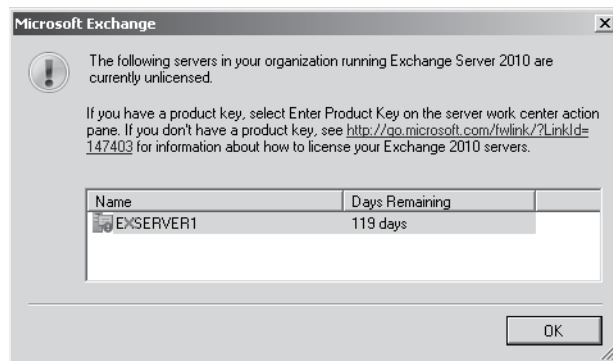
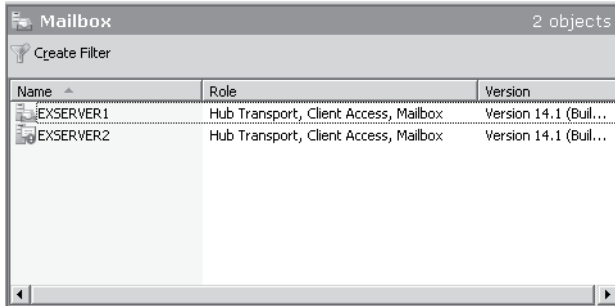


Figure 2-11 EMC reports an unlicensed Exchange server.

You can view the current licensing situation by clicking the Server Configuration node of EMC to display a list of servers in the organization. EMC uses different icons to mark licensed and unlicensed servers.

At first glance the icon's purpose is not immediately obvious, but there is one licensed and one unlicensed server in the set of servers shown in Figure 2-12. The second server is the unlicensed variety.



Name	Role	Version
EXSERVER1	Hub Transport, Client Access, Mailbox	Version 14.1 (Bull...)
EXSERVER2	Hub Transport, Client Access, Mailbox	Version 14.1 (Bull...)

Figure 2-12 Viewing license status for servers.

To resolve this problem, select the unlicensed server, and then select the Enter Product Key option in the action pane. This invokes the wizard that accepts a valid product key and generates a product identifier for the server. The change will not be active until the next time that the Information Store service is restarted. The alternative is to use the shell with a command like this:

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

Tip

Exchange does not terminate abruptly once the warning period elapses. All of its functionality continues as before, and you can even upgrade a server that has exceeded its trial period to Exchange 2010 SP1. However, you have entered the twilight zone where you might be guilty of running unlicensed software, and Exchange will continue to nag you until you enter the required license. This is not a great situation to be in as 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.

Security groups and accounts created by Exchange

As discussed in Chapter 1, Exchange depends on Active Directory and fully exploits the extendable schema to add a set of objects and properties that enable the Exchange components to work. Alongside the schema changes, the Exchange installation procedure creates a number of security groups and mail-enabled accounts that Exchange uses for different purposes. Table 2-6 lists the security groups that Exchange creates in Active Directory and their purposes. 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 2-6, many are used to hold the security principals required to enable users to perform the tasks assigned to role groups defined in Exchange's new role-based access control mechanism (see Chapter 5, "Exchange Management Console and Control Panel").

Table 2-6 Exchange USGs created in Active Direct

Group	Purpose	RBAC
Delegated Setup	Administrative role group for accounts who complete the setup for an Exchange server after it has been provisioned by an organization administrator.	Yes
Discovery Management	Administrative role group for accounts who perform discovery searches and retrieval of discovered information from user accounts.	Yes
Exchange All Hosted Organizations	Global group. This group holds mailboxes hosted on off-premise servers and is used to apply password setting objects on those mailboxes.	
Exchange Servers	Every Exchange 2007 and Exchange 2010 server in the organization is a member of this group and is used to allow the servers to mutually authenticate against each other.	
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 SP2, you'll find that the Exchange Trusted Subsystem group is created in the Microsoft Exchange Security Groups OU to support interoperability between Exchange 2007 and Exchange 2010.	
Exchange Windows Permissions	A privileged group used by Exchange to manipulate Windows permissions and Active Directory objects.	
Exchange LegacyInterOp	A group used by Exchange to perform privileged operations with legacy servers in the same organization.	
Hygiene Management	Administrative role group to manage tasks to cleanse the email stream such as anti-spam and antivirus.	Yes

Group	Purpose	RBAC
Organization Management	Administrative role group for accounts that have permission to manage objects at an organization level such as create new connectors or transport rules. Members have full control over any object in the Microsoft 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). However, although these accounts can mail-enable a public folder, they cannot change mail-enabled attributes for a folder (such as an email address), as 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 2-6, 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. It is used by the setup program to ensure that it can complete even if Active Directory replication has not yet populated details of the USG in the local domain. Clearly it is best to wait for replication to finish before you attempt to install a server.

As shown in Figure 2-13, 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 this 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 from Exchange 2010

SP1 onward. See Chapter 15, “Compliance,” for more information on how to configure the auditing of administrator actions.

- *SystemMailbox{1f05a927-xxxx-xxxx-xxxx-xxxxxxxxxxxx}*, where x is a randomly assigned number: Used for message moderation, to store 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 between 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 these items can be retrieved by administrators. Exchange creates the default discovery mailbox in 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 or region.

CAUTION!

Apart from accessing the discovery mailbox to access the results of discovery searches (see Chapter 15) you should not attempt to log onto 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.

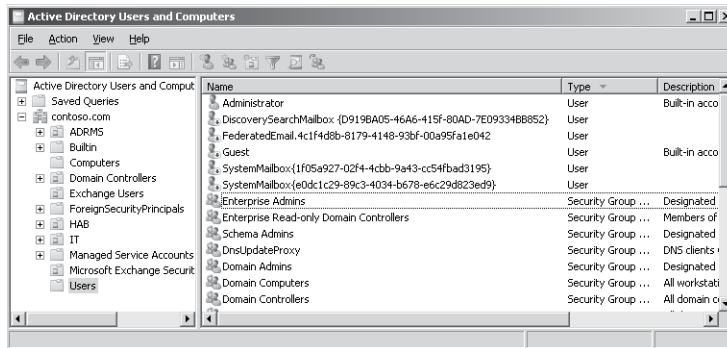


Figure 2-13 Mail-enabled Active Directory accounts created by Exchange 2010.

Two scenarios and 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 owned by Exchange. Normally you won't need to touch the Exchange Trusted Subsystem group, but there are two known circumstances when you might need to add it explicitly to an object. The first is when you configure the file share witness for a DAG to be on a server that does not have Exchange 2010 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. See Chapter 11, "Client Access Server," for more information on this topic.

The second situation is where you want to manage Exchange 2007 servers using an Exchange 2010 management console or EMS. Exchange 2007 servers have no knowledge of the Exchange Trusted Subsystem so a management operation attempted from Exchange 2010 will probably result in an "access denied" error. The solution is once again to add Exchange Trusted Subsystem to the local Administrators group on the Exchange 2007 server.

Contemplating management

Now that we've installed Exchange 2010 after making sure that our Active Directory was ready to support its deployment, we are ready to contemplate the challenge of understanding the management of Exchange. Both Exchange 2007 and Exchange 2010 are built on the foundation of Windows PowerShell, so that's the next logical step in our odyssey.

The Exchange Management Shell

How Exchange leverages Windows PowerShell.....	76	Some useful EMS snippets	129
Remote PowerShell	79	Verbose PowerShell	136
EMS basics	93	Setting language values	136
Command line versus Integrated Scripting		Execution policies.....	137
Environment	122	Testing cmdlets.....	139
Active Directory for PowerShell.....	126	But we need some control	146

MICROSOFT Windows PowerShell is an extensible automation engine consisting of a command-line shell and a scripting language. Originally code-named Monad, Windows PowerShell 1.0 debuted in 2006. Microsoft 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, as evident by the many code snippets for Exchange administration that are distributed on the Internet. The role of Windows PowerShell continues to expand across Microsoft products, and it now extends itself into Microsoft's newest 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 fully 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 together 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 stand-alone application. Many scripts are available on different Internet sites to assist with Exchange management.

How Exchange leverages 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 that process one or more Exchange objects to complex scripts that perform tasks such as mailbox provisioning. Most administrators cut their teeth with PowerShell by using the Exchange Management Shell (EMS) to do simple things, like 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, control ActiveSync, update Active Directory with user safe lists, 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—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, there was no way for administrators to 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 that Windows PowerShell now plays in the Exchange architecture and how it provides a central place to encapsulate business logic that underpins the Exchange setup program, the Exchange Management Console (EMC), the Exchange Control Panel (ECP), and the EMS.

Exchange's use of Windows PowerShell to implement functionality presented by the graphical user interface (GUI) of EMC and the setup program is probably the most extensive of any Microsoft application. As explored throughout this book, the options presented by EMC 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. It's also worth emphasizing that 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 them through role-based access control (RBAC). RBAC is a huge shift in the way that Exchange manages and grants permissions to users that is designed to function across a range of different environments, from a single-server organization to an organization composed of a mixture of on-premise and hosted servers. The need to accommodate such a wide range of environments is also the reason

Microsoft has moved from local PowerShell (where all commands are executed on a local server) to remote PowerShell (where commands are redirected through Microsoft Internet Information Services [IIS] for execution on a target server). We'll get to the details of just how remote PowerShell and RBAC work together in the Exchange Server 2010 version of EMS shortly.

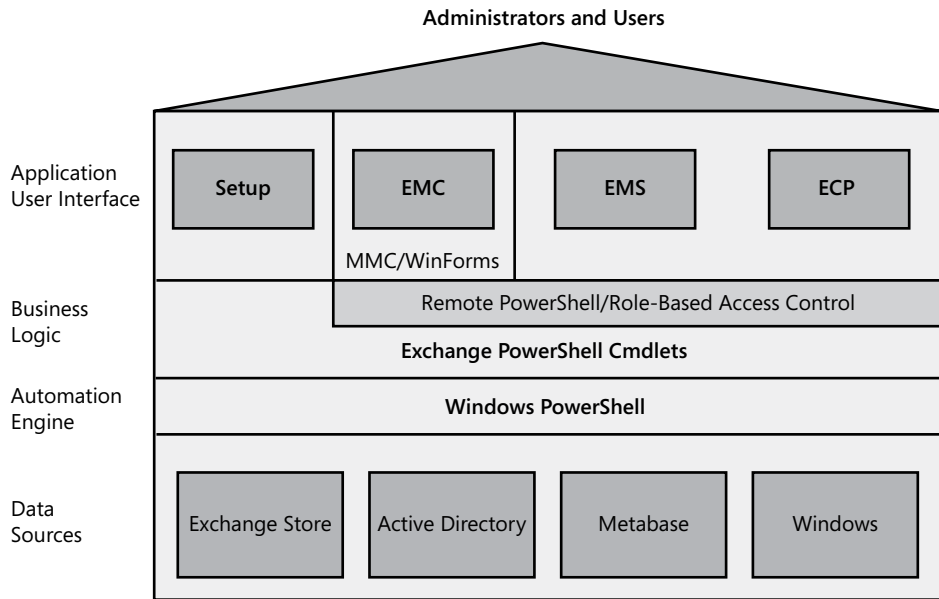


Figure 3-1 Windows PowerShell at the heart of Exchange.

Simplifying the implementation of new functionality

The critical point is that the four major 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 allows the Exchange engineers to concentrate on implementing new functionality rather than reimplementing features specifically for use by EMC, EMS, or the setup program. This approach allows 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.

The RTM release of Exchange 2010 includes 584 cmdlets that are added by EMS to join the standard set of Windows PowerShell cmdlets, including cmdlets to work with the system registry, the file system, variables (including environmental variables), and

so on. The number of cmdlets grows again to 619 in Exchange 2010 SP1. For example, the Export-Mailbox and Import-Mailbox cmdlets that are used to export and import mailbox data from Outlook personal storage (PST) files are replaced by the New-MailboxExportRequest and New-MailboxImportRequest cmdlets and other associated cmdlets used to control the requests.

For more information about these cmdlets, refer to Chapter 12, “Mailbox Support Services.”

Collectively, the set of EMS cmdlets manages objects and the properties of those objects that form Exchange. Objects include mailboxes, servers, transport rules, connectors, and so on. You can determine the exact number of cmdlets owned by Exchange with the following command:

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

INSIDE OUT

Finding the cmdlets available to you

Of course, as you will learn when we discuss RBAC in Chapter 4, “Role-Based Access Control,” an EMS session in Exchange 2010 only allows you access 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 lesser permissioned role group, such as Help Desk or Recipient Management. You can use this command to generate a full list of all of the Exchange 2010 cmdlets that your account can access:

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

By comparison, Exchange 2007 includes 394 cmdlets, 26 of which are removed in Exchange 2010 (largely because of the demise of storage groups). The 216 new cmdlets provided in Exchange 2010 reflect the new functionality in the product, such as the introduction of the RBAC model, mailbox archives, and the Database Availability Group (DAG), along with the expansion of existing functionality such as messaging records management.

Windows PowerShell has been at the heart of Exchange since Exchange 2007, and its use and syntax are fundamental skills for administrators to master. In fact, many of the more hardcore Exchange administrators prefer EMS to EMC 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 Windows PowerShell found in other chapters. To begin, let’s review the biggest change that Microsoft has made to EMS in Exchange 2010: the transition from a purely local implementation to remote PowerShell and its associated technology.

Once we understand how to connect to EMS, we'll go on to review how to use cmdlets to get work done.

Remote PowerShell

Exchange Server 2007 was the first major Microsoft server product to embrace Windows PowerShell extensively. Windows PowerShell has a relatively short history, and its 1.0 release exhibited some of the flaws that you'd expect. Inconsistencies in syntax are easily overcome, but the lack of remote capability was more serious because it required the installation of Windows PowerShell and its snap-in (set of cmdlets) for Exchange on any workstation or server from which you wanted to perform management tasks. In reality, this shortcoming is often overlooked because Exchange administrators are accustomed to having to install software before they can operate. Installing software is an acceptable requirement in environments where all the servers are under your control and within your own network, but it causes problems when you want to manage servers remotely. The Microsoft introduction of online services where companies will be able to run their Exchange environments inside large Microsoft datacenters means that remote management has taken on new importance. Exchange 2010 includes many new features designed to ease the transition to online services, and remote PowerShell provides the fundamental building block for management. The combination of remote PowerShell with RBAC allows administrators to manage objects residing on a server in a remote datacenter as easily as you can manage 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 if 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, they are retrieved across the network and processed locally.

Exchange 2010 extends the concept of remote management to 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 used for all EMS sessions in Exchange 2010. 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 to the local server to do the work. The same applies for EMC, 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 replaces local PowerShell in Exchange 2010 for all server roles except Edge servers. The sole exception is for commands used during setup, which continue to execute locally. The removal of local PowerShell and the concentration on the combination of remote PowerShell and RBAC as the basis for administrative control over Exchange components is a major change in the product.

The implementation of remote PowerShell for Exchange 2010 separates business logic into code that runs on the client and code that runs on the Exchange server. It is based on common Windows components such as Windows PowerShell 2.0 and the WS Management model (WS-Man and Windows Remote Management, WinRM). The discussion at <http://blogs.msdn.com/powershell/archive/2009/01/06/manage-winrm-settings-with-wsman-provider.aspx> provides useful background about how Windows PowerShell remoting is built on top of WinRM. Collectively, the Windows components combine to provide an effective ability to perform Exchange management operations remotely.

The logic for replacing local PowerShell with the remote model is simple. Just like the change in Exchange 2007 to force all messages to flow through the transport system so that a single common place existed to apply features such as transport rules, remote PowerShell forces Exchange administration to flow through RBAC so that a PowerShell session will only ever include the cmdlets necessary to do the job. The logic for keeping local PowerShell on Edge servers is simple, too. These servers are isolated from the rest of the organization so they do not have access to the RBAC roles. Anyone who logs onto an Edge server as an administrator operates in a management context of just that server and has complete control over that server. However, he cannot affect any other server or object in the Exchange organization.

The need to support hosting platforms such as Microsoft Business Productivity Online Services (BPOS) was a major influence on Exchange's move to remote PowerShell. Providing a secure and controllable mechanism that permits administrators to execute privileged commands to control the subset of objects that 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. Exchange 2007 controls access to its management cmdlets through access control lists (ACLs) that are linked to Active Directory accounts. If the ACLs on your account mark you as an Exchange administrator, you can use Windows PowerShell to manage Exchange; if not, you can't. However, in an online services environment where many different companies share the same multitenant server and storage infrastructure, it is highly unlikely that you will share the same Active Directory. Trust relationships and directory synchronization provide answers that have been used in hosting environments, but these solutions often require a good deal of effort to set up and maintain. Microsoft takes a new approach to the management of permissions in Exchange 2010 with the introduction of RBAC. The concept is not new and has been used in previous authentication or identity management systems on UNIX and other platforms. What's

different here is its application to provide Exchange with a method to extend RBAC so that roles apply remotely.

CAUTION!

Until Microsoft removes the functionality, it is possible to use local PowerShell with the Exchange 2010 snap-in to perform management operations on a server. However, Microsoft is not testing local PowerShell with Exchange anymore, and it is entirely possible that problems will appear in local PowerShell that will never be resolved. 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.

Flowing remotely

To understand how remote PowerShell and RBAC work together, let's 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 allows her to create new mailboxes and update the properties of existing mailboxes. We also assume that the user's account is enabled to use remote PowerShell. In many cases, people in specialist roles such as help desk personnel will use the EMC or the ECP to perform tasks, but an experienced Exchange administrator might prefer to use a command-line interface because of its power and flexibility when compared to either EMC or ECP.

Figure 3-2 lays out the various components used by remote PowerShell 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 2010, 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 used to read data from Active Directory. For the purpose of this discussion we'll assume that the account used 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 -IdentityAccountName -RemotePowerShellEnabled $True
```

All PowerShell sessions flow through IIS because even a local connection goes through *localhost*. All Exchange 2010 servers support IIS and the PowerShell virtual directory, or *vdир*, and all are members of the Exchange Trusted Subsystem security group and therefore can manipulate any object in the organization.

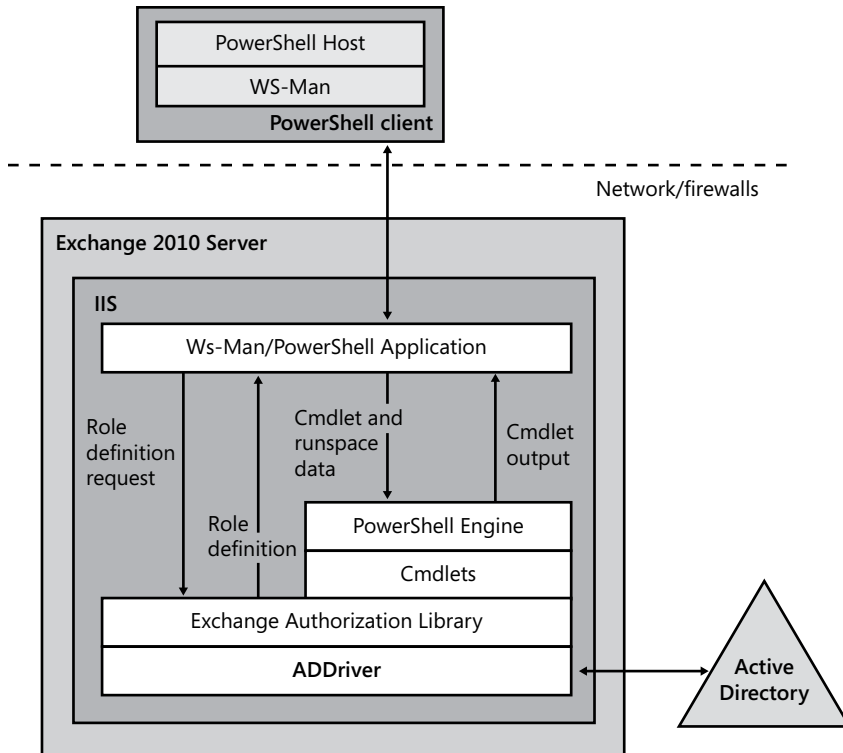


Figure 3-2 Remote PowerShell architecture.

If you run EMS on a workstation or server that has the Exchange management components 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 will have to specify the name of the server with which you want to work. This is done with the `New-PSSession` cmdlet, passing the name of the server to which to connect in the form `https://fqdn/PowerShell/`. 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.

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 for some reason (later we discuss some of the issues that might interfere with a connection), 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 via the Exchange Authorization Library (a new component in Exchange 2010). 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 (something referred to as DC affinity). This is different from the behavior that exists in Exchange 2007 because all PowerShell sessions on a server run in the same process, so you cannot have a static setting because it might be inappropriate for some sessions. In Exchange 2007, each PowerShell session functions in its own process and you have complete control over the Active Directory settings. Many PowerShell cmdlets support the *-DomainController* parameter to allow you to connect to a specific domain controller (specifying the fully qualified domain name [FQDN]) should the need arise.

Things are a little more complicated in a hosted environment where you are connected to your local network but need to work with Exchange data in a forest maintained by the hosting provider. In this instance, a directory synchronization process occurs to synchronize the basic account information from the local Active Directory forest and the data held in the Active Directory forest managed by the hosting provider.

A role group defines the set of administrative actions that a user is allowed to perform inside Exchange and can be resolved into a set of PowerShell cmdlets that the user is allowed to use within her PowerShell session. Because our user works with mailboxes as defined by the Recipient Management role group, the set of cmdlets that she can use includes commands with easily identified purposes such as New-Mailbox, Set-Mailbox, Get-Mailbox, and so on. Unlike the situation with Exchange 2007, where the complete set of cmdlets is available after you load the Exchange snap-in to a PowerShell session, RBAC ensures that the user can execute only the cmdlets required to perform her role. If you are an Exchange administrator who holds the Organization Management role, you'll have access to almost the full set of cmdlets (to gain access to the full set, you have to grant your account access to some minor roles). However, if your account has been assigned only the roles necessary to be able to work with recipients, you'll see just the cmdlets covered by the roles assigned to you.

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, EMC, or ECP 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 like setting properties on messaging connectors or one-off operations such as creating a new DAG. Instead, RBAC makes sure that users can only access the cmdlets that they need to get their job done.

Connecting to remote PowerShell

When you run EMS on a server that has the Exchange 2010 management components installed, the EMS initialization script creates an environment that is roughly equivalent to an Exchange 2007 EMS session. The code in the RemoteExchange.ps1 script attempts to create a remote session with the local host. If successful, it then proceeds to identify your account to Exchange and uses RBAC to determine the cmdlet set that you are allowed to use, and so on.

There's no obvious evidence given to a user that his role has forced RBAC to restrict the cmdlet set or parameters that he can use with cmdlets, because the initialization of a session progresses just like it would for a fully privileged user. However, once you start to execute cmdlets, you quickly realize that you can't do as much as you'd like to. 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 even use tab completion (we discuss this in a little while) to reveal a restricted property. On the other hand, you will be able to use the Set-Mailbox cmdlet to update properties that are generally exposed for user update through ECP, so (assuming *JSmith* is the alias for your mailbox) you'll be able to do things like 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 the folders and the items that each contains. All of this is controlled by RBAC, the roles that your account holds, and the scope for the roles in terms of the cmdlets and parameters defined in the role. From this discussion, you should now understand how critical RBAC is to remote PowerShell and by extension to every aspect of the Exchange 2010 management toolset.

If EMS can't connect to the local host, it attempts to connect to a server in the local Active Directory site. If no server in the local site responds, EMS attempts to connect to an Exchange server in a remote Active Directory site selected at random. The problem here is that EMS could select an Exchange server in the worst possible site at the end of an extended network connection. However, if you're in this situation, you have other problems to solve. It is not a sign of good Exchange server health if a server in the local site doesn't respond to a connection request from EMS.

The Exchange help file contains details of some of the errors that can occur during initialization of a remote PowerShell session, including the following:

- Attempting to connect without using Secure Sockets Layer (SSL; http rather than https).
- Specifying an incorrect virtual directory name (see the section "A more complex environment to manage" later in this chapter).
- Attempting to connect to an Exchange server that doesn't exist (or is offline).
- Various network errors.
- Trying to connect with an account that is not enabled for remote PowerShell (the administrator account is automatically enabled, but you might have to enable other accounts specifically with the `Set-User` cmdlet). This information is stored in the *protocolsetting* attribute of the user's account. For example, this command enables Windows PowerShell for the account Redmond:


```
Set-User -Identity Redmond-RemotePowerShellEnabled $True
```
- The Windows PowerShell initialization script can't be executed because of the server's execution policy.

Assuming all is well, you should be able to proceed. When connected, you should then be able to work in much the same manner as when you connect to Windows PowerShell remotely with EMS in Exchange Server 2007, subject to the limitation that the role group to which you belong might restrict the set of available cmdlets. The time required to set up a new session is longer than in Exchange 2007 because of the way that the session is set up, but once EMS is fully initialized it should be very familiar to anyone who has worked with EMS in Exchange 2007. To get some idea of how much work is done to create a new EMS session, use a text editor to review the code in the `RemoteExchange.ps1` and `ConnectFunctions.ps1` scripts in the Exchange binaries folder.

Be careful where you execute

One interesting aspect of connecting to a server to instantiate a Remote PowerShell session is that sometimes you end up executing commands on a server that you don't expect. For example, if you start EMS on a server in a site and EMS cannot connect to the local server for some reason, you might end up being connected to another server without noticing that this has happened. The EMS initialization script puts the server name in the title bar of the window for the EMS session so the server name to which you are connected is pretty obvious. However, it's amazing how many people don't notice this small but important fact.

As explained earlier, being connected to one server rather than another isn't usually an issue when high-performance links connect the servers. However, if the command generates a log, it will be created on the server to which you're connected, which might cause you to scratch your head if you go looking for a log on the local server. For example, if you recover a mailbox from a recovery database (refer to Chapter 9, "Backups and Restores," for more information on this topic), the log file is created in the `\Logging\MigrationLogs` directory on the server where the command executes. It can be quite frustrating to search for a log on a local server when Exchange has quite correctly created it elsewhere! If you need to connect to a specific server, you can run the `Connect-ExchangeServer` cmdlet after you initialize a session.

A more complex environment to manage

There's no doubt that EMS operates within a far more complex environment in Exchange 2010 than it did in Exchange 2007. Apart from the extra cmdlets to know and master, and some changes in syntax for older cmdlets such as the addition of new parameters and the deprecation of a small number of cmdlets, the way that Microsoft has implemented remote PowerShell creates a number of dependencies that must be present before EMS functions correctly. The major dependencies are as follows:

- Active Directory
- Windows PowerShell

- IIS 7.0
- Windows Remote Management (WinRM)
- Exchange RBAC

Figure 3-3 illustrates one of the symptoms that you might see when a remote PowerShell session aborts due to one of the most common problems reported to Microsoft. Essentially, software versions must be correct; WinRM, IIS, and the authorization and other settings on several virtual directories have to be configured correctly; and all components must be able to connect together. To make the point, it's worthwhile to discuss some common issues and their underlying causes. Apart from an inability to contact a domain controller to authorize a session, there are three known causes for this symptom:

1. The KerbAuth module (used by remote PowerShell to authenticate user connections) is not registered as a native module for the PowerShell virtual directory.
2. The WsMan module (used for Windows Remote Management functions) is not registered correctly for PowerShell because its entry is missing from the Application Host configuration file.
3. The user attempting to connect is not enabled for remote PowerShell. This is rarely the cause because most administrators run EMS when they are logged onto a mail-enabled account that is also enabled for PowerShell. To check that this is the case, you can use the `Get-User` cmdlet to validate that the account's *RemotePowerShellEnabled* property is set to `$True`.

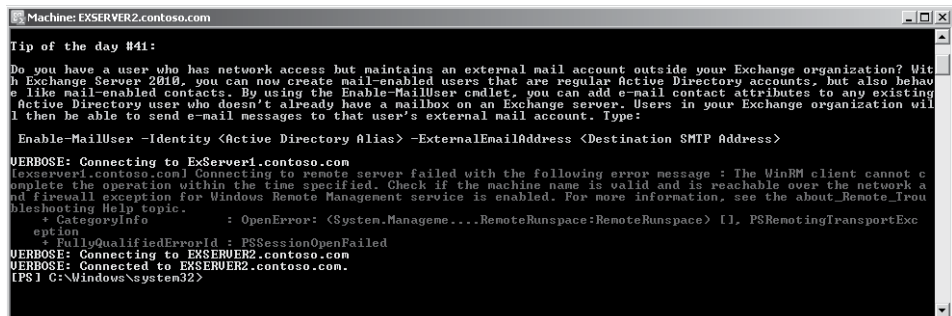


Figure 3-3 EMS fails to connect to a server.

You can check for the underlying problems in the first two cases by using the IIS Manager to view the modules registered for the PowerShell virtual directory. In Figure 3-4, we can see that the KerbAuth module is registered as a Native type enabled directly for the directory (entry type is Local). These are the correct values. The code points to the KerbAuth.DLL module in the Exchange binary directory, which is also correct.

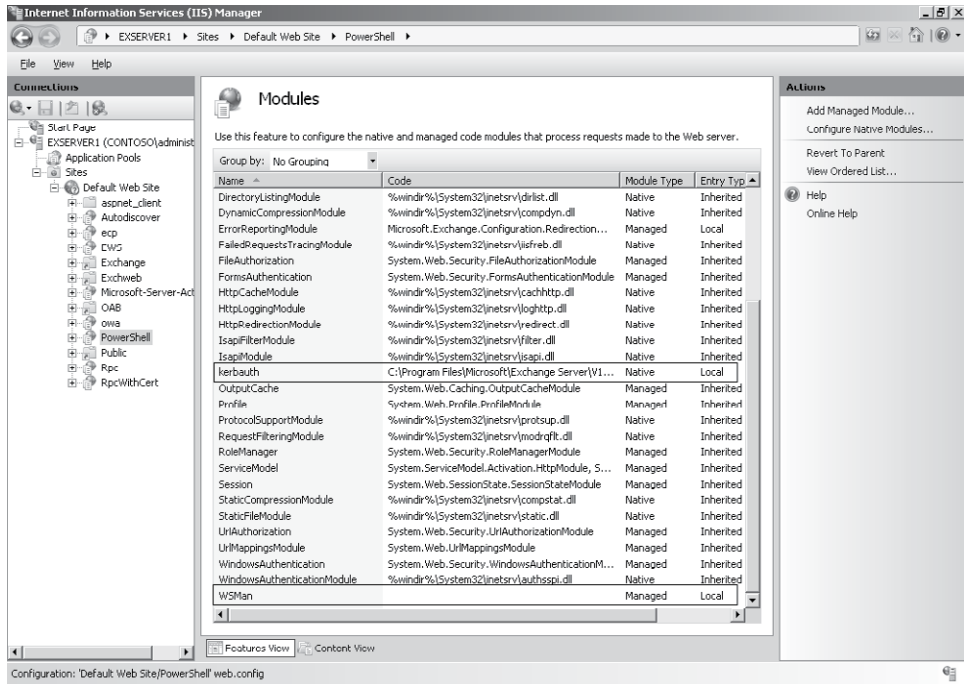


Figure 3-4 Detecting problems with IIS modules registered for PowerShell.

Often the problem lies with the WSMAN module at the bottom of the list of modules. First, there's no value for "code," so IIS can't load anything. Second, the module type is set to Managed, whereas it should be Native. To fix this problem, we have to update the %Windir%\System32\inetrv\Config\ApplicationHost.config file and register the WSMAN module in the global modules section. Open the file with a text editor and look for the section:

```
<globalModules>
```

Then add a new line in the set of global modules:

```
<add name="WSMan" image="C:\Windows\system32\wsmvc.dll" />
```

You don't have to restart IIS to make the change effective because EMS will use the updated information about the WSMAN module the next time it initializes a session.

INSIDE OUT

Accounts need to be mail-enabled

In previous versions of Exchange you didn't need a mail-enabled account (an Active Directory account that is linked to an Exchange mailbox) to run EMS or EMC as long as the account was sufficiently permissioned to access the Exchange objects. RBAC relies on mail-enabled accounts to manage its role definitions, which is why the account that you use to install the first mailbox server in an Exchange 2010 organization is automatically mail-enabled. Any attempt to use an account that is not mail-enabled or has something wrong with its mailbox (for example, it was on a server that has subsequently been removed from the organization but lingering traces remain in Active Directory) will cause RBAC to fail and you won't be able to use EMS or EMC. This issue is addressed in Exchange 2010 SP1; this version allows accounts that are not mail-enabled to use ECP to perform administrative actions. However, accounts that are not mail-enabled still cannot run EMS or EMC. Chapter 5, "Exchange Management Console and Control Panel," contains more information about how to grant ECP access to an account that is not mail-enabled.

TROUBLESHOOTING

Can't connect to the remote host because WS-Management service is not running

If you encounter a problem where EMS reports that the connection to the remote host was refused because the WS-Management service is not running, the cause could be either that the service is indeed not started or it is not configured to support remote access requests via HTTP. You can run the WinRM utility to check and update the server's configuration with the necessary settings to allow these requests. To do this, run Windows command shell (CMD) as the Administrator and then issue the WinRM QuickConfig command. If missing, this command will create the listener processes on which WinRM depends.

EMC issues the Discover-ExchangeServer command during its initialization to begin the process of connecting to a Client Access Server (CAS) server and reading information about the organization to display in the console. Figure 3-5 illustrates a common problem caused by the account being used to run EMC not having sufficient Exchange permissions to be able to execute the task. The same problem causes the error message "Starting a command

on remote server failed with the following error message: access is denied” when EMS initializes and no Exchange cmdlets are available in the EMS session.

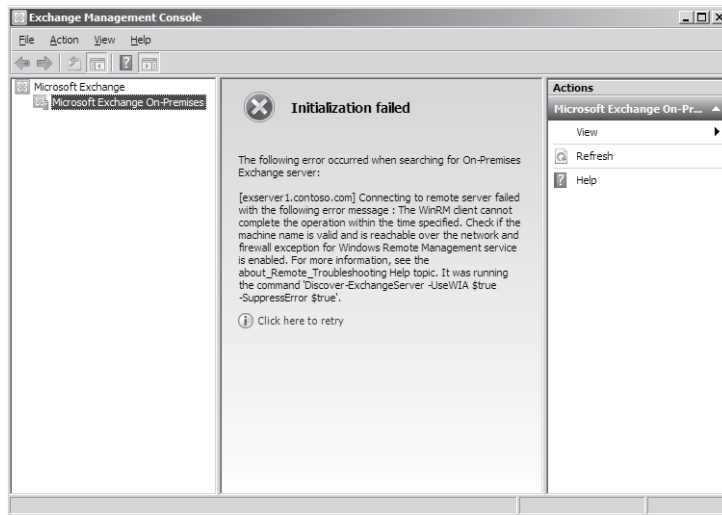


Figure 3-5 A problem with permissions causes EMC to fail.

RBAC dictates what the account with which you have logged in is able to do with Exchange. If you see an error like this, it's a good indication that something's wrong with permissions. The fix might be as simple as logging out and reusing a properly permissioned account (for example, one that holds the Organization Management role), or it could be a more sinister cause such as a build-to-build upgrade that has failed to update the default RBAC role groups properly. The solution is to rerun the part of the installation process to update the role groups, but this should only be done after you establish that this is indeed the root cause. For example, if you can't run EMC or EMS with the account that you used to perform a build-to-build upgrade following an apparent successful completion, it's a good indicator that something's gone wrong with the role groups and they need to be reapplied.

Among the less common problems reported by administrators are the following:

- The WinRM IIS extension (a server feature) is missing for some reason. The usual solution is to install the WinRM IIS feature with Server Manager and reboot the server.
- WinRM is unable to load the Exchange authorization dynamic-link library (DLL) because the environment variable pointing to the Exchange installation path (the directory where the DLL is located) is missing. The solution is to add the *ExchangeInstallPath* variable (and make sure that it points to the right place).

- The computer clock on a server is not fully synchronized with domain controllers. Among other things, the EMC initialization process validates that the server to which you connect is a member of the Exchange Servers universal security group. If the system clock on the Exchange server differs from the domain time by more than five minutes, the Kerberos ticket presented by the computer to an Active Directory domain controller will be invalid. The lack of clock synchronization will prevent any access to Active Directory so the check fails because the computer won't be able to read the group membership. To avoid this problem, make sure that the clocks on all Exchange servers are synchronized with the domain.

All of this goes to prove that the interaction among Exchange 2010, IIS, WinRM, and Windows PowerShell collectively forms a complex mechanism where everything has to function properly before administrators can do their work.

Advantages of remote PowerShell

Microsoft doesn't make changes to products like Exchange without seeing some clear advantage. In this case, three potential advantages are available. First, separating client and server processing and using IIS as the mechanism to route remote traffic allows communication through a well-understood route that can flow across firewalls and so accommodate both Microsoft Online Services and enterprise customers that operate multiple Active Directory forests.

Second, creating the ability for administrators to delegate tasks to other users on the basis of well-defined roles adds to overall system security and prevents inadvertent mistakes that can affect essential data. For example, an administrator who can amend mailbox details might not be able to add new mailboxes or affect other parts of the messaging infrastructure such as deleting the subscription that connects an Exchange Edge server to the rest of the organization.

Third, remote capability means that you can perform Exchange management tasks from a workstation without installing the Exchange management tools, meaning that you should be able to manage Exchange from just about any workstation that's connected to your network. Of course, you still need to install Windows PowerShell and any prerequisite software such as the latest version of the .NET Framework, but these components are more likely to be part of a corporate build for an administrator workstation (or integrated into the base operating system) than Exchange-specific code.

INSIDE OUT

A disadvantage is a slower startup

One disadvantage of remote PowerShell is that it is slower to start up than local PowerShell. On a cold server (one where the IIS worker process has not been started for Windows PowerShell), it can take 20 to 30 seconds before EMS is fully initialized. You can measure the startup performance of EMS on your server by feeding its initialization script to the Measure-Command cmdlet. Achieving the fastest server in terms of EMS startup became quite a challenge in terms of gaining bragging rights for many of the participants in Microsoft's Exchange 2010 Technology Adoption Program (TAP). The fastest startup reported was 1.54 seconds.

```
Measure-Command {.'C:\Program Files\Microsoft\Exchange
Server\V14\bin\RemoteExchange.ps1'; Connect-ExchangeServer -auto}
```

The output:

```
Days           : 0
Hours          : 0
Minutes       : 0
Seconds       : 4
Milliseconds  : 553
Ticks         : 45537444
TotalDays     : 5.2705375E-05
TotalHours    : 0.001264929
TotalMinutes  : 0.07589574
TotalSeconds  : 4.5537444
TotalMilliseconds : 4553.7444
```

Another way of measuring the performance of remote PowerShell is with the Test-PowerShellConnectivity cmdlet. In this example run the New-TestCASConnectivity.ps1 script from the \Scripts directory (see Figure 3-6) to create a synthetic test account to be used for the connection and then execute the test cmdlet. The script creates the test user account in the Users organizational unit (OU) of Active Directory. If you have multiple OUs beginning with "Users," the script will fail to create the mailbox. You can fix the problem by editing the script and updating the variable that holds the OU name.

```
.\New-TestCASConnectivityUser.ps1
Test-PowerShellConnectivity -ClientAccessServer ExServer1 -VirtualDirectoryName
'PowerShell (Default Web Site)' -TrustAnySSLCertificate
```

Once a server has been running for a while, starting a new EMS session is much faster because IIS is warmed up. You can measure the difference by using Measure-Command again. However, even on the fastest server with a fully warmed IIS, remote PowerShell will

never match the immediacy of a local PowerShell session simply because of the extra layers that have to be connected to make everything work.

```

Machine: ExServer1.contoso.com
[PS] C:\program files\microsoft\exchange server\v14\scripts>.New-TestCASConnectivityUser.ps1
Please enter a temporary secure password for creating test users. For security purposes, the password will be changed r
egularly and automatically by the system.
Enter password: *****
Create test user on: EXSERVER1.contoso.com
Control-Break to quit or Enter to continue:
UserPrincipalName: extest_aa6371e2b9e54@contoso.com
WARNING: The command completed successfully but no settings of 'contoso.com/Users/extest_aa6371e2b9e54' have been
modified.
You can un-enable the test user by running this command with the following optional parameters : [-UMDialPlan <dialplann
ame> -UMExtension <numDigitsInDialplan>] .Either None or Both must be present.
[PS] C:\program files\microsoft\exchange server\v14\scripts>_

```

Figure 3-6 Running the New-TestCASConnectivityUser script.

INSIDE OUT

Help isn't as extensive in remote PowerShell

Another disadvantage is that remote PowerShell does not support the same kind of help that local PowerShell does. In EMS in Exchange 2007, you can use wildcards to locate help for a group of cmdlets, as in `Get-Help *Exchange`. This won't work in Exchange 2010 on Microsoft Windows 2008 SP2 or R2 because of some limitations in remote PowerShell, but the feature reappears when you deploy Windows 2008 R2 SP1.

With all of these caveats in mind, let's explore what happens when a new EMS session starts to understand why things might be a tad slower.

EMS basics

There are more than 600 cmdlets in the set available to Exchange 2010 SP1, 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're probably not going to revisit the `Set-ReceiveConnector` cmdlet very often once the connector is working. On the other hand, there are cmdlets such as `Get-Mailbox` that you'll use daily. Some examples (in no particular order) of frequently used Exchange cmdlets include 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

- **New-MoveRequest** Set up a request to move a mailbox to another database
- **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
- **Get-TransportConfig** Return properties that control how the transport service processes messages
- **Get-ExBlog** Launches a browser to bring you to the Exchange development team's blog

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 to 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. As discussed previously, unlike Exchange 2007, you cannot use wildcards with Get-Help to retrieve information about a set of similar cmdlets.

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 cannot, therefore, 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. In Exchange 2007, when you look for help, EMS displays help for the complete set of cmdlets available to it. In Exchange 2010, RBAC controls limit help content so that you see help only for the set of cmdlets available to the roles that a user holds.

One side effect of this change is that wildcard searches using the Get-Help cmdlet are no longer possible to look for help in the set of Exchange cmdlets. For example, you can't use a command like Get-Help *Exchange* to retrieve information about any cmdlet that includes "Exchange" in its name. This is an unfortunate reduction of functionality that

Microsoft plans to address in a future release of PowerShell. However, you can still do the following:

- Use the `Get-Command` cmdlet to list the cmdlets that 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 of the cmdlets available to work with contacts (shown in the following example). You can also use the shortened alias of *gcm* for `Get-Command`. This is one way to work around the loss of functionality in the `Get-Help` cmdlet.

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: `Get-Help Get-DistributionGroup -Full`.
- Use the `-Examples` switch to see whatever examples of a cmdlet in use EMS help includes: `Get-Help Get-MailboxServer -Examples`.
- Use the `-Parameter` switch to get information about a selected parameter for a cmdlet: `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 once 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 and 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 get something done. 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 it is easy to get work done. 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 EMC and navigate to the right place to make the change through the 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 the EMC and other places throughout Exchange, such as setting up servers and databases in the setup program (the blog written by Glen Scales at <http://gsexdev.blogspot.com/> provides many good examples of how to call EMS cmdlets). 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 each other in a common manner to allow them to process data in a consistent manner, no matter what cmdlet provides the data. Programmers therefore do not have to worry about reformatting data for input to specific cmdlets, so the task of assembling different cmdlets together into a script to do a job is much easier. Microsoft built Windows 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 plain text, 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 really very elegant.

Command editing

It should already be obvious that you could do a lot of typing to enter commands into Windows PowerShell, make the inevitable mistakes, correct, and try again. To make life a