Microsoft

Inside OUT

The ultimate, in-depth reference

Hundreds of timesaving solutions

Supremely organized, packed with expert advice

Microsoft Exchange Server 2013: Connectivity, Clients, and UM

Paul Robichaux *Microsoft MVP for Exchange Server*

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

Library of Congress Control Number: 2013948709
ISBN: 978-0-7356-7837-8

Printed and bound in the United States of America.

First Printing

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

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

Contents at a Glance

Table of Contents

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

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

microsoft.com/learning/booksurvey

Foreword for Exchange 2013 Inside Out books

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

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

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

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

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

Enjoy!

Rajesh Jha
Corporate Vice President - Exchange
Microsoft Corporation

Introduction

This book is for experienced Exchange administrators who want to gain a thorough understanding of how client access, transport, unified messaging, and Office 365 integration work in Exchange Server 2013, the latest version of the Microsoft enterprise messaging server first released in October 2012 and updated on a frequent basis since. It isn't intended to be a reference, and it isn't suitable for novices.

In 2011, when Tony Redmond and I were working together to present the Exchange 2010 Maestro workshops in cities throughout the United States, we spent a lot of time talking about the nature of an ideal Exchange book. It should be comprehensive enough to cover all the important parts of Exchange, with enough detail to be valuable to even very experienced administrators but without just parroting Microsoft documentation and guidance. As far as possible, it should draw on real-world experience with the product, which of course takes time to produce. Out of those talks came Tony's idea to write not one but two books on Exchange 2013. A single book would either be unmanageably large, both for author and reader, or would omit too much important material to be useful.

Although Tony's *Exchange 2013 Inside Out: Mailbox and High Availability* (Microsoft Press, 2013) draws on his long and broad experience with the nuances of the Exchange mailbox role and how to put it to work, this book covers all the other things Exchange does, including client access, transport, unified messaging, and the increasingly important topic of Office 365 integration. Because Exchange 2013 is an evolution of Exchange 2010, we decided to use *Microsoft Exchange Server 2010 Inside Out* (Microsoft Press, 2010) as the base for the new book. For the topics in this book, so much has changed since Exchange 2010 that only a small amount of the original material remains. The rest is new and was written to take into account the many changes and updates that Exchange 2013 has undergone since its original release.

I have had the good fortune to work with and around Exchange for nearly 20 years. During this time, I've seen the Exchange community, product team, and product evolve and grow in ways that might not have been predictable back in 1996. If you went back to, say, 2000 and told the Exchange product group, "Hey, in 2013, your product will be deployed to hundreds of millions of users worldwide, many with tiny handheld computers that are more powerful than your desktop, and a whole bunch of them running as a Microsoft-hosted service," you'd be bound to get some skeptical looks, and yet here we are.

I hope that you enjoy this book and that you'll read it alongside Tony's *Microsoft Exchange Server 2013 Inside Out: Mailbox and High Availability*. The two books really do go together. Tony and I exchanged technical editing duties for our respective books, so we share responsibility for any errors you might find.

Acknowledgments

I was incredibly fortunate to receive a great deal of help with this book from a variety of sources. A large group of Exchange experts from the Microsoft Most Valuable Professional (MVP) and Microsoft Certified Systems Master (MCSM) communities volunteered their time to read early drafts of the chapters as they were produced; their mission was to identify shortcomings or errors and to suggest, based on their own experience, ways in which the book could be improved. This book is much better thanks to their efforts, which I very much appreciate. My thanks to Kamal Abburi, Thierry Demorre, Devin Ganger, Steve Goodman, Todd Hawkins, Georg Hinterhofer, Miha Pihler, Maarten Piederiet, Simon Poirier, Brian Reid, Brian R. Ricks, Jeffrey Rosen, Mitch Roberson, Kay Sellenrode, Bhargav Shukla, Thomas Stensitzki, Richard Timmering, Steven van Houttum, Elias VarVarezis, Johan Veldhuis, and Jerrid Williams. My thanks also go to the broader MCM and MVP communities, particularly Paul Cunningham, Brian Desmond, and Pat Richard, for discussing topics or sharing scripts that informed the material I wrote.

In addition to these volunteers, I benefited greatly from the efforts of many people from the product team, including Diego Carlomagno, Bulent Egilmez, David Espinoza, Kern Hardman, Pavani Haridasyam, Tom Kaupe, Roy Kuntz, Lou Mandich, Jon Orton, Tony Smith, Greg Taylor, and Mini Varkey. Extra thanks to Rajesh Jha for taking the time to write the foreword for both books—no easy task considering how often Tony and I have hassled him about various matters.

Finally, you wouldn't have this book at all if it weren't for the stalwart efforts of Karen Szall, Valerie Woolley, and a cast of dozens at Microsoft Press. Karen never lost her temper despite the many vigorous discussions we had about my failure to meet deadlines or my obstinacy toward some of the requirements imposed by the Microsoft crack legal department. Thanks to them all for producing such a good-looking finished product.

Errata & book support

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

http://aka.ms/EXIOv2/errata

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

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

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

We want to hear from you

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

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

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

Stay in touch

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

CHAPTER 3

Client management

Although administrators think of Exchange Server as a complex, server-based system, the reality is that many of the millions of Exchange users worldwide think of their email system as Microsoft Outlook. This is a testament to the Microsoft Office team's branding efforts, but it also reflects a simplistic view of the Exchange client landscape. The truth is that there are six categories of Exchange clients:

- Outlook remains the Microsoft premium fat or rich client. Microsoft long ago made a conscious decision to tie client-side and server-side features together in Exchange and Outlook so that key features in each new release require you to deploy the client and server together for maximum benefit. Outlook 2013 still uses Messaging Application Programming Interface (MAPI) (although it's now tunneled over HTTPS), and Outlook 2007 and Outlook 2010 are still fully supported.

- Outlook Web App has come a long way since Microsoft introduced the first versions in Exchange 5.5. The modern Outlook Web App client looks and behaves very much like Outlook 2013, and it supports a broad array of browsers on both conventional computers and mobile devices. The addition of a special touch mode for tablets, coupled with an offline mode for selected modern browsers, shows some tantalizing indications that Microsoft intends the web-based Outlook Web App experience to match or surpass native mobile-device clients in both flexibility and capability.

- On the other hand, the fact that Microsoft is now shipping Outlook Web App clients as native applications for the Apple iPad and iPhone is an indication that there's a place for purpose-built mobile clients. These clients, collectively known as Outlook Web App for Devices (perhaps indicating a future release for other platforms), don't use Exchange ActiveSync; instead, they are based on a combination of Exchange Web Services (EWS) and HTML5. The current iOS versions provide full calendar and mailbox access, including push notification support, offline capability, and a wealth of other features formerly reserved for Apple's native device clients.

- Speaking of native clients, another category of Exchange client includes Exchange ActiveSync (EAS) clients such as Apple Mail for iOS and the Windows Phone 8 version of Outlook. These clients, the EAS protocol itself, and the management thereof are described fully in Chapter 4, "Mobile device management."

- EWS is the protocol that Outlook 2011 for Mac OS X (hereafter just called Outlook 2011) uses exclusively. Many other applications, including Outlook 2010 and Outlook 2013 and the Lync client, use EWS because it provides a straightforward, non-MAPI way to access and modify pretty much every type of Exchange data object, including messages, contacts, rules, and folders. Another example: SharePoint uses EWS to access and store data in site mailboxes. EWS is also portable across platforms; Linux, Mac OS X, iOS, Windows Phone, and Android clients also use EWS.

- A last category comprises clients using POP3 and IMAPv4. Although these protocols are mature and remain popular on Linux clients, they offer limited features and poor performance compared to newer protocols such as EAS and EWS. Microsoft has invested little to no engineering resources in updating Exchange 2013 POP/IMAP support compared to previous versions, and these protocols are discussed only briefly in this chapter.

In Exchange 2013, Microsoft has made dramatic changes to the client experience in several ways. It's added new features, such as site mailboxes, that use and require the latest version of Outlook. It's made many changes intended to improve the built-in Outlook Web App client and extended and improved the EWS and EAS application programming interfaces (APIs) of which other clients take advantage.

Choosing a client

The first question many organizations have when they consider deploying a new version of Exchange is, "What client should we use?"

Outlook

Over the years, the strength of the relationship between the Exchange and Outlook product teams has waxed and waned. Like most other large enterprises, organizational changes and political maneuvering have influenced the degree to which these teams work together. The most noticeable area of collision between the two teams' plans has been the release schedule of their respective flagship products. In the past, some versions of Office have shipped before the corresponding Exchange release, whereas others have shipped after Exchange. In either case, this poses a conundrum for companies that want the better-together experience that Microsoft always promises. Deploying Exchange or a new version of Office is a challenge, and doing them both concurrently is even more challenging. For example, Exchange 2010 was released before Outlook 2010, limiting the ability of early Exchange

adopters to realize full value from their deployment. Office 2013 didn't ship until after Exchange 2013, but the delay in getting Exchange 2013 Cumulative Update 1 (CU1) out the door provided a handy opportunity to plan and execute Office deployments while waiting for the ability to deploy Exchange 2013 into existing Exchange 2007 or Exchange 2010 environments.

What version of Outlook should you deploy? Answering this question is easier for small companies than it is for large ones. The law of numbers conspires to create much greater complexity when a new application must be distributed to tens of thousands of desktops and issues such as user training, preparing the help desk to support the rollout, and the cost of new software licenses and potential hardware upgrades are considered. For example, the introduction of the Office fluent interface (featuring the ribbon) in Outlook 2007 caused a lot of controversy because it was unfamiliar to users. Office 2013 makes a number of large changes to the user interface and application features, and these changes have not been universally welcomed by users so far. This is why so many companies continue to run older versions of Outlook; they see little value in going forward with an upgrade that promises great cost for new licenses and deployment while offering little obvious return in the form of user productivity, lower support costs, or anything else. The fact that the Exchange server Client Access Licenses (CALs) no longer include a license for Outlook also makes it harder for companies to justify an early upgrade.

Table 3-1 briefly summarizes major features in each Outlook version and how they work with Exchange 2013.

TABLE 3-1 Comparing different versions of Outlook

Outlook Version	Major Features
Outlook 2003	Introduction of cached Exchange mode and smarter networking to enable faster and more efficient synchronization between server folders and local replicas. Exchange 2010 requires Outlook 2003 SP2. Not supported by Exchange 2013.
Outlook 2007	Introduction of Autodiscover functionality to enable automatic configuration of user profiles. Movement away from public folders as the repository for shared data such as free/busy and Offline Address Book (OAB) to use web-based distribution instead. First implementation for managed mail and retention policies.

Outlook Version	Major Features
Outlook 2010	The first 64-bit version of Outlook (also available for 32-bit platforms). Supports features such as MailTips and message tracking from within Outlook. Far more developed and feature-complete version of messaging record management (document retention) policies. Supports cross-organization calendar sharing to help customers deploy in mixed on-premise/hosted deployments. Supports conversation view of email threads (also works with earlier versions of Exchange) and the ability to ignore threads in which you're not interested in email. Outlook 2010 also supports personal archives located on Exchange 2010 servers and can open up to three Exchange mailboxes in addition to the primary mailbox.
Outlook 2013	Revamped user interface, including touch mode for Windows tablets and touchscreen devices. Support for site mailboxes and modern public folders. Changes sync behavior in cached Exchange mode. Supports EAS connections to Outlook.com. Various user interface additions, including inline replies and the Weather Bar, a subwindow that displays weather in the calendar view.

Outlook 2013

Outlook 2013 brings some interesting new functionality to the equation. Whether the new features are worthwhile enough to consider an upgrade is different for every company.

Since Microsoft introduced cached Exchange mode in Outlook and Exchange 2003, it has continually worked to make sync faster and more bandwidth-efficient. However, one area it hasn't previously addressed much is the question of what should be synced in the first place. Outlook 2013 offers two new features to improve the user's experience: Exchange Fast Access and the new sync slider.

As Exchange and Outlook have evolved together, the way synchronization works has changed, too. Exchange Fast Access is the latest stage in this evolution; the basic idea is that a sync operation that takes long enough for the user to notice should continue in the background. Exchange Fast Access immediately syncs the user's most recent email and all his calendar data whereas older mail items are synced in the background. For example, a user who returns from a long vacation will see her most recent messages right away. You can turn off this feature by setting the HKEY_CURRENT_USER\software\policies\microsoft\office\15.0\outlook\hybrid\localcaching DWORD to 0, but you can't configure any other aspect of Fast Access.

The sync slider is intended to handle a different problem. The traditional approach to sync control has been to allow (or force) the user to choose which folders to synchronize. This is great if you're a power user, but it falls far short for users who don't necessarily know which folders they actually use because they file and find items through Outlook search tools. The new Outlook sync interface (shown in Figure 3-1) instead asks users to choose how much

mail they want to sync based on time. The idea here is that even nontechnical users can decide whether they need a month, a year, or all their email synced, so the slider enables the user to make that choice. Outlook is supposed to handle the rest.

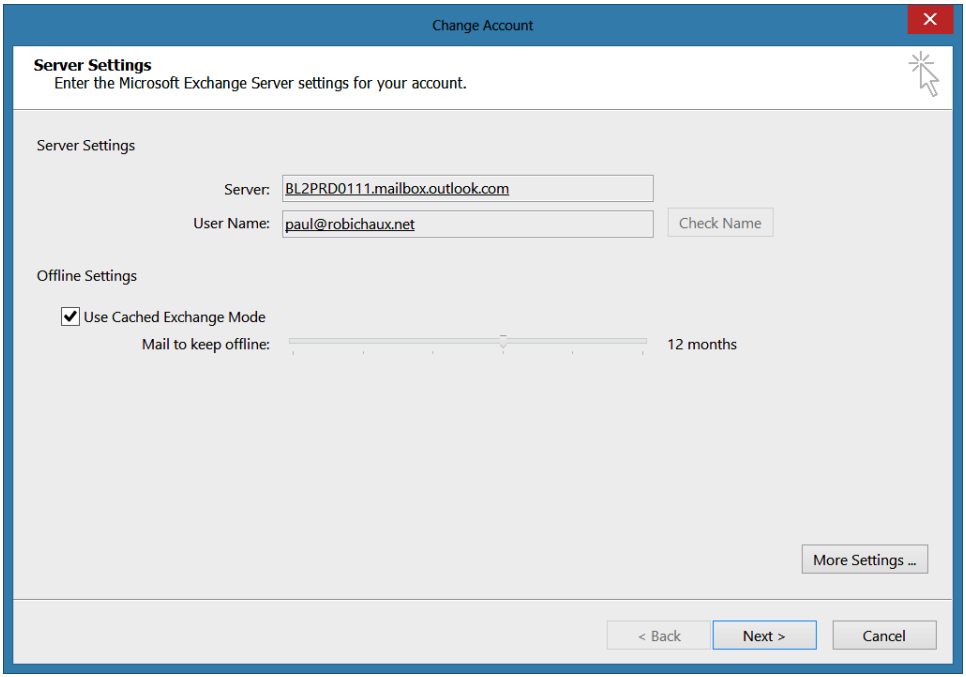

Figure 3-1 The Outlook account settings dialog box, which now includes a slider that controls synchronization behavior

If the slider is set to anything other than All (the rightmost value), you see a subset of your mail while in cached mode. If you scroll to the bottom of a list or perform a search, you'll see that Outlook provides a link telling you that more results are available on the Exchange server; clicking that link loads additional messages from the server. Interestingly, the slider doesn't affect sync for calendar items, contacts, tasks, or notes; all those items are always synchronized.

By default, Outlook 2013 syncs 12 months of mail, which is plenty for many users. However, you might want to apply a consistent value for this setting to avoid confusing users (and the resulting help desk calls) who don't see mail they expected. The SyncWindowSetting registry value (a DWORD under HKEY_CURRENT_USER\Software\Policies\Microsoft\Office\15.0\Outlook\Cached Mode) enables you to set the slider value through a Group Policy object (GPO). Set the value to the number of months of mail data you want to sync; legal values are 0 (meaning the entire mailbox), 1, 3, 6, 12, or 24. The sync mechanism deposits mail in a compressed OST file; more precisely, some data items in the file are

compressed, and others are not. Microsoft claims up to a 40 percent space savings compared to older versions of Outlook.

Another change is that Outlook 2013 can open up to 9,999 Exchange mailboxes concurrently, not all of which have to belong to the same Exchange organization. This is a significant increase over the Outlook 2010 limit of 10 mailboxes! By default, Outlook imposes a limit of four mailboxes. This is deliberately set to prevent Outlook from taking up huge amounts of system resources on a client PC, which would occur if someone attempted to open 10 or 20 mailboxes. However, you can increase the limit for concurrent open mailboxes up to the maximum by updating the value held in the registry at HKCU\Software\Microsoft\Exchange\MaxNumExchange.

Outlook 2013 also includes some miscellaneous new features, such as a weather display and the ability to warn you when you use phrases such as "see attached" or "I've attached" but then don't include an actual attachment.

Outlook 2010 and Outlook 2007

Microsoft fully supports Outlook 2007 and Outlook 2010 with Exchange 2013, provided that they are updated to the required versions. Outlook 2010 requires SP1 with the November 2012 cumulative update (CU), and Outlook 2007 requires SP3 and the November 2012 CU.

If you use either of these versions, you'll miss out on some key Exchange 2013 features:

- You can't access site mailboxes from Outlook, although they are still available from within Microsoft SharePoint.
- Your users won't see data loss prevention (DLP) Policy Tips warnings in Outlook 2013.
- Your users can't install or run apps from the Office Store.

Outlook 2007 users will miss out on some additional features that were first introduced in Exchange 2010:

- No user interface is available to display the MailTips the server provides.
- Outlook 2007 doesn't understand the internal identifiers Exchange uses to connect related items in a conversation, so none of the conversation-related features (including conversation views, the ability to clean up a conversation, and the Ignore button) are supported.
- Integration with the settings slabs in Outlook Web App to allow managing group information, editing unified messaging (UM) settings (such as call answering rules),

and so on are missing. However, users can still open Outlook Web App options slabs directly to access these options.

- Personal archives are accessible only if you deploy the update for Outlook 2007 released by Microsoft in late 2010.
- Outlook 2007 can render voice mail previews as plain HTML in the message body but lacks the control necessary to play the voice content if you click part of the voice mail preview. Also, it cannot process protected voice mail.
- You cannot send Short Message Service (SMS) messages from Outlook 2007.
- There is no support for retention tags and policies.

Earlier versions of Outlook

Outlook 2003 and earlier versions are no longer supported. Apart from the countless improvements and bug fixes in newer versions, there are two important technical reasons the earlier versions aren't supported. First, Exchange 2013 requires clients to use Autodiscover, and Outlook 2007 was the first Outlook client to support it. Second, Exchange 2013 allows clients to connect only through RPC-over-HTTPS (or Outlook Anywhere), which was first supported in Outlook 2003.

Outlook Web App

As Outlook Web App and Outlook have matured, Microsoft has made Outlook Web App look and behave more like Outlook with each successive release. That raises the question of what the company's long-term intentions are. I think it's safe to say that the Office team will keep making new versions of Outlook as long as there is an Office team and, likewise, that the Exchange team will keep improving Outlook Web App. The enhancements in Outlook Web App 2013, though, seem to point to something else: Microsoft is laying the groundwork to obviate the need for third-party Exchange ActiveSync clients on smart phones and tablets in two ways:

- By improving the browser-based experience as evidenced by the combination of a touch mode in Outlook Web App, designed to make Outlook Web App friendlier on devices that lack a mouse or trackpad, and the availability of offline storage
- By shipping native apps for Apple iOS (and possibly for other platforms in the future) that can replace the built-in Mail and Calendar applications

In late 2012 and early 2013, a pair of serious bugs in Apple's iOS caused problems for Exchange administrators who suddenly found their servers flooded with transaction logs. Since then, Microsoft and Apple have begun working much more closely to ensure that iOS

and Exchange get along well, and the Exchange team has added both bug fixes and new features to help prevent a misbehaving client from causing server-side problems. Even with these changes, though, it probably makes sense for Microsoft to enable Exchange 2013 customers to ditch the built-in clients for many use cases (although some actions, such as sending a message from a built-in photo app, for example, might still require use of the built-in client).

New features in Outlook Web App 2013

The biggest new feature in Outlook Web App 2013 is its new interface, which was explicitly designed to look like Outlook 2013. The interface features much more white space, and most icons have been either removed or replaced by text labels. For instance, there are no icons in the folder list on the left side of the window, and the icon-based toolbar of previous versions is largely gone (see Figure 3-2). Opinions are divided over this visual style; some people really like it, whereas others say that it uses screen space inefficiently and that there's too much white.

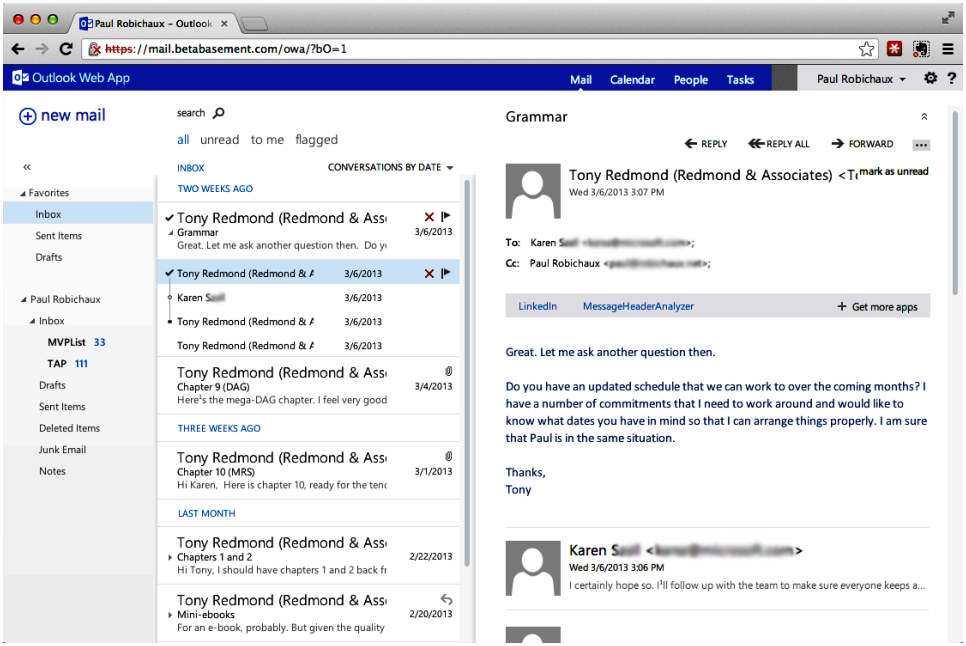

Figure 3-2 The Outlook Web App 2013 mail interface in Chrome on Mac OS X

Whether or not you like it, though, it's safe to say that this design will be with us for a while, given that the same design elements are used in Xbox, Windows 8, and the other components of the Office 2013 family.

Extending Outlook Web App with apps

Almost every significant desktop and mobile operating system platform has its own store in which third parties can sell applications for the platform. Apple started it, but Microsoft, Google, BlackBerry, Nokia, and other vendors have all followed suit. The expansion of the store model has now expanded further with the introduction of the Office Store in Office 2013. Microsoft now provides a centralized place where companies of all sizes can sell add-ins for Office, such as executable add-ins for Microsoft Excel or Outlook, document templates for Office Word, or applications that plug in to Outlook 2013 and Outlook Web App 2013. This last category, of course, is what this chapter focuses on the most.

Microsoft refers to this new application development model as the *cloud app model*; it is predicated on the idea that a lightweight app that connects to web service APIs is a useful way to add new functionality to desktop Office applications, SharePoint sites, and Outlook Web App. Applications built using this model are essentially containers of HTML and JavaScript that connect to websites to offer actions based on the context and content of the document or page on which they are triggered. If you remember Microsoft Smart Tags for Office 2007, this model should sound familiar; the cloud app model shares with it the notions that the document or page is the center of activity for the user and that the app should offer services that make sense in context.

The Outlook client itself has long had a number of extensibility models that enable developers to create software that integrates with Outlook on the client. The new model supplements but does not replace the existing extension capabilities. For example, cloud apps for Outlook Web App (and SharePoint) are installed on the server, either for individual users or the server, but not on individual users' computers. The apps are constrained in the types of data they can access because they run in the security context of the user's browser. However, because these apps can integrate with third-party web services, they can offer a surprising degree of utility. The Office Store catalog at *http://office.microsoft.com/en-us/store/apps-for-outlook-FX102804983.aspx* lists several dozen apps that are intended to plug in to Outlook 2013 and Outlook Web App 2013, including extensions that integrate consumer services such as Twitter, LegalBox, and LinkedIn, plus utilities of various sorts. You can also deploy apps by publishing them in an internal app catalog, which gives you a straightforward way to publish apps for enterprise users. This app catalog can be stored on an Exchange server, a SharePoint site, or in a file.

Administrators can block individual apps, turn off apps for the organization, or turn off apps for an individual user. The last feature enables you to prevent an individual user from installing apps that run in her local instance of Outlook 2013 because the Outlook Web App app model requires the apps to be stored in the user's mailbox on the server. In addition, separate Outlook 2013–specific features enable you to control app behavior within Outlook. For more information on configuring and managing these apps, see the "Managing Office Store apps for Outlook Web App" section later in this chapter.

Browser and operating system support

The best way to summarize the state of browser and operating system support for Outlook Web App 2013 is to say it's broad. Microsoft characterizes Outlook Web App support in three ways:

- Light refers to a fairly limited set of features. In Light mode, there's no drag-and-drop, and the user experience looks essentially just like Outlook Web App Light mode in Outlook 2007 (with a few minor stylistic changes). Figure 3-3 shows the Outlook Web App Light mode in Microsoft Internet Explorer 10 on Windows 8.

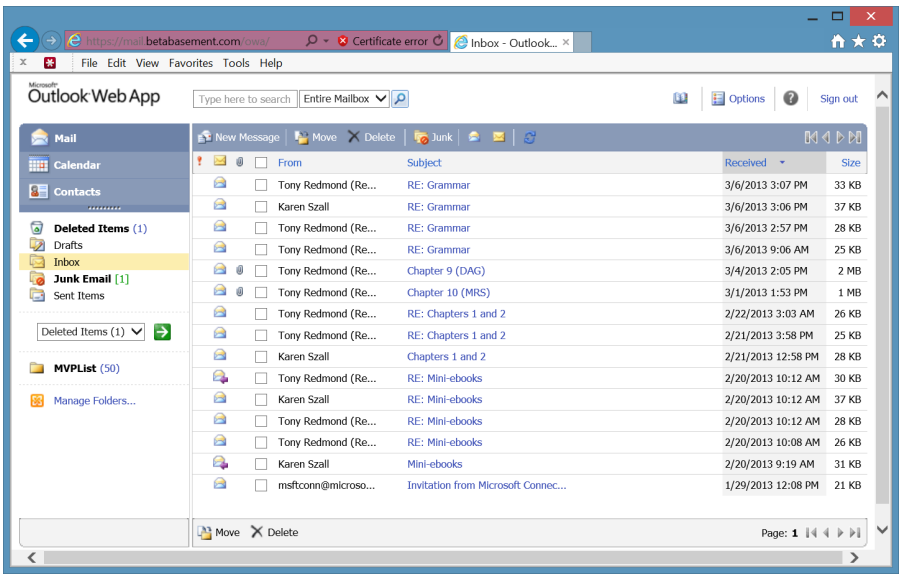

Figure 3-3 The Outlook Web App 2013 mail interface in Light mode on Internet Explorer 10 on Windows 8

- Good is better than Light; it adds some additional features to the Light experience.
- Best gives users the full Outlook Web App 2013 experience, including automatic refresh of message lists, autocompletion of address fields, drag-and-drop, and more. Some browsers also get touch mode and offline support.

Table 3-2 summarizes the support levels for various combinations of operating systems and browsers. Internet Explorer versions 9 and 10, Safari version 6 on Mac OS X, Firefox version 17, and Chrome version 24 are the minimum versions for the Best mode experience across platforms. Chrome, Safari, and Internet Explorer 10 offer offline access in addition to the Best experience.

There is currently no way to force downgrades for a client; for example, if you want to force your Firefox users to get the Good or Light experience, you can't. You can append "?layout=light" to the Outlook Web App URL to force it into Light mode for a particular client, although this is not officially documented anywhere. In addition, you can use Set-OWAVirtualDirectory –LogonPageLightSelectionEnabled to control whether users can choose Light mode themselves or Set-OWAVirtualDirectory –OWALightEnabled to control whether Light mode is available at all.

TABLE 3-2 Browser and operating system support for Outlook Web App 2013

Operating System	Browser	Support level
Windows XP, Windows Server 2003	Internet Explorer 7	Light
	Internet Explorer 8	Good
	Firefox 17+	Good
	Chrome 24+	Good plus offline access
Windows Vista, Windows Server 2008	Internet Explorer 8	Good
	Internet Explorer 9	Best
	Firefox 17+	Good
	Chrome 24+	Good plus offline access
Windows 7	Internet Explorer 8	Good
	Internet Explorer 9	Best
	Internet Explorer 10	Best
	Firefox 17+	Best
	Chrome 24+	Best plus offline access
Windows 8	Internet Explorer 10	Best plus offline access
	Firefox 17+	Best
	Chrome 24+	Best plus offline access
Mac OS X 10.5+	Chrome 24+	Best plus offline access
	Firefox 17+	Best
	Safari 6+	Best plus offline access
Linux	Firefox 17+	Best
	Chrome 24+	Best plus offline access

Notice that Table 3-2 doesn't mention mobile browsers; that's on purpose. Microsoft only supports touch mode for Outlook Web App on a smaller set of browsers: Safari for the Apple iPad and Internet Explorer 10 for Windows RT and Windows 8.

Deprecated features from Outlook Web App 2010

Microsoft giveth, and Microsoft taketh away. In every new release of Exchange, along with the new features we get, some old ones are removed, and Outlook Web App 2013 is no exception. The deprecated feature that's garnered the most commentary is spell checking; Outlook Web App 2013 depends on the browser to do it rather than including it as a feature. This is an eminently reasonable decision, given the capabilities of modern browsers. Some other features were cut from the RTM version of Exchange 2013 that you might or might not miss:

- Attachment previews are now generated by an Office Web Apps server instead of by the WebReady feature included in Exchange 2010. See the "The role of Office Web Apps Server" section later in this chapter for more details.
- Concerning distribution list moderation, you can't moderate messages sent to distribution lists in Outlook Web App 2013.
- S/MIME encryption and signatures aren't supported in Exchange 2013, although Microsoft documentation says they will be supported in a future version.
- You can only have the reading pane on the right side of the window; there's currently no option to move it to the bottom.
- You can't reply to email messages that are embedded as attachments to other messages.

Mac OS X

If you have Mac OS X users, you essentially have four client choices. Which one you choose will depend in large measure on the number of Mac users you have, how vocal they are, and their tolerance for (or appetite for) aggravation.

- You can deploy Outlook 2011, part of Mac Office 2011. Although it shares a name with Windows Outlook, Mac Outlook is a very different beast, with a completely different user interface and many differences in functionality compared to the Windows version. Because it is a Mac-native application, and because the entire Office suite is familiar to most Mac users, this is a fairly safe choice. Having said that, Outlook 2011 has a reputation for being buggy and slow, though many of the supposed bugs are actually design choices that some Mac users don't like.
- You can use Apple's Mail and iCal applications, which come bundled with the operating system. They support EWS, and they are generally stable and performant. However, they are not very good as email and calendar clients compared to Outlook Web App or Outlook, with many missing features and some annoying behaviors

(such as sometimes failing to hide deleted messages) that stem from their legacy as IMAP/POP clients. Apple has gradually been improving the degree of Exchange support in each Mac OS X release, and die-hard Mac users will probably prefer this option to having to learn to use Outlook.

- You can let your Mac users have Outlook Web App, which requires little to no effort on your part and gives them a pretty good experience overall. However, because Outlook Web App is a browser-based application, it requires the use of a supported browser, and it doesn't offer all the features the native desktop clients do.

- You can run Outlook 2007, Outlook 2010, or Outlook 2013 in a Windows virtual machine. This gives users a true cross-platform experience because they are literally running the same code your Windows users do. Many organizations already provide virtualized Windows desktops for their Mac users; if not, setting up and deploying this option on a wide scale can be a challenge (and an expensive one at that).

Outlook Web App for Devices

Despite the rather clumsy name, Outlook Web App for Devices is a very cool addition to the field of Exchange clients. Released as an almost complete surprise in July 2013, the Outlook Web App for Devices clients run on most models of iPad (iPad 2 and later) and iPhone (iPhone 4S and later). The visual appearance of the client is very faithful to the browser-based version of Outlook Web App running on modern desktop browsers; the icons, typography, spacing, color scheme, and so on are nearly identical to what you see when loading Outlook Web App in Internet Explorer 10 or a recent version of Chrome or Firefox. Perhaps more importantly, the app offers a number of features that aren't available in the built-in apps shipped by Apple, including the ability to send and read messages protected with Active Directory Rights Management Services (AD RMS), access to personal archive mailboxes, the ability to display MailTips, and full integration of Office apps.

The app includes modules for email, calendar, and contact access. The calendar portion of the app (see Figure 3-4) is a huge improvement over the native iOS calendar app; it maintains a very close visual resemblance to the desktop Outlook 2013 client and provides full access to shared calendars. The biggest feature of interest in the contacts module is that contacts from your Exchange accounts can be synchronized with the local contact store, meaning that name resolution for phone calls will work properly.

The implementation of the app is interesting. Rather than using Exchange ActiveSync, as the native clients on iOS and Windows Phone do, Outlook Web App for Devices uses EWS for mail synchronization. The app itself includes a middleware layer that Microsoft calls PAL (for "platform abstraction layer") that ties the JavaScript implementation of Outlook Web App together with native functionality on the device. Local storage is provided by the SQLite database engine included with iOS. From the server's perspective, the traffic

generated by the mobile app looks like a mix of browser Outlook Web App operations and EWS traffic.

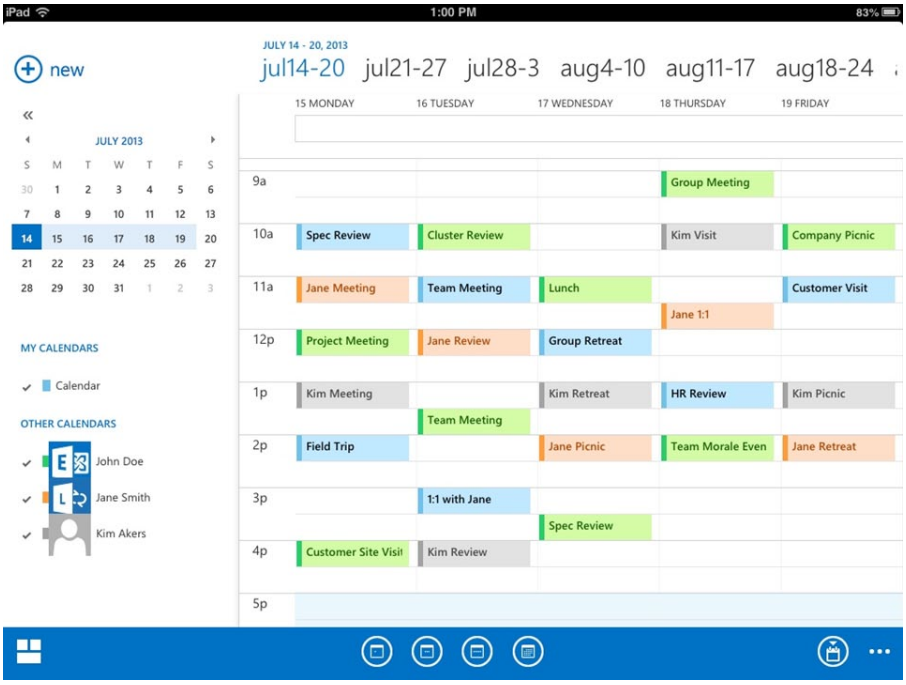

Figure 3-4 Outlook Web App for Devices can display shared calendars, and the result is visually nearly identical to the desktop Outlook 2013 client

The app maintains its own separate data store, so if it receives a remote wipe request, it erases all the application data and settings but doesn't affect any other data on the device. Likewise, security policies that you apply with mailbox or Outlook Web App policies on the server will be honored and enforced by the app, without any impact on the device operating system or on other applications. This is a welcome change for bring-your-own-device (BYOD) organizations because it means that issuing a remote wipe to a user won't remove the user's photos, music, or other personal data from a device that he owns.

As with the Office productivity apps for iOS, Outlook Web App for Devices was released first for Office 365 users. It is currently only supported for Office 365 tenants; Microsoft has said that it will enable it for use with on-premises Exchange 2013 deployments, but as of late 2013 it has not yet done so. As a means of giving customers incentives to move to Office 365, this isn't a bad strategy, but it seems to leave a bad taste in the mouths of many on-premises customers who feel like second-class citizens.

Managing Outlook for Windows

You can do much of what you need to do to manage Outlook clients on Windows by changing settings on the Client Access Server (CAS) itself, as described in Chapter 1, "Client access servers." This section talks specifically about actions you might need or want to perform to ensure the smooth functioning of your organization's Outlook clients.

Managing Outlook Anywhere

Outlook Anywhere was first introduced in Exchange 2003, and it has evolved quite a bit since then. The benefits of allowing clients to get their email without requiring a virtual private network (VPN) connection are significant, but previous versions of Outlook and Exchange have sometimes made Outlook Anywhere configuration somewhat complicated. Exchange 2013 simplifies the Outlook Anywhere world by making it the only protocol Outlook can use to access Exchange mailboxes. This eliminates much of the confusion inherent in trying to figure out which protocol the client would use under different circumstances. Exchange 2013 greatly streamlines the Outlook Anywhere deployment experience, too, because all you need to do is install a valid Secure Socket Layer (SSL) certificate on your client access servers and then make a few minor configuration tweaks. In this case, "valid" means that the certificate is issued by a certification authority (CA) the clients can trust—either a public CA or an internal CA whose certificate chain is on the trust list for the clients. Although you may install a public CA certificate on your Mailbox servers, this is not necessary.

Right out of the box, if you don't do anything, Outlook Anywhere works fine for internal clients. It might or might not work for external clients, depending on how you've set up your Internet-facing and internal-facing CAS servers. When you first install a plain Exchange 2013 Mailbox server, the Outlook Anywhere virtual directory is configured with an internal hostname that matches the machine FQDN, but the external hostname is blank, and SSL is not required to connect. For all your Internet-facing CAS servers, you should set an external hostname and enable the use of SSL for both internal and external clients. For example, suppose you wanted to configure an Internet-facing server named PAO-EX01; that's easily done with the following command:

```
Set-OutlookAnywhere –id 'PAO-EX01\rpc (Default Web Site)' –ExternalClientRequiresSSL
$true –InternalClientRequiresSSL $true –externalHostname 'mail.betabasement.com'
```

The "CAS authentication methods" section in Chapter 1 outlines the authentication methods Exchange can use in various scenarios, including Outlook Anywhere. An unmodified Exchange 2013 installation will have negotiate authentication set up for external clients and NTLM/Kerberos set for internal clients; this is generally optimal as is, so don't change it unless you have a very good reason.

Coexistence among Exchange 2007, Exchange 2010, and Exchange 2013 Outlook Anywhere is fairly simple to set up. The key is to point all incoming Outlook Anywhere traffic at an Exchange 2013 CAS. However, the Exchange 2013 CAS role doesn't have the remote procedure call (RPC) proxy code contained in RPCproxy.dll because it's not an RPC proxy! That means the Exchange 2013 CAS uses HTTP proxying to send the encapsulated RPC-over-HTTP packets to an Exchange 2007 or Exchange 2010 CAS, which does have RPCproxy.dll and thus can de-encapsulate the Outlook Anywhere packets. This design has a couple of side effects: you must have the RPCproxy.dll installed on your Exchange 2007 or Exchange 2010 CAS servers, and you must have those servers configured for Outlook Anywhere. You must enable Outlook Anywhere on every Exchange 2007 and Exchange 2010 CAS, even if it's not Internet-facing, because Exchange 2013 can proxy traffic to internal servers as part of its proxy process. Coexistence proxying also requires an authentication change; you must configure your Exchange 2007 and Exchange 2010 CAS servers to allow integrated Windows authentication on the /rpc virtual directory.

INSIDE OUT **Everything is internal**

One consequence of the Outlook Anywhere changes in Exchange 2013 is that Outlook always displays the internal hostname in the Exchange Proxy Settings dialog box, even if Outlook is connecting to the external hostname because it's on an external network. Microsoft claims in KB article 2754898 that this is by design, but it's not clear why the company would design it this way; keep this in mind in case your users complain that their Outlook clients are connecting to the "wrong" server.

Managing Autodiscover

The "Autodiscover" section in Chapter 1 describes what the Autodiscover protocol is and how it works. The key thing to remember with respect to the use of Autodiscover in Outlook is that the Autodiscover XML manifest typically gives you all the clues you need to understand the cause of any problems you encounter. The Exchange 2013 implementation of Autodiscover is different in a few respects from that of earlier versions. First, remember that Exchange 2013 emits two EXHTTP nodes as part of its Autodiscover response: one for the internal Outlook Anywhere configuration and one for the external version, in that order. Outlook clients are supposed to try these two configurations in order, ignoring any older EXPR elements. The new EXHTTP nodes are part of why you have to update Outlook 2007 and Outlook 2010 with the November 2012 (or later) CU before they work properly with Exchange 2013. In addition, Autodiscover itself doesn't require any configuration because it merely publishes the URLs and service names that are configured on other objects, such as the Exchange ActiveSync virtual directory and the endpoints for Outlook Anywhere.

Using the Exchange Remote Connectivity Analyzer

One of the coolest things about the Exchange team is its habit of innovating in ways that other product groups later copy. For example, Exchange was the first Microsoft product to use what we now think of as AJAX; the team was the first to produce a best-practices analyzer to help customers ensure that their deployments were optimized; and it was the first to include support for Microsoft Windows PowerShell. Another first that deserves mention in the context of client management is the Exchange Remote Connectivity Analyzer, or ExRCA. ExRCA is a tool originally championed by the Exchange product support team, which was seeking a better way to help customers troubleshoot connectivity problems. The tool analyzes many aspects of the connection between an Outlook or mobile client and your Exchange server, including the certificates used, the CAS configuration, and how Autodiscover is configured, and it tells you when it finds problems in a clear, easy-to-read report that makes it easy to resolve problems. Figure 3-5 shows a sample ExRCA report.

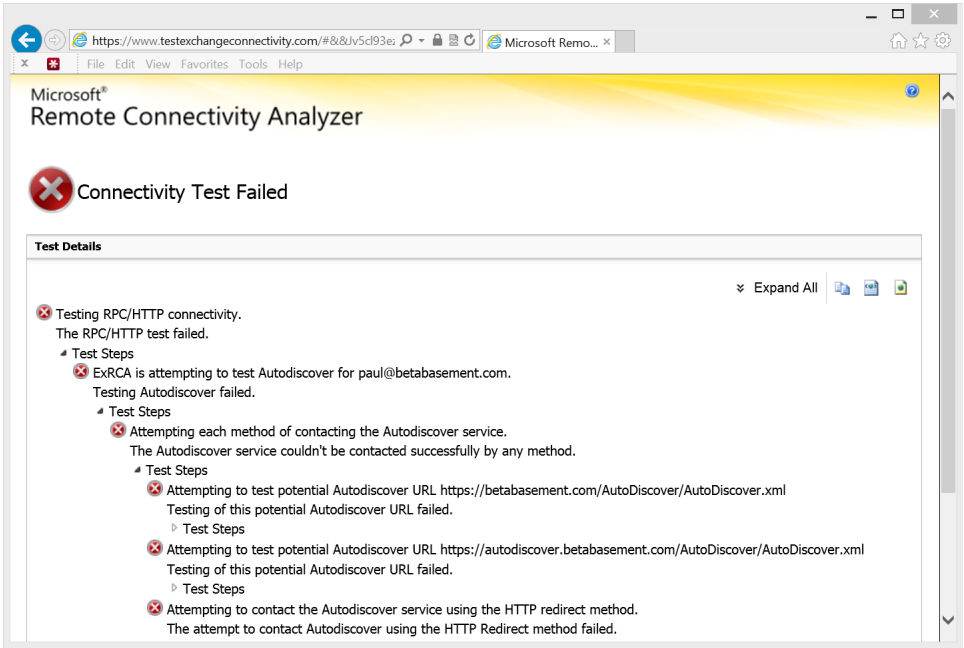

Figure 3-5 An example, intentionally bad: an ExRCA report showing multiple failures

The ExRCA tool itself is available in two versions. The online version, hosted at *http://www.testexchangeconnectivity.com*, performs tests for Outlook Anywhere, Autodiscover, EWS, and EAS connectivity. These tests can be performed against your own on-premises Exchange installation or against an Office 365 tenant. The online ExRCA also performs Lync Autodiscover and connectivity tests. There is a second, client-based version that you

download from the Clients tab of the ExRCA website and run on a local client. The local version, properly known as the Microsoft Connectivity Analyzer, is useful when you want to see why a particular client (or a client in a particular location or network) is having trouble.

Using the online version of ExRCA

The biggest thing to remember when using ExRCA online is that it requires you to provide credentials for an account in your Exchange organization. Although Microsoft is trustworthy, it is of course a bad idea to use an administrative account for this; instead, you should use an ordinary user account, ideally one that you use only for testing and that is normally disabled in Active Directory except when you're actually testing.

The first page of the ExRCA website asks you to choose a test (Figure 3-6). The Autodiscover and Outlook Anywhere tests are probably of greatest interest to most administrators, although there are other tests for various aspects of Exchange, Lync, and Office 365 connectivity.

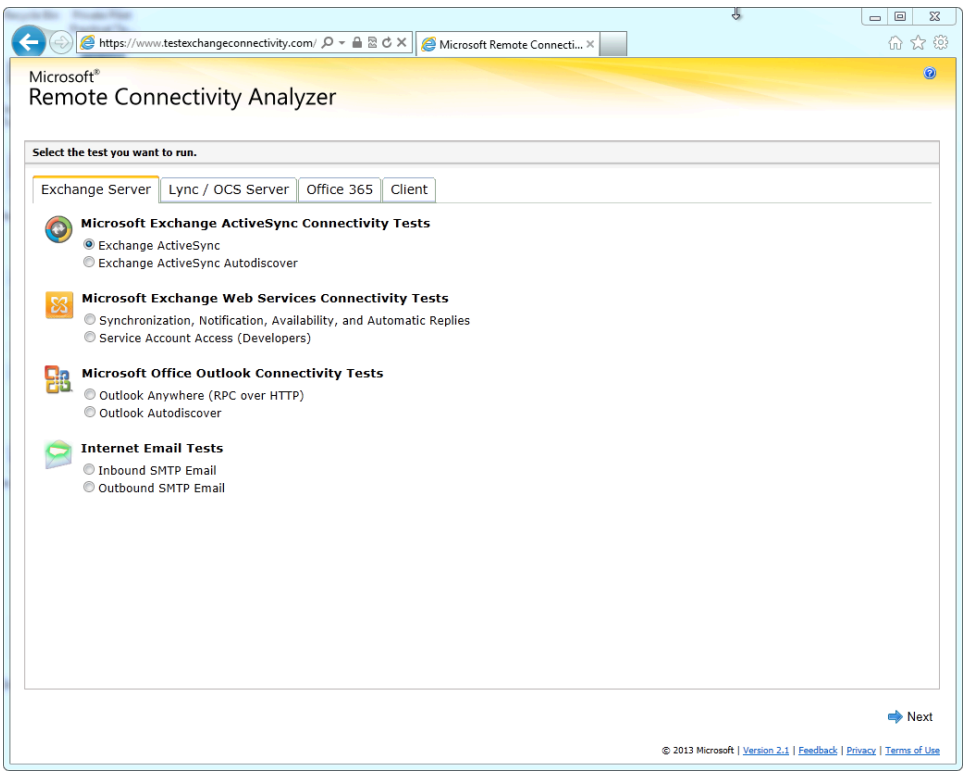

Figure 3-6 The opening page of ExRCA, requiring you to choose a test to run

When you've chosen a test type, you see a page similar to that shown in Figure 3-7. You must supply credentials for the test account you're using, and you must select a check box that indicates that you understand and accept that the working account you specify might be compromised and that you accept responsibility for it. You also have to fill out a CAPTCHA verification field; when you've done so, you can start the test by clicking the Perform Test button.

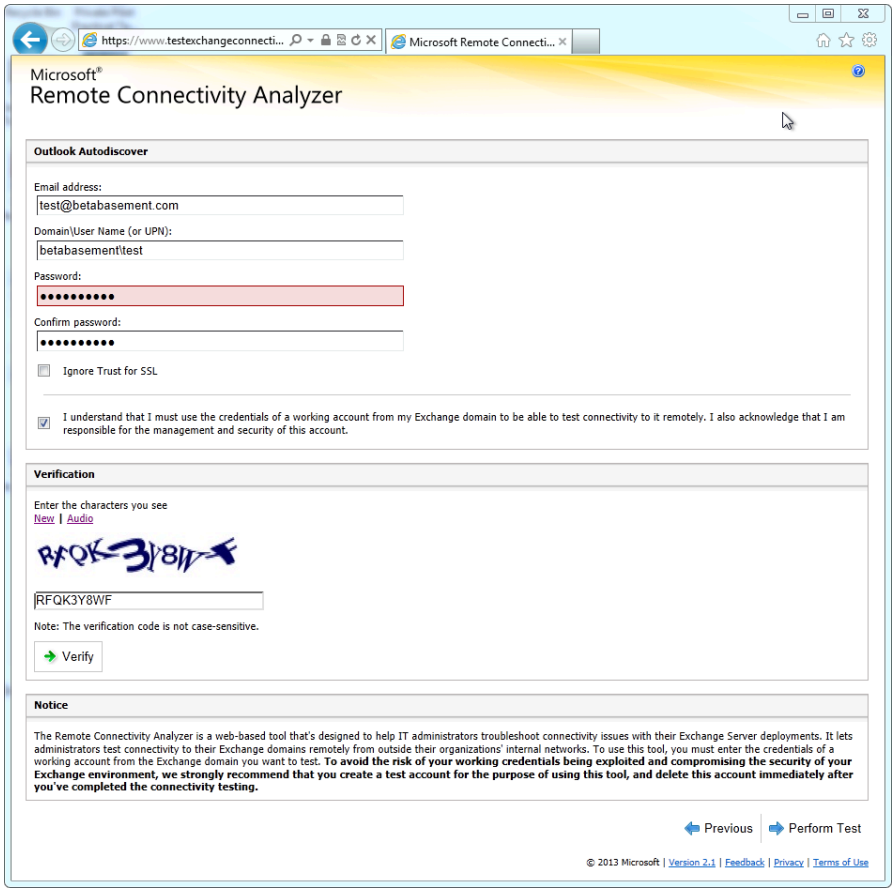

Figure 3-7 ExRCA requires logon credentials, which you must carefully safeguard

ExRCA then performs the requested test. When it's done, you get a neatly formatted report page. The report makes clear which operations succeeded or failed; failed operations usually include a link labeled "Tell me more about this issue and how to resolve it" that takes you to a page with more information on the specific failure. This is where ExRCA really shines; it's simple to run ExRCA against a new Exchange deployment (or one for which you have newly assumed responsibility) and get clear prescriptive guidance on how to fix the

issues ExRCA finds. You can save ExRCA results as XML or HTML reports if you want to, which can be handy when tracking the progress of issues over time.

Using the Microsoft Connectivity Analyzer

The Microsoft Connectivity Analyzer (MCA) is relatively new; released in early 2013, it complements ExRCA by providing a downloadable tool you can run directly from a client machine. MCA is packaged using the Microsoft ClickOnce technology, so it can easily be installed directly from a web browser even by users who don't have local administrative privileges on their computers.

To download MCA, open the Client tab on the ExRCA website and click the Microsoft Connectivity Analyzer link; the MCA installation process walks you through the actual installation. Be aware that you will need version 4.5 of the .NET Framework installed on the client PC and that (depending on your browser) you might also need to configure the browser to allow ClickOnce deployments. When you have the tool running, you see the start page shown in Figure 3-8. Note that this is providing essentially the same options as the ExRCA page, but it's formatted and worded in a much more approachable way for users who aren't Exchange administrators. Clicking any of the links on this page prompts the user for whatever other information MCA needs, including logon credentials. When you've plugged in the requested information, clicking Next starts the test.

After the test completes, you see a summary page; if any problems were found, the page typically says "Administrator Assistance May Be Required." Separate buttons enable the user to save the test results (presumably to give to the administrator) or review them himself. When you review an MCA report, you'll notice the same basic format as you see in ExRCA; each test is labeled with an icon indicating whether it passed or failed, and most tests have a disclosure triangle by which you can expand the results to see more details. MCA reports also have the Tell Me More link, which is handy when you're asked to review test results MCA has gathered but is probably less useful for end users.

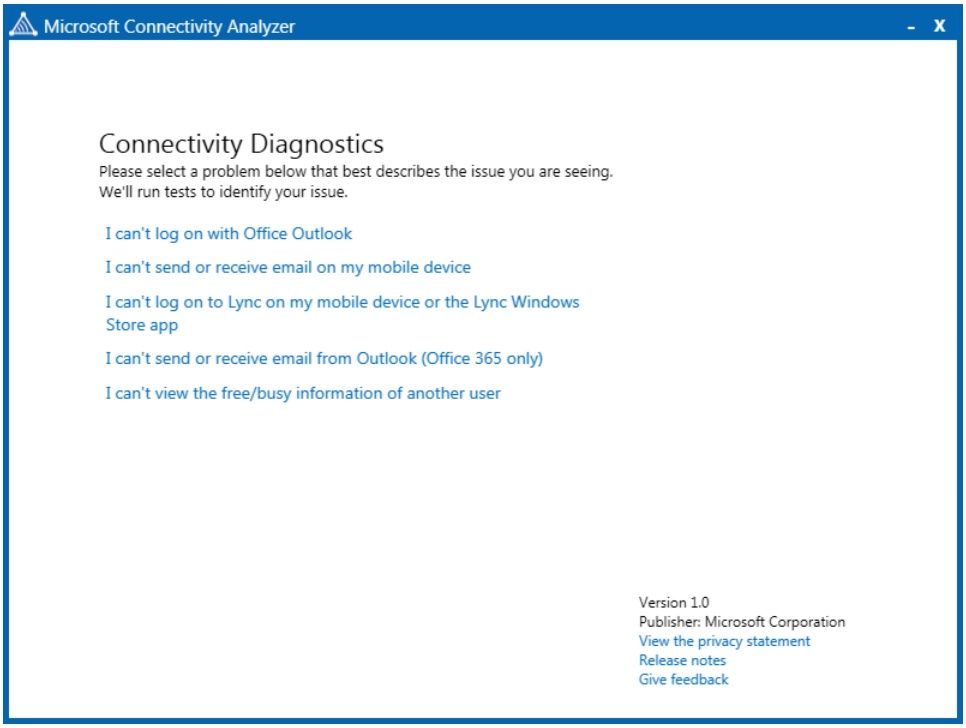

Figure 3-8 The Microsoft Connectivity Analyzer main page, a simplified rendition of the options available in ExRCA

Outlook settings and group policies

Outlook has long offered a set of Group Policy–based controls to simplify administration in large environments. You can apply these customizations by using the traditional method of attaching the Office-specific administrative templates (available from *http://technet.microsoft.com/en-us/library/cc178992.aspx*) or by using the Office Customization Tool (OCT), a topic that deserves a book of its own due to its complexity. The mechanics of applying Group Policy object (GPO) customizations to Outlook are outside the scope of this book, but it's worth mentioning some of the most interesting settings; a full list is available from *http://technet.microsoft.com/en-us/library/ff631135.aspx*.

- You can prevent Outlook from using PST files in two ways. First, you can prevent it from using PST files at all. Second, you can allow it to open existing PST files but prevent it from allowing them to grow. Both of these Group Policy settings are described more fully later in the chapter in the "Controlling PST files" section.

- You can disable Exchange Fast Access by using the HKEY_CURRENT_USER\software\policies\microsoft\office\15.0\outlook\hybrid!localcaching value, although this doesn't expose a user interface in either OCT or the GPO template file.
- You can control how many months of cached email data are synchronized with the HKEY_CURRENT_USER\software\policies\microsoft\office\15.0\outlook\hybrid!syncwindowsetting value, as described earlier in the chapter.
- By default, Outlook searches both locally cached email and email on the server; Microsoft calls this hybrid searching. If you want to restrict Outlook to performing searches on only local messages, you can set the User Configuration\Administrative Templates\Microsoft Outlook 2013\Outlook Options\Preferences\Search Options GPO option.
- Normally, Outlook helpfully tries to turn Internet URLs and Universal Naming Convention (UNC) paths into clickable hyperlinks. Some organizations try to discourage users from clicking links in email and might want to turn this feature off; to control this, use the Internet and network path into the hyperlinks setting under the User Configuration\Administrative Templates\Microsoft Outlook 2013\Outlook Options GPO.
- You can disable the display of MailTips by using the User Configuration\Administrative Templates\Microsoft Outlook 2013\Outlook Options\Preferences\Email Options GPO setting or the HKEY_CURRENT_USER\software\policies\microsoft\office\15.0\outlook\options\mail!disablemailtips value. Setting this doesn't have any effect on the display of Policy Tips for data loss prevention.

The new Office app extensibility model gets its own set of controls. All of them live in the GPO template under User Configuration\Administrative Templates\Microsoft Office 2013\Security Settings\Trust Center node:

- If you don't want your Office users using apps for Office, you can block them using the Block Apps for Office setting; there's no way to block apps only within Outlook. You either block apps from all desktop Office programs or none.
- If you want to let users run apps for Office but not install them from the Microsoft Office Store, set the Block The Office Store setting.
- There are multiple settings for controlling the activation behavior of apps; these are intended to give you a way to throttle or block apps that use too much CPU or RAM when run from within Outlook. These controls are mentioned at *http://technet.microsoft.com/en-us/library/jj219429.aspx#BKMK_Managing*. However, as of this writing, Microsoft hasn't documented specific values that might make sense for these GPO settings, so I recommend leaving them alone.

Pre-staging OST files for Outlook 2013 deployment

When a user synchronizes her mailbox with Outlook for the first time using cached Exchange mode, Outlook downloads every item in every folder in her primary mailbox and stores it in a local OST file—that's the cache to which "cached Exchange mode" refers. This can be a slow process, depending on the speed of the user's network connection and the amount of email to be downloaded; for large deployments, the network burden of having many users downloading their email at the same time can be an issue, too. To simplify the process of deploying Outlook, and to reduce the amount of network traffic, Microsoft supports a process of creating an initial OST file. You can think of this much like preseeding a DAG replica; the idea is to create a copy of the OST file over a high-speed network, save it, and then provide it to the user offline. The basic process works like this:

1. Log on to a machine running Outlook 2013, using an account that has permission to log on to the user's mailbox. This can be the user's mailbox or a separate administrative account that has the Send As and Receive As permissions on the target user.
2. Delete any existing OST files or Outlook profiles. Although this step is not strictly necessary, it reduces the chances that you'll accidentally copy the wrong OST file.
3. Create a new Outlook profile for the target user, making sure that cached Exchange mode is enabled. Set the sync slider described earlier to the appropriate value for the amount of mail you want the OST to contain.
4. Launch Outlook and wait for the OST to synchronize.
5. Quit Outlook and then move the OST file from %userdata%\Local Settings\Application Data\Microsoft\Outlook to where it is reachable from the user's computer. You could also burn it to a DVD, put it on a USB stick, and so on.

Repeat this process for each of the users for whose mailboxes you want OST files. With a bit of work, you could probably automate this process if you have a large number of mailboxes to deploy.

To stage the OST file, do the following:

1. Log on to the user's machine as the user.
2. Copy the OST file back to its home; if you accept the Outlook defaults when creating the profile, this will be %userdata%\Local Settings\Application Data\Microsoft\Outlook.
3. Create a new Outlook profile for the user, specifying the OST location you used in step 2.
4. Log on to Outlook.

When Outlook starts, it will download any changes that were made to the original mailbox contents on the Exchange server to bring the OST up to date. However, because you allowed the OST to synchronize initially, the volume of changes that have to be synced should be much smaller than that required for a full download.

Controlling PST files

PST files are an unwelcome part of many Exchange deployments. In the early days of Exchange, it made sense to let clients keep their own stores of email to help relieve the storage burden placed on the server; a typical email server might only have been able to support a few hundred 25 MB mailboxes, so letting users keep their own local stashes of mail made sense. However, the messaging world has changed in several important ways. Exchange efficiently supports mailboxes of 10 GB or even larger, and storage is cheap enough that having terabytes of space on a server is no longer uncommon. Compliance and records retention are increasingly important for many organizations, too. Given these factors, it makes sense to examine whether PST files are still useful and relevant. There are many good reasons to banish PST files from your organization, including the fact that mail stored in local workstation PSTs is essentially invisible from any compliance, security, or backup tools you have in place for your Exchange data.

If you want more information about compliance issues, see Chapter 11, "Compliance management," in *Microsoft Exchange Server 2013 Inside Out: Mailbox and High Availability* by Tony Redmond (Microsoft Press, 2013).

Assuming that you want to limit or eliminate the use of PST files in your organization, you have three choices: configure Outlook to restrict them, use the Microsoft PST Capture tool to find and ingest PST files on your network, or use a third-party tool.

Restricting use of PST files in Outlook

The Outlook Group Policy settings give you control over two aspects of how Outlook uses PST files. First, you can prevent Outlook from opening PST files at all with the *DisablePST* value (HKEY_CURRENT_USER\Software\Policies\Microsoft\Office*X*.0\Outlook\DisablePST, where *X* is the version of Outlook—for instance, 15.0 for Outlook 2013). When this value is present and set to 1, Outlook will not open any PST files that might be present in the user's Outlook profile, nor will it allow users to create new PST files or add existing PST files to a profile.

Your second option is to allow users to open their existing PSTs but not to allow those PST files to grow in size. The *PSTDisableGrow* value (HKEY_CURRENT_USER\Software\Policies\Microsoft\Office\X.0\Outlook\PST\PSTDisableGrow) prevents Outlook from allowing the PST file to grow in size; users can open their existing files and remove items from them, but users cannot add new items.

Although you can deploy these settings on individual machines, it is much more productive to use Windows Group Policy objects (GPOs) to do the work for you. Microsoft provides GPO templates for each version of Office that already incorporate the settings; you add the appropriate administrative template to the GPO that targets the desired users, enforce the settings you want, and allow the GPO mechanism to replicate the settings to target computers.

Using the Exchange PST Capture tool

After years of ceding the world of PST management to third parties, Microsoft introduced its own PST management tool, PST Capture. The function of the tool is simple: You install an agent on each of your workstations and then you install the PST Capture service itself on a server. The agents scan individual machines, looking for PST files, and then send information about the files they find to the central server.

PST Capture really involves a two-step process: discovering the PST files according to search criteria you plug in and then importing the resulting files to users' mailboxes. If the target mailboxes are within your Exchange organization, the PST Capture service sends the mail through CAS servers; if you're importing PSTs to mailboxes hosted on Office 365, the capture tool sends mail data directly to the cloud.

To use the tool, first you install the PST Capture service and console; Microsoft recommends installing them on a dedicated computer. The service requires a service account with permission to read and modify mailboxes on the Exchange servers you're using. To be more precise, the account under which you run the service needs the Exchange Organization Management role-based access control (RBAC) role.

Install the agent on computers you want to be able to scan. The agent is packaged as an MSI file that supports silent installation, so you can push it using Group Policy or other automated installation methods. The agent normally requires you to specify the PST Capture server to which you want it to talk, but you can do this from the command line like so:

```
msiexec /i PSTCaptureAgent_x86.msi /q CENTRALSERVICEHOST=deathToPSTs.contoso.com
SERVICEPORT=6674
```

The more interesting question is really where you install the agent. Obviously, you want it on machines that are likely to have PST files. That might include desktop or laptop computers the organization owns, file servers, or network-attached storage (NAS) devices. (Although NAS devices can be scanned with the console, they can't normally run the agent.) You have to trade off the work required to install (and, later, remove) the agent on the computers in your organization versus the benefit of identifying and getting rid of PST files, bearing in mind that users with non-domain-joined machines running Outlook Anywhere might still have PST files stashed away on laptops or desktops at home.

After you install agents, you can perform a search. You can control which computers are included in the search and specific folders you want included or excluded in the search. For example, you can exclude the Windows system directory with a single check box. You can schedule the search or perform it immediately.

When you've completed a search or supplied your own manual import list containing the names and paths of the PSTs you want to import, you can start an import operation. You can import PSTs to a specific folder in the target user's mailbox, or you can dump the contents in the Inbox. For example, say the PST has a top-level folder named Old Mail with subfolders named 2011 and 2012. If you select the Inbox option, the user sees a new folder named Old Mail as a subfolder of the Inbox. If you select a specific target folder, the Old Mail folder will be created as a child of that folder. You can also specify that you want PST items imported into the user's archive mailbox, presuming that he has one. In either case, you must use the PST Capture console to link each PST file with the mailbox to which it should be imported. Because the console shows you the computer name and local path of each PST it finds, it should be simple to identify which PSTs belong to which users, but specifying that in the console can be a time-consuming operation.

For more details on the care and feeding of the PST Capture tool, see the Microsoft documentation at *http://technet.microsoft.com/en-us/library/hh781033(v=exchg.141).aspx*.

Exploring third-party solutions

When Microsoft delivered the initial version of the PST Capture tool, it entered a market that already contained a number of solutions for identifying and importing PST files, including PST Attender from Sherpa Software, the Migrator product from TransVault, and others. As with many other Microsoft add-ons, the PST Capture tool does much but not all the work third-party tools do. You have to decide whether the additional features (which might include better reporting, more flexible scheduling, or a wider range of configuration options) justify the expense of buying a tool you might not need to run on an ongoing basis.

Blocking client connections to a mailbox

Exchange enables you to disable any or all client connection protocols, including MAPI, on a per-user basis. You might need to do this if you want to prevent users from running earlier versions of Outlook that don't support features you need (such as messaging records management features like managed folders) or because the users in question are required to use Outlook Web App.

Whereas Exchange 2007 and Exchange 2010 required you to make this change through the Mailbox Features tab of the user's account properties, in Exchange 2013, you can do it

either through Exchange Management Shell (EMS) (by using Set-CASMailbox) or the mailbox features tab of the user's mailbox properties dialog box in Exchange Administration Center (EAC) (Figure 3-9). If you disable MAPI for a user, that user cannot use Outlook to connect to her mailbox.

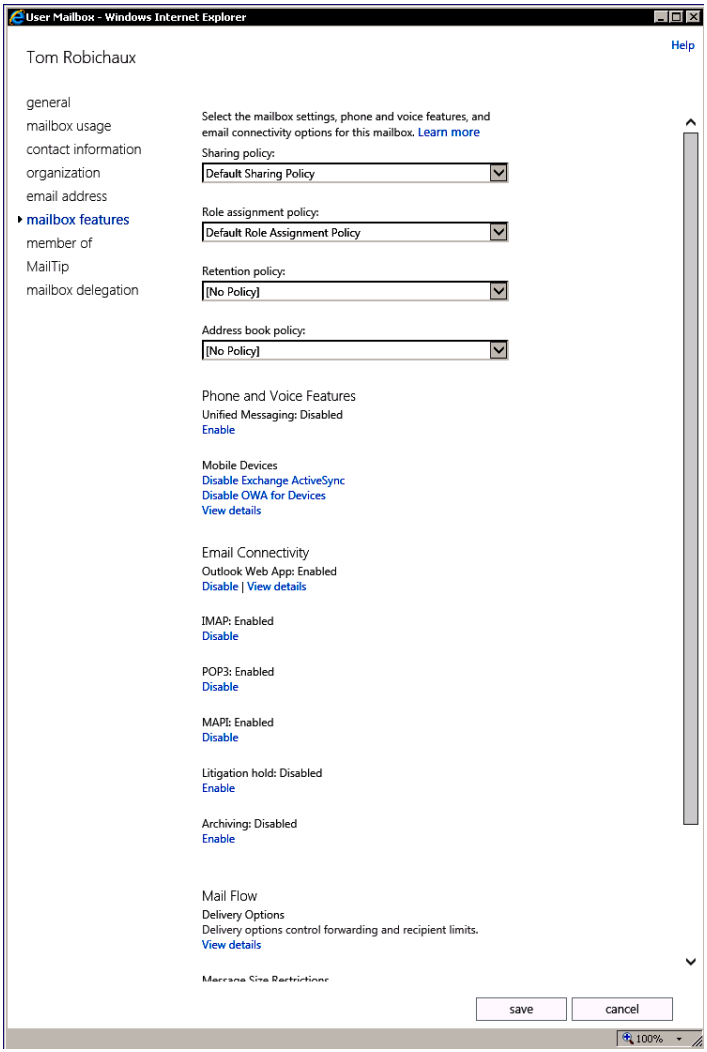

Figure 3-9 Disabling MAPI access for a user

The Set-CASMailbox cmdlet supports a number of parameters to control how an individual mailbox can use MAPI to connect to a mailbox on an Exchange server:

- **MAPIBlockOutlookRpcHTTP** Enables you to determine whether you allow Outlook clients to connect over RPC over HTTP through Outlook Anywhere. Set the parameter to $True to block RPC over HTTP access and $False to allow access.
- **MAPIBlockOutlookVersions** Enables you to control which versions of Outlook can connect to Exchange. You might use this setting to force users to upgrade to a more modern version of Outlook by blocking Outlook 2007. If a user attempts to use a blocked version of Outlook, he will see the error message shown in Figure 3-10. Outlook clients configured for cached Exchange mode continue to work offline, but they cannot connect to the server until an administrator lifts the block.

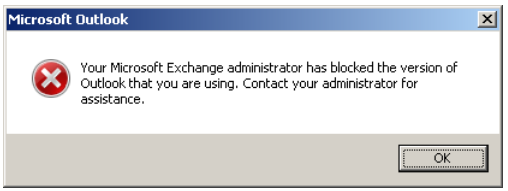

Figure 3-10 A user discovers that he can't use this version of Outlook to connect to Exchange

- **MAPIBlockOutlookNonCachedMode** Enables you to determine whether you allow Outlook clients to connect in online mode to the server. Set this parameter to $True to allow online access or to $False to force clients to connect in cached Exchange mode. Somewhat confusingly, users blocked from online access see the same message shown in Figure 3-10, followed by another error message to tell them that Outlook is unable to open their default email folders. Pointing to the version of Outlook rather than the need to use cached Exchange mode might confuse the help desk when users report their problem.

Microsoft identifies Outlook builds using a scheme of major release, minor release, build number. The major release number is shared across all the Office applications. The Office 14 suite includes Outlook 2010, Office 15 includes Outlook 2013, and so on. (Microsoft did not produce an Office 13 suite.) The minor release indicates whether the build is in the original RTM build, a service pack, or a cumulative update, and the build number is incremented daily to include code and fixes checked in by engineers. Here are the build numbers for some example Outlook versions:

- Outlook 2007: 12.4518.1014
- Outlook 2007 SP1: 12.6425.1000

- Outlook 2010: 14.0.4760.1000
- Outlook 2013 RTM: 15.0.4481.1003

To discover the client version to specify in the MAPIBlockOutlookVersions parameter, you can look at the properties of Outlook.exe itself, use the Programs control panel, or check with the useful list of client versions that Microsoft maintains at *http://technet.microsoft.com/en-us/library/aa996848.aspx*.

Before you start blocking any particular version of Outlook on Exchange 2013, you might want to know which users are currently using which versions to connect to your existing Exchange infrastructure. On Exchange 2007 and Exchange 2013, this command will give you a CSV file listing each user who's running Outlook along with the version of Outlook she's using. This helps simplify decisions about which versions to block on which mailboxes:

```
Get-MailboxServer | Get-LogonStatistics | Select
UserName,ClientName,ClientVersion,LogonTime | 'Export-Csv
-Path ExchangeClientVersions.csv
```

When you know which versions you want to block, the next step is to construct a version string that does the trick when passed to Set-CASMailbox. A single version by itself blocks only that version. A range of two versions blocks those versions and any in between. Specifying a single version blocks all versions either before or after that version, depending on where you put it. You can combine multiple version strings by separating them with semicolons. A few examples might help make this clearer:

- Set-CASMailbox –MAPIBlockOutlookVersions "12.0.6504.5000" blocks Outlook 2007 SP2 only. Any other version can connect.
- Set-CASMailbox –MAPIBlockOutlookVersions "12.0.4518.1014-14.0.6023.1000" blocks Outlook 2007 SP2 only and all versions up to and including Outlook 2010 SP1. Any earlier or later version can connect.
- Set-CASMailbox –MAPIBlockOutlookVersions "-15.0.4128.1014" blocks any version earlier than the public beta of Outlook 2013, including the public beta itself. Any later version can connect.
- Set-CASMailbox –MAPIBlockOutlookVersions "-15.0.4481.1003" blocks any client version later than Outlook 2013 RTM. Any earlier version can connect.

When you're blocking, you should always include an explicit allow for version 6.0.0 to support Exchange server-side MAPI connections (server connections always use MAPI version 6.0), like this:

```
Set-CASMailbox –Identity 'Simpson, Katherine' –MAPIBlockOutlookVersions '-6.0.0;
-15.0.4128.1014'
```

You have to wait up to 120 minutes for the cached information about the mailbox to expire from the Store's cache. Alternatively, you can restart the Information Store service, but apart from test situations, this is definitely not the best approach because it will affect all the mailboxes connected to the server.

You can check whether any restrictions are in place for any protocols on a server by using the Get-Mailbox cmdlet to examine the ProtocolSettings property of each mailbox. If a restriction is in place for a specific client version, you see that version number listed. If an administrator has completely disabled MAPI access for the mailbox, you see "MAPI" and no version number. For example:

```
Get-Mailbox –Server PAO-EX01 | Where {$_.ProtocolSettings –ne $Null} | Select Name,
ProtocolSettings
```

```
Name                          ProtocolSettings
--------                      ----------------
Cannon, Paul                  {MAPI§§§§-6.0.0;10.0.0-11.5603.0§§§§}
Kol, Ayla                     {MAPI§0§§§§§§§}
Simpson, Katherine            {OWA§1, IMAP4§0§§§§§§§§, POP3§0§§§§§...
```

You can also use the Get-CASMailbox cmdlet to check for MAPI blocks. Get-CASMailbox is more interesting because it also allows you to return the value of the MAPIEnabled property (False if the user is completely blocked from using MAPI) and to see the details of all the protocol settings you can set on a mailbox. However, you cannot specify a server name to check against, so Get-CASMailbox is less efficient because it will scan the entire organization unless you restrict its scope by using a server-side filter to focus on one server:

```
Get-CASMailbox –Filter {ServerName –eq'ExchServer1'} | Where {$_.ProtocolSettings
–ne $Null} | Select Name, ProtocolSettings, MapiEnabled
```

```
Name                    ProtocolSettings                MAPIEnabled
--------                ----------------                -----------
Cannon, Paul            {MAPI§§§§-6.0.0;10.0.0-...      True
Kol, Ayla               {MAPI§0§§§§§§§}                 False
Simpson, Katherine      {OWA§1, IMAP4§0§§§§§§§§...      True
```

In addition to imposing blocks on MAPI connections, you can use the Set-CASMailbox cmdlet to disable client access to other protocols. For example:

- **To disable access to POP3** Set-CASMailbox –Identity Bond –PopEnabled $False
- **To disable access to IMAP** Set-CASMailbox –Identity Bond –ImapEnabled $False

- **To disable access to Outlook Web Access** Set-CASMailbox –Identity Bond –OWAEnabled $False
- **To disable user access using Outlook Web App for Devices** Set-CASMailbox –Identity Bond –OWAforDevicesEnabled $False. Note that this setting might not do anything useful when run against an on-premises mailbox because Outlook Web App for Devices is only officially supported against Office 365 mailboxes.
- **To disable access to ActiveSync** Set-CASMailbox –Identity Bond –ActiveSyncEnabled $False

Blocking client access to a Mailbox server

Implementing blocks on a mailbox basis is useful, but sometimes you want to block all access to a Mailbox server. For example, you might want to update the server with some software or apply and update without having users impose load on the server or potentially interfere with the upgrade. One of the many advantages to the Exchange Database Availability Group (DAG) architecture is the way it simplifies maintenance; you can put a DAG member into maintenance mode, work on it, and then bring it online again. This process, which is described fully in Chapter 9, "The Database Availability Group," in *Microsoft Exchange Server 2013 Inside Out: Mailbox and High Availability*, means that you'll probably never have to block client access to a DAG member server explicitly. What if you have a standalone server, though?

You could apply such a block with EMS by searching for all mailboxes hosted in active databases on the server and using the Set-CASMailbox cmdlet to disable MAPI access, but it is more convenient to be able to apply the block centrally. For all versions from Exchange 2000 to Exchange 2007, you could block MAPI clients from connecting to a Mailbox server by configuring the Disable MAPI Clients key in the registry. This key is intended to enable administrators to require the deployment of a base-level version of Outlook. Put another way, it stops users from attempting to connect with earlier versions that might not meet your company's security requirements because the earlier software doesn't include recent anti-spam and antivirus features such as beacon blocking.

This registry key doesn't work on Exchange 2010 or Exchange 2013. For those versions, if you want to block MAPI connections on a particular CAS, you must take another tack. Choose from two approaches if you need to block connections to a Mailbox server.

- Use the Set-RPCClientAccess cmdlet. This cmdlet allows you to block all MAPI connections coming from specific versions. For example, this command blocks access to any version of Outlook prior to Outlook 2007 (major release 12).

```
Set-RPCClientAccess -Server ExCAS01 -BlockedClientVersions
"0.0.0-5.65535.65535; 7.0.0-11.99999.99999"
```

The problem is that all connections to all Mailbox servers supported by the CAS server will be blocked. This might be an effective method to use on small sites that have just one CAS server and one Mailbox server.

- On larger sites that support multiple CAS and Mailbox servers, you can set a per-mailbox block with the Set-CASMailbox cmdlet for every mailbox on the server that you want to maintain. For example:

```
Get-Mailbox –Server PAO-EX01 | Set-CASMailbox –MAPIBlockOutlookVersions
'-6.0.0;10.0.0-12.4406.0'
```

Both mechanisms are equally effective as a block. The choice between the two therefore comes down to whether you can block all connections flowing through a CAS server no matter what Mailbox server they are destined for, or you need to block connections to just one specific Mailbox server.

Using the Office Configuration Analyzer Tool

Outlook for Windows is a complex program that's evolved over a long period of time. Despite the fact that parts of it have been completely rewritten over time, there are still occasions when a particular machine (or group of machines, depending on whether you use imaging or cloning) might not behave the way you expect. Microsoft used to provide a free tool known as the Outlook Configuration Analyzer Tool, or OCAT, to help diagnose problems with Outlook installations. For Office 2013, Microsoft replaced OCAT with OffCAT, the Office Configuration Analyzer Tool. The purpose of OffCAT is to analyze the configuration and installation of the Office applications on a given machine and report back on any problems, real or potential, that exist. OffCAT is similar in spirit to the Exchange Best Practices Analyzer (ExBPA) and other tools that look at a static configuration and check it against a set of best practices defined by one of Microsoft's product teams.

OffCAT is available from Microsoft's website at *http://www.microsoft.com/en-us/download/details.aspx?id=36852*. It's packaged in a number of ways, including as a ZIP file containing the application and its support files or as a Windows Installer file that actually installs the application for you. No matter how you get it onto the target system, once you launch it, you'll be in familiar territory because it works very similarly to ExBPA. When you launch OffCAT, it asks you if you want it to check for updates to its rule base, which is packaged as a separate downloadable XML file so that updates to the rules don't require updates to the application itself. After you've installed updates, you see a page that lists all of the Office programs from Access to Word; each program's name is a link that takes you to a page like that shown in Figure 3-11.

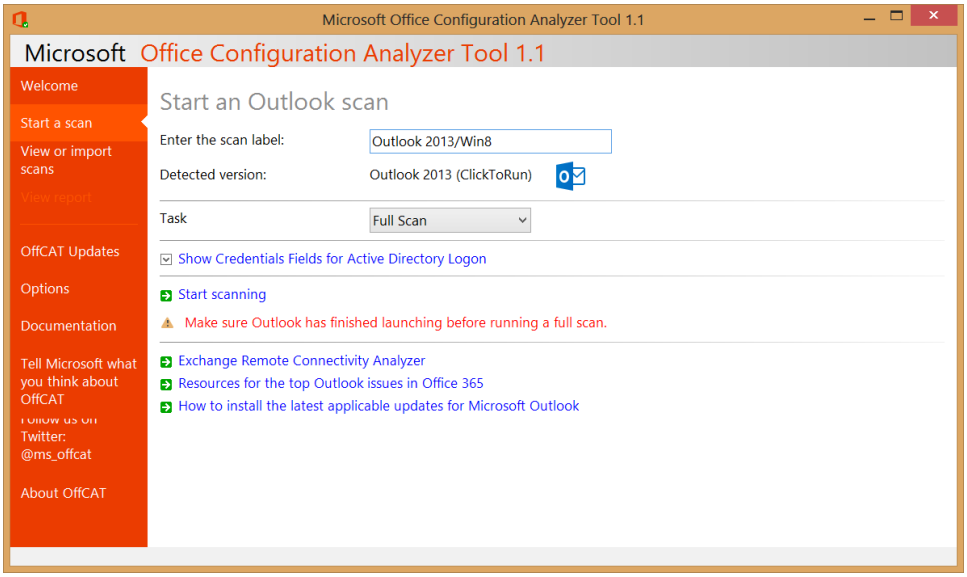

Figure 3-11 Starting an Outlook scan

Once you start the scan, the OffCAT display changes to reflect which specific scan tasks are being executed. In Figure 3-12, the tasks include a check of the current Outlook profile (helpfully named "Outlook") for consistency and possible corruption and checks of the folders in both the primary mailbox and personal archive mailbox for the user who is currently logged on. OffCAT performs several dozen checks of various configuration and data items, including some related to the Outlook configuration on the local machine for the current user; some related to the Outlook installation and configuration; and some related to folders, items, and other structures inside the current user's mailbox. For this reason, you must have Outlook running to perform an OffCAT scan, and if you quit Outlook (or it crashes) during the scan, the results you get might be incomplete or even unusable.

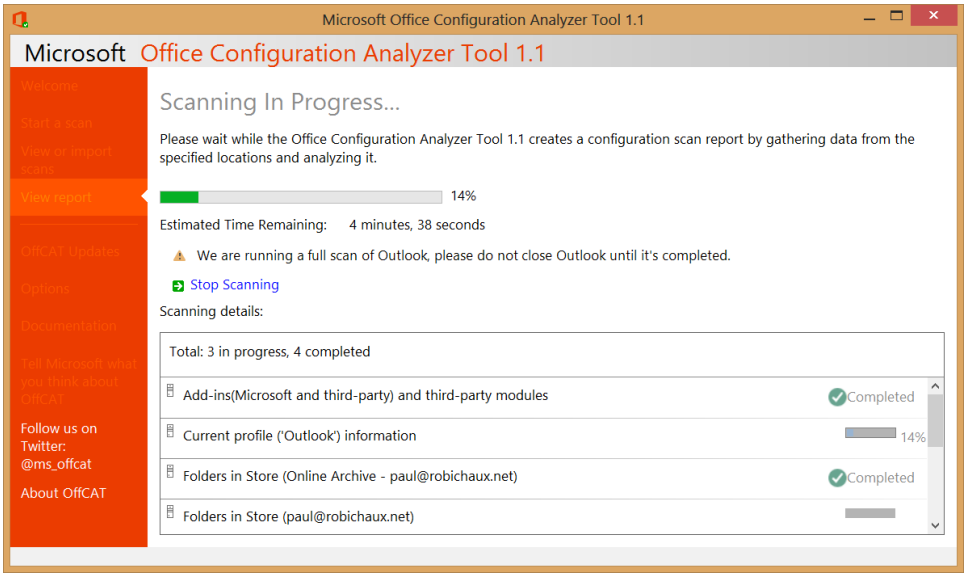

Figure 3-12 An OffCAT scan of Outlook 2013 in progress

The amount of time required to run the scan varies according to the speed of the computer being scanned, the amount of data in the user's mailbox, and a number of other factors. Having said that, a typical scan of Outlook alone will normally take less than five minutes. At the end of the scan, you'll see a results report (Figure 3-13) that is reminiscent of the results shown by various other Microsoft configuration analysis tools. In this case, the scan results include a note that there were items found in the Sync Issues folder, that mysterious synchronization problem report repository that often fills with items for no apparent reason. Other errors found by the scan, but not visible in the figure, include several problems related to a corrupt calendar item from 2008. Each problem reported includes links to Microsoft documentation or Microsoft Knowledge Base articles that propose corrective action for the reported problem.

You can also run OffCAT from the command line, meaning that you can push it to client systems and run it with a logon script or as part of a GPO. Although it might not be necessary to periodically scan all your client systems for problems related to Office configuration, it would certainly be a good idea to scan machines on which users have reported problems, and it might be worthwhile to scan your clients as part of your preparations for upgrading to a new release of Office.

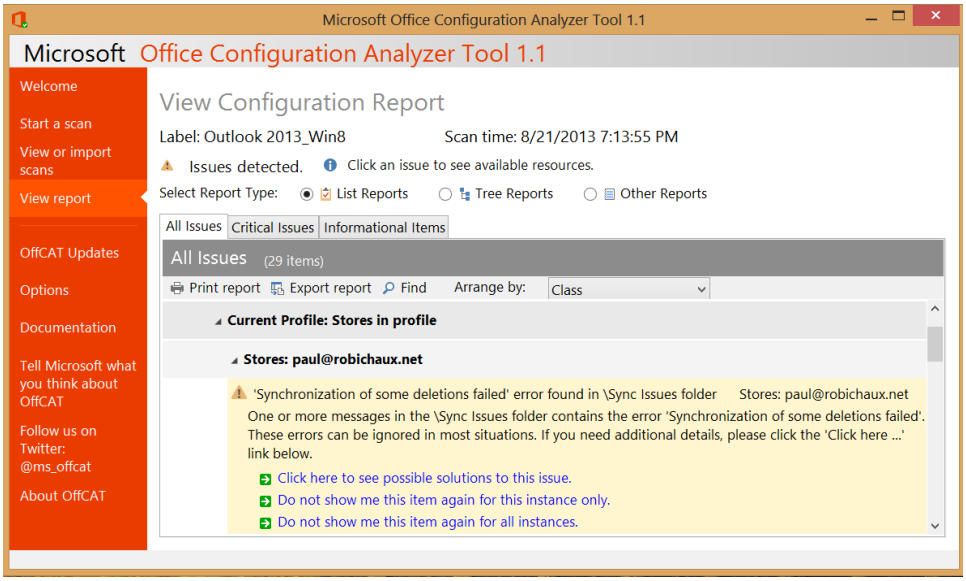

Figure 3-13 The results of an OffCAT scan showing problems with data in the scanned mailbox

Managing Outlook Web App

Users don't typically think of Outlook Web App as an application on its own; they think, "Oh, it's the webpage I use to get email." Exchange administrators, however, know quite well that Outlook Web App is a separate application that runs under Internet Information Server (IIS); its role is to retrieve user mail data, display it, and interact with the user. In Exchange 2013, Outlook Web App only runs on Mailbox servers, so some of the ways we interact with and manage it as administrators have changed from previous versions. The primary vehicle for changing Outlook Web App settings is the Outlook Web App virtual directory created during installation of the Mailbox role; you will become quite adept at using Set-OWAVirtualDirectory and Set-OWAMailboxPolicy to control the various options in Outlook Web App.

Outlook Web App mailbox policies

Exchange 2013 supports the ability to allocate different levels of functionality to Outlook Web App users through policies. As with the other types of policies Exchange supports, Outlook Web App policies are intended to enable you to create a group of settings and then apply those settings to mailboxes without having to modify the individual mailboxes. Microsoft added Outlook Web App policies in Exchange 2010 to give administrators a more granular way to control access to Outlook Web App features. All the features Outlook

Web App policies can control can also be controlled by changing settings on an individual Mailbox server, and many of them can be modified by changing settings on user mailboxes.

Exchange includes a default Outlook Web App policy, but that default isn't applied to any mailboxes unless you manually do so. You can create as many Outlook Web App mailbox policies as you like and then apply a maximum of one Outlook Web App policy to each mailbox. If you don't apply any policies to a mailbox, a user's access to Outlook Web App features is controlled by the segmentation properties defined for the Outlook Web App virtual directory on each CAS server. Figure 3-14 shows the EAC view for Outlook Web App mailbox policies, which you use to create and modify policies. EAC helpfully shows you a summary of the currently selected policy's settings on the right side of the window.

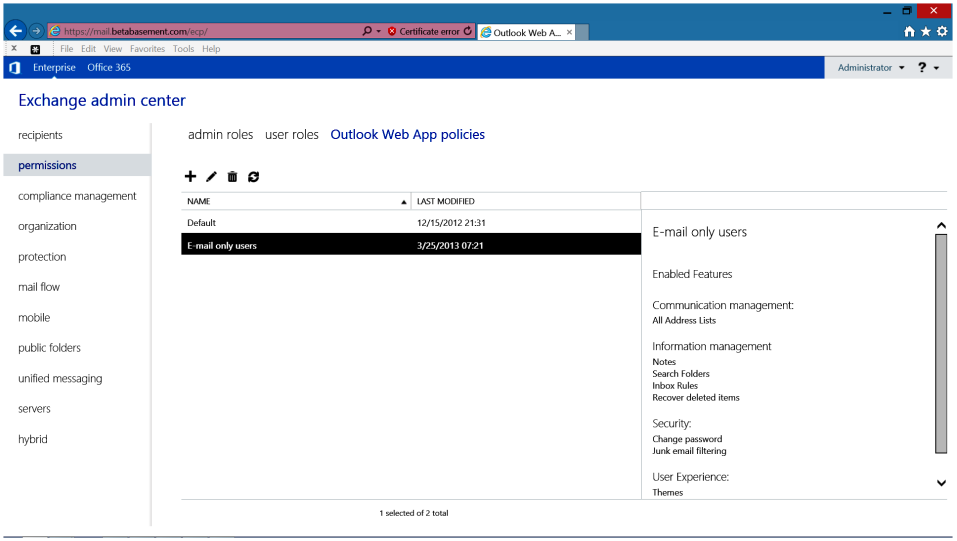

Figure 3-14 Viewing the list of Outlook Web App policies in EAC

The easiest way to apply any mailbox Outlook Web App policy, including the default policy, to a set of mailboxes is with the Set-CASMailbox cmdlet. For example, this command fetches all the mailboxes that belong to the North America organizational unit (OU) and pipes them to Set-CASMailbox to apply the default Outlook Web App mailbox policy:

```
Get-Mailbox -OrganizationalUnit 'North America'| Set-CASMailbox
-OwaMailboxPolicy 'Default'
```

What should you put in the default Outlook Web App policy? It depends. The default Outlook Web App policy included with Exchange basically duplicates the default out-of-the-box segmentation properties of the Outlook Web App virtual directory as it's installed

on a Mailbox server. It permits access to all Outlook Web App features, including the premium client.

To create a new Outlook Web App mailbox policy, open the Permissions section of EAC, select the Outlook Web App policies tab, and click the plus (+) icon. A wizard then enables you to select which features you want users to access (Figure 3-15). You can choose any or all of the available features in the policy, several of which were added in Exchange 2013 CU2.

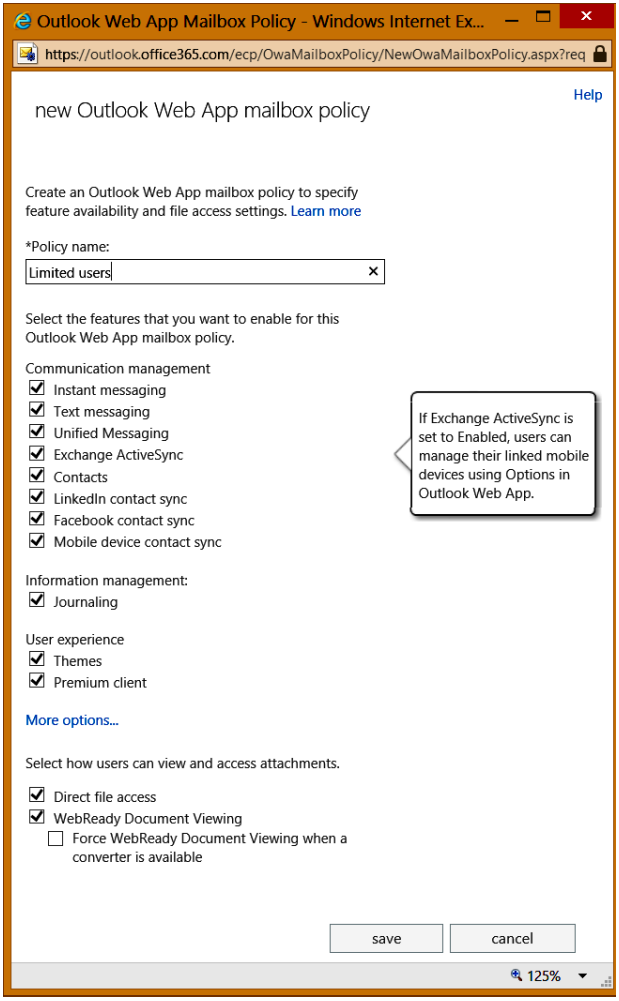

Figure 3-15 Creating a new Outlook Web App mailbox policy

Table 3-3 lists the features shown on the Features tab of the EAC dialog box for an Outlook Web App mailbox policy. Some of these features depend on other components (text messaging, public folders, and instant messaging, for example), and others require a very good reason before you disable them. For example, it usually doesn't make much sense to disable the Change Password feature because handling user requests to change passwords creates extra work for help desks.

TABLE 3-3 Outlook Web App features controllable through Outlook Web App policies

Feature	Meaning	Available through
Instant messaging	If enabled, and if you've configured Lync properly, users can access IM functionality from within Outlook Web App, including the ability to view presence information. If disabled, these features are unavailable.	Outlook Web App
Text messaging	If enabled, users can create and send text (SMS) messages from Outlook Web App. If disabled, this feature is removed.	Outlook Web App
Unified messaging	If this feature is enabled and the mailbox is enabled for UM, users can access and manage their UM settings through Exchange Control Panel (ECP). If disabled, the option is removed.	ECP
Exchange ActiveSync	If enabled, users can access details of the mobile devices they have synchronized, including the ability to wipe devices if they are lost and retrieve logs containing details of synchronization operations. If disabled, the option is removed from ECP.	ECP
Contacts	If enabled, users can access their Contacts folder in Outlook Web App. If disabled, the icon is removed from Outlook Web App.	Outlook Web App
LinkedIn contact sync	Controls whether Office 365 users are allowed to synchronize their LinkedIn contacts with their Exchange contacts folder.	Outlook Web App
Facebook contact sync	Controls whether Office 365 users are allowed to synchronize their Facebook contacts with their Exchange contacts folder.	Outlook Web App
Mobile device contact sync	Controls whether users running Outlook Web App for Devices are allowed to sync their Exchange contacts to the device using the app.	Outlook Web App for Devices
All Address Lists	If enabled, users can see all defined address lists in the directory. If disabled, they can see the Global Address List (GAL) only.	Outlook Web App

Feature	Meaning	Available through
Public Folders	If enabled, users can access and work with public folders. If disabled, the icon is removed from Outlook Web App.	Outlook Web App
Journaling	If enabled, users can see the Journal folder in their folder list. If disabled, Outlook Web App hides the folder.	Outlook Web App
Notes	If enabled, users can see and modify note items in Outlook Web App. If disabled, Outlook Web App hides the Notes icon.	Outlook Web App
Search Folders	If enabled, users can access search folders created by Outlook. If disabled, these folders are suppressed.	Outlook Web App
Inbox rules	If enabled, users can create and modify rules through ECP. If disabled, the option is suppressed. However, Exchange continues to respect any rules created with Outlook.	ECP
Recover Deleted Items	If enabled, users can recover deleted items. If disabled, users cannot recover deleted items with Outlook Web App, but Exchange will continue to preserve these items in the Recoverable Items folder.	Outlook Web App
Change password	If this feature is enabled, users can change their account password from Outlook Web App. If disabled, Outlook Web App will not prompt users when their password is approaching its expiry date (prompts start 14 days in advance), and they cannot see the option to change their password in ECP.	Outlook Web App /ECP
Junk email filtering	If enabled, users can access the options to control junk mail processing such as blocked and safe user lists. If disabled, the option is removed from ECP.	ECP
Themes	If enabled, users can select a theme other than the default and apply it to Outlook Web App and ECP. If disabled, the option is suppressed.	Outlook Web App /ECP
Premium client	If enabled, users can use the premium client with a browser that supports this client. If disabled, users are forced to use the standard client no matter what browser they use.	Outlook Web App
Email signature	If enabled, users can access the option to create or modify email signatures and apply them to outgoing messages. If disabled, the option is removed from ECP.	ECP

Feature	Meaning	Available through
Calendar	If enabled, users can access the Calendar application. If disabled, the icon is removed from Outlook Web App.	Outlook Web App
Tasks	If enabled, users can create and manage tasks in Outlook Web App. If disabled, the option is suppressed.	Outlook Web App
Reminders and notifications	If enabled, Outlook Web App provides users with notifications of new messages, meeting reminders, and so on. If disabled, these notifications are suppressed.	Outlook Web App

Managing Outlook Web App mailbox policies in EMS

A new policy can also be created with EMS. For some odd reason, this is a two-step process. First, you create the new policy with the New-OWAMailboxPolicy cmdlet, and then you use the Set-OWAMailboxPolicy cmdlet to define which features are enabled or disabled by the policy. For example, here's a policy that allows users to use the premium client while removing some of the more esoteric features:

```
New-OWAMailboxPolicy -Name 'Limited OWA features'
Set-OWAMailboxPolicy -Identity 'Limited OWA features'
-ActiveSyncIntegrationEnabled $True -AllAddressListsEnabled $True
-CalendarEnabled $True -ContactsEnabled $True -JournalEnabled $True
-JunkEmailEnabled $True -RemindersAndNotificationsEnabled $True
-NotesEnabled $True -PremiumClientEnabled $True -SearchFoldersEnabled $False
-SignaturesEnabled $True -SpellCheckerEnabled $True -TasksEnabled $True
-ThemeSelectionEnabled $False -UMIntegrationEnabled $False
-ChangePasswordEnabled $True -RulesEnabled $True -PublicFoldersEnabled $False
-SMimeEnabled $True -RecoverDeletedItemsEnabled $True
-InstantMessagingEnabled $False -TextMessagingEnabled $False
```

There are a number of Outlook Web App mailbox policy settings that are only available through EMS, too:

- DefaultTheme enables you to specify the name of an Outlook Web App theme that users receive by default, for instance, Set-OWAMailboxPolicy –DefaultTheme "Orange". The only way I've found to get a list of theme names is to look at the version-specific folder under V15\ClientAccess\OWA; for example, on an RTM Exchange 2013 server, the theme folders will be in V15\ClientAccess\OWA\15.0.516.30\Owa2\resources\themes.

- DelegateAccessEnabled controls whether users who have delegate access to another mailbox can open the other mailbox in Outlook Web App. Users who are delegates can still open whatever mailboxes they have access to from Outlook. In a similar vein,

ExplicitLogonEnabled controls whether a user is allowed to open another user's mailbox from within Outlook Web App without logging out and logging back in as the target user.

- DisplayPhotosEnabled, SetPhotoEnabled, and SetPhotoURL affect whether and how sender photos are displayed within Outlook Web App; SetPhotoEnabled governs whether users can set their own photo from within Outlook Web App. If they cannot, Outlook Web App attempts to use photos from Active Directory if they are present and DisplayPhotosEnabled is set to $True. SetPhotoURL is the URL users are sent to when they attempt to change their photo from within Outlook Web App; it is often set to a SharePoint page.
- IRMEnabled controls whether Outlook Web App allows users to read or send messages that have been protected with Active Directory Rights Management Services (AD RMS).
- PredictedActionsEnabled enables an Outlook Web App feature that is supposed to customize the commands and icons the user sees according to what they're doing. This is an intriguing idea (much like adaptive menus in Office 2003), but it's not widely used in Outlook Web App, and it's not clear yet whether users are even aware that it exists.
- OrganizationEnabled turns off some organization-level settings. For example, when this is set to $False, users don't see separate options for internal and external out of office (OOF) messages, and the Resources tab on calendar items is hidden.

Applying an Outlook Web App mailbox policy

After you have created an Outlook Web App mailbox policy, you can use either EAC or EMS to apply the new policy. EMS is simple; you use Set-CASMailbox with the –OWAMailboxPolicy switch. For example, to apply the policy defined earlier, you could do the following:

```
Set-CASMailbox –Identity 'Andrews, Ben (IT)' –OWAMailboxPolicy 'Limited OWA Features'
```

If you'd rather use EAC, you can apply the policy by selecting the user on the Recipients page of EAC, opening its properties dialog box, and clicking the View Details link beneath the Outlook Web App label. That displays the dialog box shown in Figure 3-16; use the Browse button to select the policy you want to apply.

Exchange enforces the new policy the next time the user logs on to her mailbox.

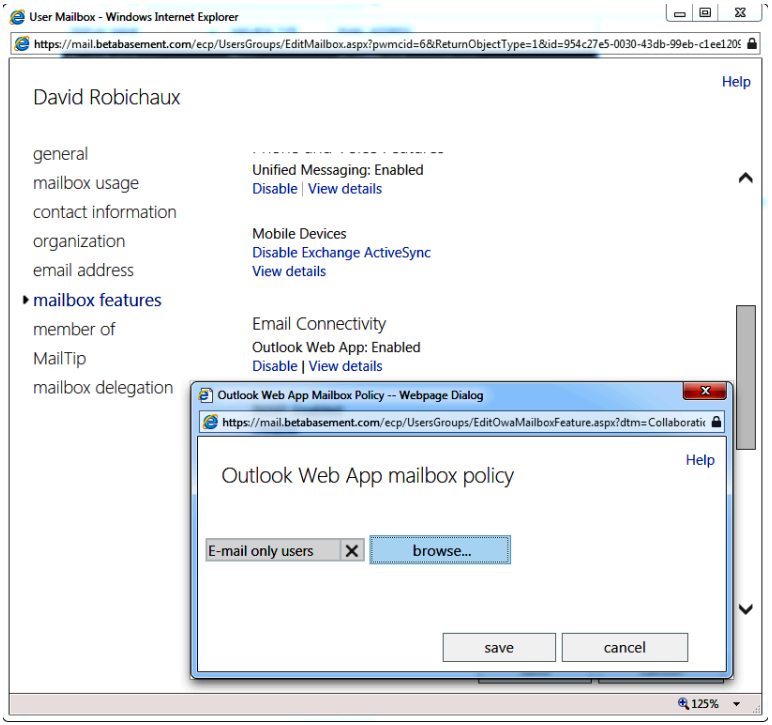

Figure 3-16 Selecting an Outlook Web App mailbox policy for a user

Controlling offline Outlook Web App use

Both EAC and EMS enable you to control whether users may use Outlook Web App in offline mode. This isn't a simple on/off toggle, though. By default, any user who has an offline-compatible browser is allowed to turn offline mode on. That's because the default value of the AllowOfflineOn setting in the default policy is AllComputers. Outlook Web App 2010 and earlier versions allowed users to specify whether the computer from which they were logging on should be treated as a public or private computer. Outlook Web App 2013 doesn't give users this choice by default. If you want to restrict offline use to private computers, you can, but it's a two-step process. First, you use Set-OWAMailboxPolicy –AllowOfflineOn PrivateComputersOnly and then you enable the private/public computer choice by running Set-OWAVirtualDirectory –LogonPagePublicPrivateSelectionEnabled.

If you want to prevent offline access for some reason such as the fear that users will accidentally leave important or sensitive data on a public computer, set the value of AllowOfflineOn for the target policy to NoComputers and then apply the policy as desired. Remember, the policy won't be applied until the next time each user logs on.

Of course, there's another aspect to controlling offline use of Outlook Web App: the client browser itself has to be configured to allow offline access. In Internet Explorer, this takes the form of the Allow Website Caches And Databases setting (Figure 3-17), which must be enabled for the browser to actually cache any data. Although this setting is enabled by default in Internet Explorer 10 on Windows 8, you might find that it's disabled on other versions, or your users might accidentally turn it off, not realizing what it does. To check or change the setting, do the following:

1. Open the Internet Explorer settings dialog box with the Tools | Options menu command.
2. On the General tab, click the Settings button in the Browsing history command group.
3. When the Website Data Settings dialog box appears, switch to the Caches And Databases tab and verify that the Allow Caches And Databases check box is selected.
4. Click OK to dismiss the Website Data Settings and Internet Options dialog boxes.

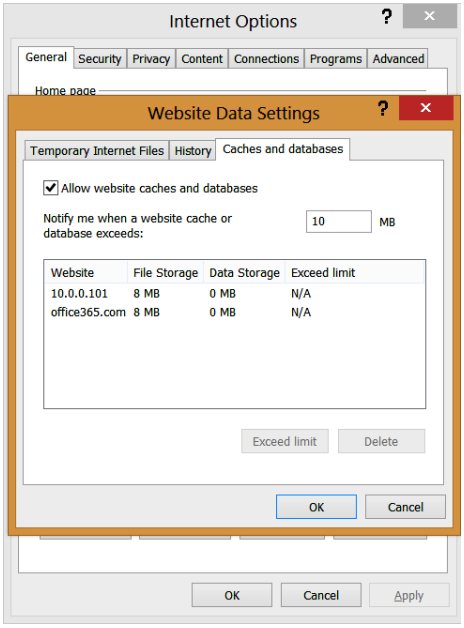

Figure 3-17 Make sure that Internet Explorer is configured to allow web apps to store data locally if you want OWA offline mode to work

Controlling attachment access and rendering

Although feature segmentation is the most obvious use of Outlook Web App policies and receives the most attention, you can also control other aspects of how users work with Outlook Web App through these policies. In particular, you have a fair amount of flexibility in specifying how users may work with attachments in Outlook Web App.

When a user receives a message with an attachment, administrators can control:

- Whether the user sees the attachment at all; if the attachment file type is blocked, the user cannot access it through Outlook Web App.
- Whether the attachment type is allowed and whether the user must use the rendering tools available within Outlook Web App to see a web-based rendering of the attachment data (a feature known as WebReady Document Viewing). This feature displays only HTML text, graphics, and XML-formatted files.
- Whether a rendered attachment should be displayed using the Office Web Apps component (WAC) if it's available. WAC supports Microsoft PowerPoint, Word, and Excel files.
- If rendering is available, whether a user must see a web rendering first before downloading or opening the file.
- Whether the user can open the attachment file directly or whether it must first be saved to disk (which allows local anti-malware scanners to scan it before opening).

This sounds like a fairly complex set of options, and it is. It is challenging to balance users' need to work with documents sent as attachments and the potential security risks that come along with allowing access to complex document formats. The broad scope of Outlook Web App controls for attachment access gives you the tools to adjust what your users can do based on their needs and your organization's security policy.

Attachment access

Outlook Web App categorizes files into four groups:

- *Allowed* files are deemed innocuous and safe to open on the client computer. The list includes types such as Word documents (.doc and .docx extensions) and Windows bitmaps (.bmp extension) that you can be reasonably sure will not contain malicious code.
- *Blocked* file types pose a significant risk to a computer when a user opens them because they can contain executable code. These files include types such as Windows batch files (.bat extension) and Windows command files (.cmd extension).

- *Force save* files are those that a user cannot open directly; instead, she must save them to disk before they can be opened. These types include Adobe Shockwave (.swf) and Director (.dcr) files.
- *Unknown* files are those that are not included in the other lists. The Outlook Web App mailbox policy or Outlook Web App virtual directory setting specifies what should be done with these files; the default is to require them to be saved to disk before opening.

Outlook Web App performs special processing for attachments marked as Force To Save. This means that the user has to save the attachment to his local disk before he can view its contents. As Outlook Web App downloads the attachment from the server, it checks whether it is XML or HTML. In this case, Outlook Web App runs some code called Safe HTML to strip out any malicious XML or HTML code. If the attachment is another type, Outlook Web App examines the content to see whether it actually contains XML or HTML code. This check is performed to ensure that no attachment is ever downloaded that could contain malicious code that could introduce a virus or another dangerous program onto the PC. If hidden XML or HTML code is detected, Outlook Web App strips the attachment and replaces it with a text file to tell the user that the attachment was removed.

The list of file types that are allowed, blocked, and Force To Save can be managed through EAC only. There are actually separate lists for the file types and MIME types you want to allow; you can add items to the allowed, blocked, or forced-save list by either file type or MIME type, using the appropriate value: AllowedFileTypes, AllowedMimeTypes, BlockedFileTypes, BlockedMimeTypes, ForceSaveFileTypes, and ForceSaveMimeTypes. There are separate copies of these lists for each Outlook Web App mailbox policy and each Outlook Web App virtual directory; as a best practice, you should use Outlook Web App mailbox policies to control file access so that the settings you want are consistently applied to users no matter what server they communicate with.

The role of Office Web Apps Server

In Exchange 2007 and Exchange 2010, Microsoft licensed a set of third-party libraries for WebReady Document Viewing. This was a sensible move given that the third-party supplier had already solved the problem of how to render many file types efficiently in a web browser, but it also meant that Microsoft was at the mercy of the vendor for updates to handle new file formats or fix security problems. As part of the Office 2013 release, the Office team built Office Web Apps Server, a separate, standalone server application that, among its other capabilities, Exchange 2013 can use to render PowerPoint, Excel, and Word documents. The Office Web Apps feature is also known as Web Apps component (WAC).

Setting up Office Web Apps is outside the scope of this book, but the TechNet documentation at *http://technet.microsoft.com/en-us/library/jj219458(v=office.15* describes the process well.

Assuming that you have Office Web Apps configured, integrating it with Exchange is simple because there are essentially only two tasks you need to perform. First, you must tell your Exchange servers where the Office Web Apps farm is. You do this with the Set-OrganizationConfig cmdlet and its WACDiscoveryEndPoint parameter, which accepts the URL of the WAC farm. After doing so, the second step is to configure the Outlook Web App virtual directory or (preferably) Outlook Web App mailbox policy to enable the use of WAC for rendering content on public and/or private computers, which you do with the WacViewingOnPrivateComputersEnabled and WacViewingOnPublicComputersEnabled parameters to Set-OWAMailboxPolicy or Set-OWAVirtualDirectory (both of which are true by default).

Optionally, you can force users to view documents using WAC before saving them to disk. This is annoying to end users, but it helps reduce the risk that they'll leave copies of sensitive documents lying around. If you want to enable this feature, the ForceWacViewingFirstOnPrivateComputers and ForceWacViewingFirstOnPublicComputers parameters to Set-OWAMailboxPolicy or Set-OWAVirtualDirectory enables it.

Managing Outlook Web App virtual directory settings

Many of the settings available to control Outlook Web App behavior are duplicated on the Outlook Web App mailbox policy and the Outlook Web App virtual directory objects. This gives administrators some flexibility; in small organizations with only a handful of servers, it's easy to apply settings directly to the Outlook Web App virtual directories, and larger organizations can use Outlook Web App mailbox policies to ensure that the right users get the right settings no matter what servers they use. A fair number of settings are unique to the Outlook Web App virtual directory, though. The Outlook Web App virtual directory settings that pertain to proxying, redirection, and client authentication were discussed in Chapter 1. The integration settings used with Lync are described in Chapter 7, "Integrating Exchange 2013 with Lync Server." That leaves us with a fairly eclectic group of settings available for Set-OWAVirtualDirectory, some of which are nonetheless quite useful.

The Outlook Web App 2013 logon page has intentionally been designed to have a very clean, spare look. The default version includes text fields for the user's logon credentials and a big Outlook Web App logo, and that's all. This is a sharp contrast to the cluttered look and tiny print of the Exchange 2007 and Exchange 2010 Outlook Web App logon pages. However, the new design also takes away some options that were formerly right on the logon page; users can't tell Exchange whether they are on a public or private computer, nor can they voluntarily use the Light mode when they have a slow or unreliable connection. You can fix these two issues by using the LogonPageLightSelectionEnabled and LogonPagePublicPrivateSelectionEnabled parameters to Set-OWAVirtualDirectory; setting them to true enables the corresponding option on the Outlook Web App logon page. You might have to run iisreset to force the changes to appear, though. In addition to these

changes, the logon page is commonly used to display an informational message, such as a warning telling users that unauthorized access is prohibited. You can add these types of messages by editing the logon page, but any such edits will be overwritten when you deploy an Exchange cumulative update or service pack.

You can set the default language users will see when they log on. Outlook Web App uses the language set for a user's mailbox to render the user interface, but it can't do that until the user has logged on. If the user has set a preferred language in his browser, Outlook Web App renders the logon page in that language. However, Set-OWAVirtualDirectory –LogonAndErrorLangauge enables you to set the default language users see when they haven't specified one themselves; you must supply the language code (LCID) of the language you want to use. (Language selection for mailboxes is covered in more detail in Chapter 5, "Mailbox management," in *Microsoft Exchange Server 2013 Inside Out: Mailbox and High Availability*.

You can also change the way Outlook Web App interprets what the user types into the user name field of the logon page. By default, Outlook Web App accepts credentials in three formats. A user named Erik Rucker could thus choose to enter his credentials as domain\username (contoso\ruckere; Microsoft refers to this format as full domain), or he could use his Universal Principal Name (UPN) of ruckere@contoso.com. A third option is to use just the user name, but for this to work Outlook Web App has to know what default domain to use—if Erik just types in ruckere, Outlook Web App has no way to know which of the available Active Directory domains it should sign in to. To solve this, set the default domain by using the DefaultDomain switch. If you want to require users to use a particular format, you can set it with the LogonFormat switch: –LogonFormat FullDomain requires domain\username, –LogonFormat UserName accepts the user name if (and only if) the default domain is also set, and –LogonFormat PrincipalName enables UPN sign-in, but only for users whose UPN is the same as their email address.

Managing Outlook Web App timeouts

You're probably familiar with the timeout values Outlook Web App 2003 and later support; the idea behind these timeouts is that, after a certain period of inactivity, Outlook Web App automatically logs the user out so that a nosy or malicious person can't piggyback on a legitimate user's session. By default, Outlook Web App sessions time out after six hours. This behavior is controlled by two parameters to the Set-OrganizationConfig cmdlet:

- ActivityBasedAuthenticationTimeoutEnabled controls whether timeouts are applied. (The default is $True.)
- ActivityBasedAuthenticationTimeoutInterval specifies the time after which a session is considered idle and thus closes.

Managing Office Store apps for Outlook Web App

In the mobile device world, we have the "bring your own device" (BYOD) model, which democratizes mobility by putting the choice of which device to use, and which apps to run on it, in the hands of individual users. Microsoft is now extending a similar level of choice to end users by allowing them to install and run Office Store apps that run inside Outlook Web App. These apps are hosted on the Mailbox server; by default, individual users can install and run apps, as can administrators. Many organizations will want to retain a degree of control over app installation, so fortunately Exchange 2013 includes some controls. The apps themselves are bundles that can contain HTML, CSS, and JavaScript, along with a manifest file that specifies the app's capabilities (such as whether it can run in Outlook, Outlook Web App, or both) and the level of privilege required to install. An application developer can mark an application as installable by users or administrators. The development model for Outlook-based apps (described at *http://msdn.microsoft.com/en-us/library/fp161135.aspx*) is quite flexible. You can write applications that work on specific types of messages, your apps can modify their appearance or behavior depending on the device where they run, and apps that can run within Outlook 2013 can take advantage of extra services by calling Exchange Web Services routines.

Exchange 2013 CU2 ships with four built-in apps:

- Action Items analyzes the text of your email messages and suggests action items (in the form of Exchange tasks) that are related to the message content.
- Bing Maps scans messages and calendar items for addresses and offers you maps and directions by adding a tab of map data to the window.
- Suggested Meetings reviews the text of messages and suggests appointments that might be added to your calendar.
- Unsubscribe provides a simplified interface for unsubscribing from newsletters, sales email, or other possibly unwanted messages.

Because these apps are built in, you cannot remove them, although you can disable them.

Who can install and configure apps?

Thanks to the RBAC infrastructure that underlies Exchange, when Microsoft adds new features, it often provides a separate management role to control the use of those features. Outlook apps are no exception; Exchange 2013 adds four new RBAC roles. The Org Marketplace Apps role grants permission to install and configure apps that come from the Microsoft Office Store; the Org Custom Apps role grants the ability to install and manage apps that come from internal enterprise distribution points. In the same vein, the My

Marketplace Apps and My Custom Apps user roles grant users the ability to install and manage their own apps.

Enabling or disabling apps at the organization level

By default, the app integration feature is enabled. You can change this with Set-OrganizationConfig –AppsForOfficeEnabled; when it is set to $false, no new apps can be activated for or by any user in the organization. However, changing this setting doesn't remove any existing apps, nor does it prevent users from accessing them. If you want to disable user access to apps completely, you must remove any apps you've added and then disable the built-in apps as described in the next section.

Installing, removing, and configuring apps

Figure 3-18 shows the Apps tab of the Organization slab in EAC. The installed apps are listed; for each app, you can see who provides the app, whether it's available to users, and which users can access the app. When you select an app, the details pane on the right side of the window changes to show the app version, what permissions it requires, and a description provided by the app vendor.

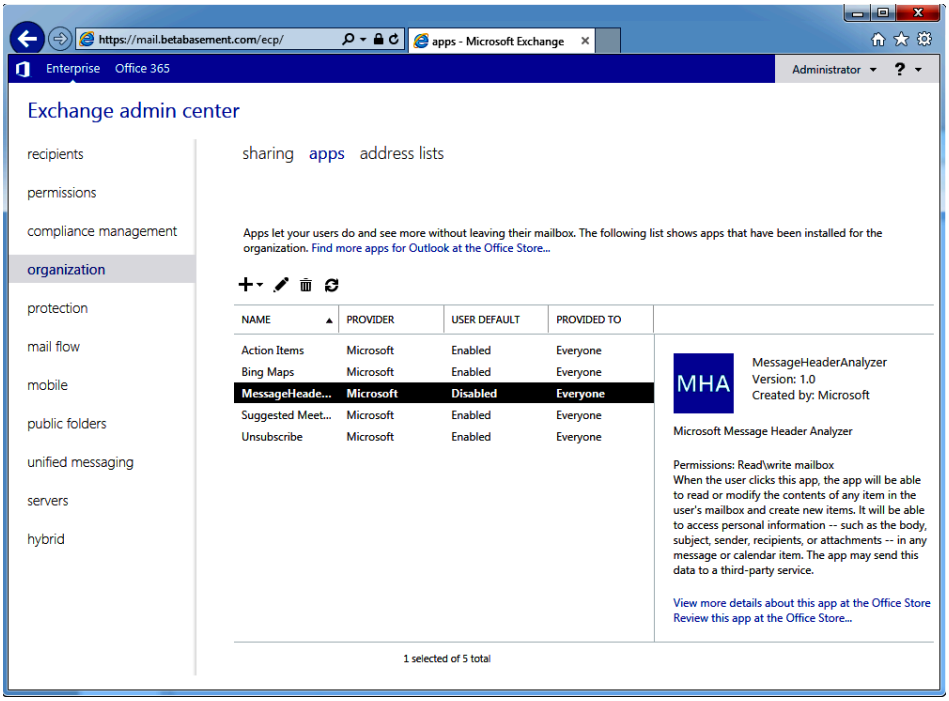

Figure 3-18 The Apps tab of the Organization slab in EAC, showing the installed apps available to users throughout the organization

To add or remove an app, just use the appropriate icons in the toolbar. You can add apps from the Office Store itself or from a URL you specify; the latter option enables you to add apps from a SharePoint app catalog or a local or shared folder. When you install an app, you'll see a confirmation dialog box similar to the one in Figure 3-19; in this case, the LinkedIn app is only asking for permission to read mailbox items, and the summary text reflects that.

INSIDE OUT Apps outside the United States

The Office Store is not yet available in every market that Microsoft serves. In many countries, when you visit the store page, you'll see a message stating that "there are no apps available for Office or SharePoint available for your country/region at this time." This mirrors what's happened with other vendors' app stores; the Apple App Store, Amazon's app store, Google Play, and the Microsoft Xbox Music and Xbox Video have all rolled out to additional countries over time after first being introduced in the United States, and it appears that Microsoft is doing the same in this case. Music, video content, and books are usually licensed separately in each region or territory; downloadable apps can be too, although the Office Store license agreement doesn't seem to place any restriction on transnational app sales. Microsoft has made no public statements about its plans to take the Office Store worldwide, though, so you might have to keep checking for its availability if you're in an area that can't use it currently.

You can also add or remove apps by using EMS; the New-App and Remove-App cmdlets correspond to the toolbar icons. However, using New-App means that you don't get any of the additional data shown in the Office Store.

When you install a new app, it shows up as disabled, and users have no access to it. To change the app's availability, click the pencil icon to open the settings dialog box shown in Figure 3-20. The app can be made available to users by selecting the Make This App Available To Users In Your Organization check box, but just making it available doesn't mean that users will necessarily be able to use it. The group of three option buttons in this dialog box lists the states an individual app can take on: optional and enabled by default, optional and disabled by default, or mandatory. The "by default" in the first two options is there because users can enable or disable optional apps themselves, whereas apps marked as mandatory are always enabled. Users won't see any explicit notification of new apps, and the apps themselves don't appear until the next time a user launches Outlook 2013 or opens Outlook Web App. If you're deploying a new app, you'll need to tell your users about it yourself.

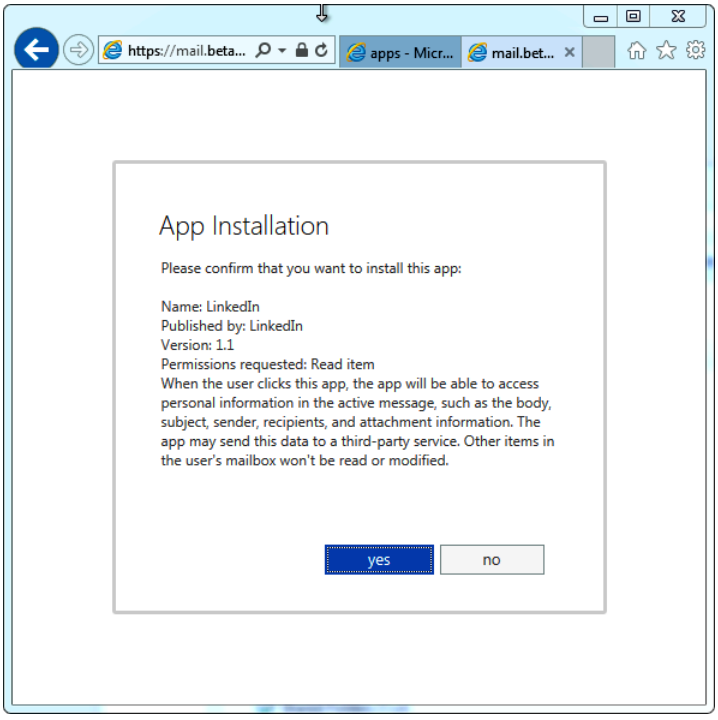

Figure 3-19 The app installation confirmation dialog box

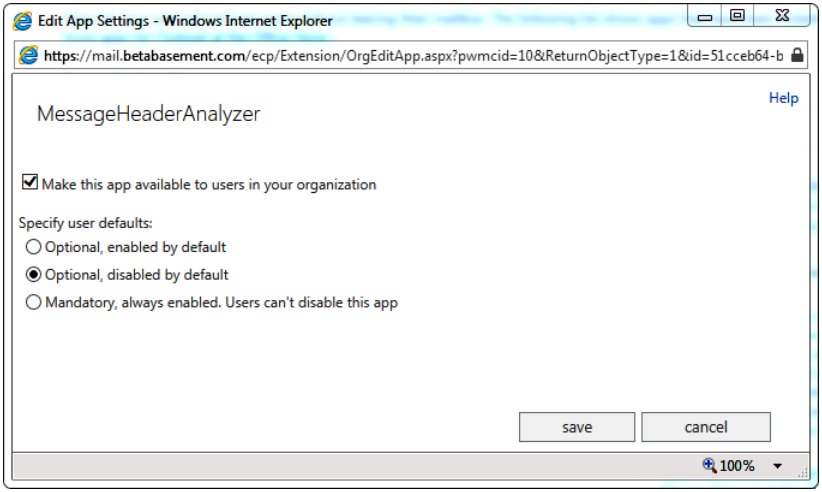

Figure 3-20 The app settings dialog box

Removing apps is easy; select the app, click the Delete icon (it looks like a trashcan), and answer the confirmation dialog box by clicking Yes. The app is removed immediately, and users can no longer access it.

Managing apps from EMS

There are several new cmdlets used for managing apps for individual users. Microsoft classes these as recipient cmdlets because they enable, disable, add, remove, or configure apps for one or more specific users. Each app has a unique identity, represented as a globally unique identifier (GUID), but most of the time you'll use the app's friendly name to keep track of which apps are present.

The Get-App cmdlet returns a wealth of data about individual apps, but to get it, you'll need the application ID. To get the ID for an individual app, you can do something like this:

```
get-app -OrganizationApp | ft DisplayName, AppID
```

```
DisplayName                    AppId
-----------                    -----
LinkedIn                       333bf46d-7dad-4f2b-8cf4-c19ddc78b723
MessageHeaderAnalyzer          62916641-fc48-44ae-a2a3-163811f1c945
Bing Maps                      7a774f0c-7a6f-11e0-85ad-07fb4824019b
Suggested Meetings             bc13b9d0-5ba2-446a-956b-c583bdc94d5e
Unsubscribe                    d39dee0e-fdc3-4015-af8d-94d4d49294b3
Action Items                   f60b8ac7-c3e3-4e42-8dad-e4e1fea59ff7
```

The OrganizationApp parameter specifies that you just want to see applications that are scoped to the organization. These might be apps from the Office Store or custom enterprise apps; the distinction between organization apps and user apps is that the organization apps are stored at the organization level and are potentially available to all users in the organization.

There are several other interesting properties on the individual applications, including the XML for the application manifests (stored in the ManifestXml property) and the application's scope and permissions.

If you run the Get-App cmdlet by itself, you get a summary of the apps installed in your organization:

```
Get-App
```

```
DisplayName                    Enabled                   AppVersion
-----------                    -------                   ----------
LinkedIn                       False                     1.1
MessageHeaderAnalyzer          False                     1.0
```

```
Bing Maps                       True                    1.0
Suggested Meetings              True                    1.0
Unsubscribe                     True                    1.0
Action Items                    True                    1.0
```

Each app's display name and version are shown. The value in the Enabled column reflects whether the app is enabled or disabled by default, not whether it's available to any particular user. To see the state of applications for a particular user, you specify that user with the –mailbox switch to Get-App, like so:

```
get-app -mailbox paul
```

```
DisplayName                     Enabled                 AppVersion
-----------                     -------                 ----------
LinkedIn                        True                    1.1
MessageHeaderAnalyzer           True                    1.0
Bing Maps                       True                    1.0
Suggested Meetings              True                    1.0
Unsubscribe                     True                    1.0
Action Items                    True                    1.0
```


Notice that this summary shows the LinkedIn and MessageHeaderAnalyzer apps as enabled for my mailbox, even though their default state in the previous output was disabled. That's because I enabled those apps directly for my mailbox.

You can change the enabled state of apps with the Enable-App and Disable-App cmdlets. These change the default state of the app for all users unless you pass the –Mailbox parameter. For example, if you install a new app and then use Disable-App on it, the app will be disabled by default for new users, but they can still enable it themselves.

If you want to change the enabled or disabled state of an app for users directly, you do that with the Set-App cmdlet. For example, to prevent users from seeing or using the MessageHeaderAnalyzer app, you could use a command like the following to turn it off:

```
Get-App | where {$_.DisplayName –like "MessageHeaderAnalyzer"} | Set-App
–OrganizationApp –Enabled:$false
```

The Enabled property controls whether the app is enabled for (and thus visible to) users. You can use the –DefaultStateForUser switch to control whether an enabled app is turned on for users by giving it a value of Enabled or Disabled; if you use –DefaultStateForUser AlwaysEnabled, that forces the app on.

Which users see these changes depends on two other parameters to Set-App. The –ProvidedTo switch can be set to Everyone or to SpecificUsers; in the latter case, you can

either pipe in a set of mailboxes or use the –UserList switch. For example, to make the app named ScreenShotChecker available to all users in the Legal group, you could do the following:

```
$a = Get-DistributionGroupMember Legal
Get-App | where {$_.DisplayName -like "ScreenShotChecker"} | Set-App -OrganizationApp
-ProvidedTo SpecificUsers -UserList $a -DefaultStateForUser AlwaysEnabled
```

Self-service app management for users

After an app is installed, either by you or a user, an individual user might be able to enable or disable optional applications from within Outlook Web App 2013 or Outlook 2013 (which actually uses the EAC options component, as shown in Figure 3-21). Their ability to do this depends on their having the appropriate RBAC roles, as described earlier in the chapter (which they will have by default). Users can manage apps from Outlook Web App by clicking the Options icon (the gear in the upper-right corner of the window) and choosing Manage Apps, or from Outlook 2013 by opening the backstage view and choosing the Manage Apps link. In either case, the view similar to that shown in Figure 3-21 will appear; one important difference between this view and the one shown in Figure 3-18 is that this view includes a user-installed app (as indicated by the value of *User* in the Installed By column).

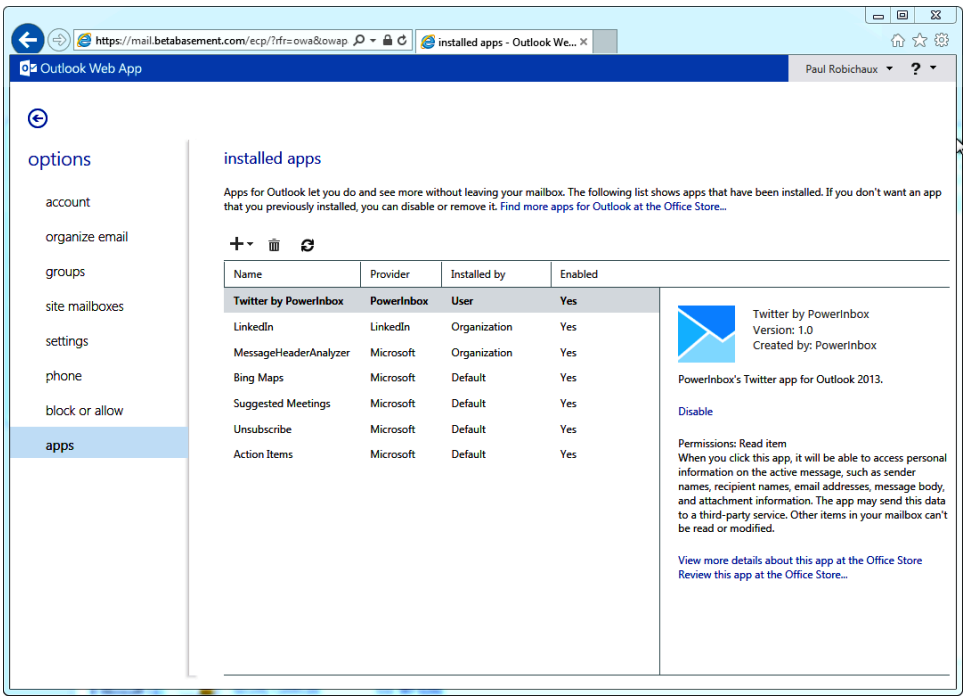

Figure 3-21 The list of installed apps for an individual user mailbox

From this screen, users can add and remove their own apps. They can't disable or enable user apps; the presumption is that if you don't want access to an app, you can just remove it instead of disabling it. Individual users also cannot change the enablement state for organization apps; only administrators can do that.

Customizing Outlook Web App

Themes define the color scheme and graphic elements used for Outlook Web App. Exchange 2007 introduced support for customizable Outlook Web App themes; Exchange 2013 continues this support by including a set of 22 themes that users can apply to customize their Outlook Web App session's appearance. Administrators don't have control over user choice and cannot impose a theme on users, although they can set a default theme that users will have unless they change it.

You can also create your own theme and include corporate logos, color schemes, and so on. Creating a complete theme is a very extensive customization of the Outlook Web App user interface (UI). The simplest way to do this is to copy one of the existing themes (located in \Program Files\Microsoft\Exchange Server\V15\ClientAccess\owa\Current\themes) and then edit the files you find there, being careful to preserve exact dimensions.

Many companies liked the idea of incorporating some aspect of their corporate identity in Outlook Web App without doing the work to create a theme. The complete source code of the Outlook Web App application is distributed with Exchange, and the classic solution to the problem is to customize some of the files used for the default theme. Microsoft published customization guides for Exchange 2007 and Exchange 2010, but as of this writing has not yet done so for Exchange 2013. However, the most common customizations—color scheme and logo changes on the logon page—are fairly simple to apply.

Figure 3-22 shows the logon page with callouts showing the parts of the logon page. You can use this information to develop the necessary customizations to comply with corporate branding. The graphic files and style sheets used to render the logon page are stored in Program Files\Microsoft\Exchange Server\V15\FrontEnd\HttpProxy\Owa\ Auth\Current\themes\Resources. You are free to edit any of these, although the items shown in Figure 3-22 are the most common ones that administrators want to customize.

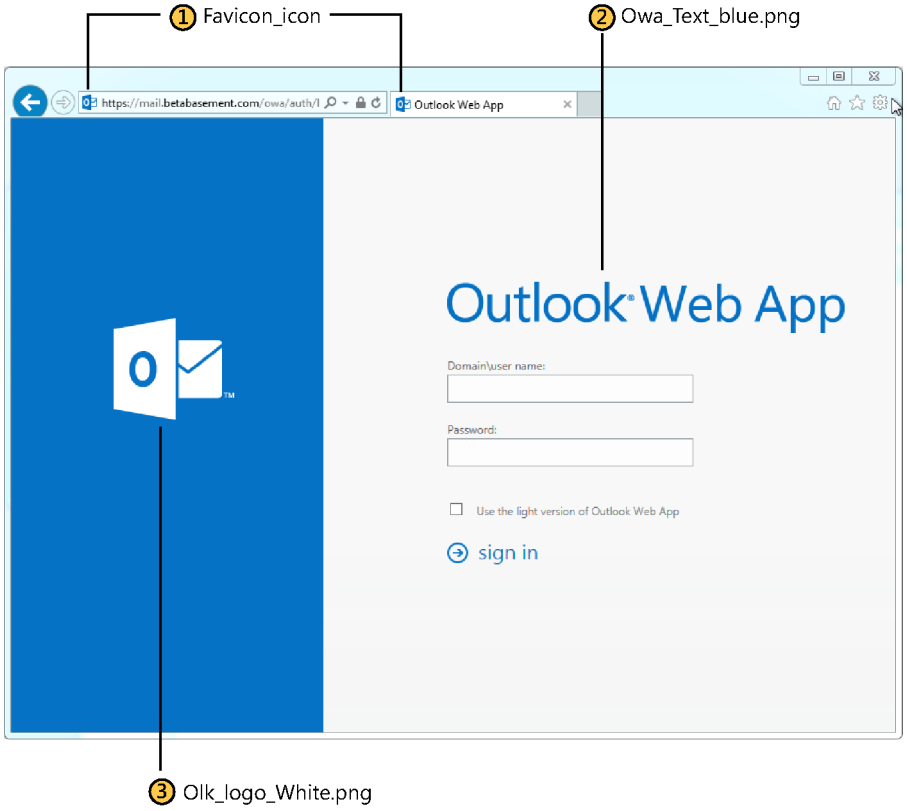

Figure 3-22 Components of Outlook Web App customization

Another common request is to add text or graphics to the logon page, either instead of or in addition to changing the existing graphics. This is much more challenging now that the Outlook Web App logon page includes touch support; rather than a largely static page, as in previous versions, the Outlook Web App 2013 logon page is full of scripts that dynamically determine what kind of device the user is connecting from. However, you can make simple customization by doing the following.

To customize the logon page with some new text, do the following:

1. Make a copy of the \Program Files\Microsoft\Exchange Server\V15\FrontEnd\HttpProxy\owa\auth\Logon.aspx file. This file tells IIS how to render the logon page when a user connects to Outlook Web App.

2. Create a file that contains the HTML you want displayed on the page and then save it in the same directory as Logon.aspx. Give it an easily remembered name.

3. Open the Logon.aspx file with a text editor and search for the <div class="signInError" string. This is where Outlook Web App displays an error message if logon fails. Right above this line, insert an #include directive to instruct Outlook Web App to read and display the text contained in the file you created in step 2; wrap the line in a tag, using class "signInExpl". The code in Logon.aspx will then look something like this:

```
<div class="myCustomText" class="signInExpl">
<!-- #include file="contoso-disclaimer.inc" -->
</div>
<div class="signInError…
```

4. Save Logon.aspx and restart an Outlook Web App session. (You don't have to restart Microsoft Internet Information Services [IIS] or any of the Exchange services.)

5. You can take the same approach to update Logoff.aspx if you want users to see a customized message when they sign out of Outlook Web App.

6. When you are happy with the customization, you can apply it on all CAS servers by copying your modified files. There is no automatic mechanism to apply this kind of customization on every CAS server in an organization.

INSIDE OUT Customizations will be overwritten by future product updates

The other thing to remember is that any customization of one of the Outlook Web App components will almost certainly be overwritten by cumulative updates and service packs. CU1 overwrites Outlook Web App customizations, and there is no reason to doubt that future updates will be any different. That's why you should keep careful documentation about any customization you apply to Outlook Web App to make it easier to apply it after you upgrade Exchange. You should also keep a copy of both the original and customized versions of any file you change so you can review them in the future. It's also fair to say that there is no guarantee that Microsoft will not change the way Outlook Web App works in a future version and render this method of customization—or any method of customization—invalid, so be prepared to build some time to test and perhaps do a little recoding for Outlook Web App customizations into every deployment plan.

Managing Outlook for Mac

Although it shares the Outlook name with its Windows sibling, the Mac version of Outlook is a very different beast. It doesn't use MAPI, relying exclusively on EWS instead. It lacks some features of the Windows version (such as support for MailTips, Policy Tips, personal archives, data leak prevention policy tips, site mailboxes, modern public folders, and retention policies), but in exchange it adds some Mac-specific features such as the ability to insert pictures easily from an iPhoto library and full integration with Apple's Spotlight desktop search tool. As an Exchange administrator, virtually none of what you know about administering Outlook for Windows will be useful when administering and managing Mac Outlook, with some notable exceptions.

- First is Autodiscover. The Mac version of Outlook fully supports Autodiscover. If your Exchange environment passes the ExRCA Autodiscover tests, then Outlook 2011 should connect to it just fine. Keep in mind that Mac Outlook doesn't perform service connection point (SCP) lookups, as Windows Outlook does. From that standpoint, just treat Mac Outlook like a Windows Outlook client that's trying to connect from outside the corporate network.

  However, you might notice an issue when a roaming client (such as a Mac laptop) connects from the corporate network and then roams to outside the corporate network. If Autodiscover finds an internal-facing URL, Outlook 2011 will happily keep trying to use it even after the laptop roams back to the Internet or another network where it should be using an external-facing URL. The fix for this is to edit the account settings of the affected account and put the correct external URL back in.

- Second is that, unlike Windows Outlook, the Mac version doesn't have any way to see the current connection status or to force a reconnection, meaning there's no easy way to force a new Autodiscover request. The fastest way to do this is to quit and re-launch the client. To force a reconnection, you can also use the Work Offline menu item under the Outlook menu; use it to switch to offline mode and then switch back to online mode.

- Third is that the Mac client can import PST files from Outlook 2003, Outlook 2007, Outlook 2010, and Outlook 2013. Mac Outlook doesn't use OST files. It also cannot export email to PST.

- Fourth is that Set-CASMailbox has separate parameters that enable you to block the EWS edition of Entourage and Mac Outlook—EwsAllowEntourage and EwsAllowMacOutlook.

Mac Outlook has a useful logging feature that is in a somewhat unusual location. If you enable connection logging, it will give you details on Autodiscover requests, folder and item synchronization through EWS, and Lightweight Directory Access Protocol (LDAP)

access to domain controllers (DCs) and global catalogs (GCs). To turn on logging, you must do the following:

1. Launch Outlook 2011.
2. Choose the Error Log command in the Windows menu.
3. When the Errors window appears, click the large gear icon in the upper-right corner of the window.
4. In the resulting dialog box, make sure the Turn On Logging For Troubleshooting check box is selected and then click OK.

Outlook immediately creates a new file named Microsoft Outlook_Troubleshooting_0.log on the Mac OS X desktop. As long as Outlook is running, it will continue to append entries to this log until you repeat the preceding steps and clear the logging check box.

Managing Outlook Web App for Devices

Managing Outlook Web App for Devices is a weird combination of managing Outlook Web App and managing Exchange ActiveSync devices. If you disable EAS access to a server or a mailbox, clients using that mailbox or server won't be able to connect with Outlook Web App for Devices. You have finer-grained control, though, because there are several settings in the Outlook Web Access mailbox policy object that let you control what users of the mobile app may do. As of Exchange 2013 CU2, the only Outlook Web App for Devices–specific argument to Set-OWAMailboxPolicy is AllowCopyContactsToDeviceAddressBook, which controls whether the device is allowed to cache the user's contacts in the device's address book. If this setting is false, the user can still see her contacts in the app, but they're not visible to the built-in phone app or other apps that depend on the system address book for name or number resolution. However, a number of other parameters (such as –IRMEnabled) are of interest because they control features that are available through Outlook Web App for Devices. If you create an Outlook Web App access policy that disables one of these features (say, integration with Office apps), the feature will be disabled for any user who is subject to the policy, whether the user accesses Outlook Web App through the browser or through the mobile app.

POP3 and IMAP4

POP3 and IMAP4 are Internet email protocols that a wide variety of clients and servers support. Fans of these protocols love the lightweight nature of their connections, which is one of the reasons they have long been the protocols of choice for free email services such as Outlook.com and Gmail (Hotmail-supported POP3; Gmail supports both protocols). POP3

is the older and less functional protocol. IMAP4 is more functional than POP3 but less functional than MAPI.

Nevertheless, modern IMAP4 clients, including Outlook, can build a rich range of features around the rudimentary but superefficient communications to download messages from a server. Of course, unlike MAPI, POP3 and IMAP4 are both protocols that clients use to retrieve messages from a server. Both of these protocols transfer mail to the client. IMAP and POP clients must use SMTP to send outbound mail.

Apart from age, the fundamental differences between the two protocols are as follows:

- POP3 downloads messages to a client and removes them from the server.
- POP3 supports a very limited set of folders on the server (essentially, the Inbox).
- IMAP4 can leave copies of downloaded messages on the server.
- IMAP4 can access any folder a server exposes and download messages from the folders to client-side replicas.
- IMAP4 allows a live-sync mode in which the client holds open a connection to the server; this provides a more Outlook-like sync experience in which messages trickle in to the Inbox as they arrive instead of arriving in batches when a POP3 connection is made.

The majority of clients that connect to Exchange 2013 through POP3 and IMAP4 belong to four categories:

- Users in an educational establishment such as a university, where the priority is on providing basic email services at the lowest possible cost.
- Users who access an Exchange mailbox with IMAP to avoid having to buy Outlook licenses. (Of course, now that the Windows 8 and Windows RT Mail clients support using EAS, users on those platforms can get the benefits of faster and more robust synchronization from EAS, using their existing Exchange mailbox client access licenses [CALs]).
- Users who don't like Outlook. Often, these people have used a client such as Eudora or Thunderbird for many years and don't see a reason to change.
- Users who run an operating system that doesn't support the premium version of Outlook Web App or who simply prefer to use IMAP. Many Linux and UNIX users are in this category. In fact, so are users of Surface RT devices, given that no native version of Outlook is available in Windows RT currently.

The attraction of using free POP3 or IMAP4 clients is the avoidance of Outlook license fees. This is less of an issue in large corporations that negotiate enterprise licensing agreements with Microsoft that include the entire Office application suite. For this reason, relatively few users in large corporate deployments use POP3 or IMAP4 clients. Outlook Web App is available if they don't want to use Outlook, and it's easier for the help desk if a limited number of clients are in use. Another reason is that POP3 and IMAP4 clients are purposely designed to work across any server that supports these protocols. They therefore do not support features that are specific to Exchange, such as MailTips, unified messaging integration, and so on.

For the remainder of this discussion, I focus on setting up the Exchange 2013 IMAP4 server and configuring clients to connect to the IMAP4 server. The steps to set up and configure POP3 access are conceptually similar, but because very few Exchange sites actually use POP, it's not covered here.

Configuring the IMAP4 server

When you install Exchange 2013, the setup program creates the Microsoft Exchange POP3 and Microsoft Exchange IMAP4 services to support client connections through these protocols but does not start the services. There are actually two services for each protocol. The Microsoft Exchange IMAP service runs on the CAS, whereas the Microsoft Exchange IMAP Backend service runs on the Mailbox role, with corresponding services for POP. If you have a multirole server, you'll see both services.

The services aren't started or enabled by default as part of the Microsoft overall strategy of reducing the attack surface of computers by disabling unneeded services; its thinking is that because most sites won't use IMAP or POP, the services should remain off. Therefore, the first step to support POP3 or IMAP4 clients is to start these services. In addition, you should change the startup state for the services from Manual to Automatic so that Windows starts them every time the server is booted:

```
Set-Service msExchangeImap4 -StartupType Automatic
Set-Service msExchangeImap4Backend -StartupType Automatic
Start-Service -Service msExchangeImap4Backend
Start-Service -Service msExchangeImap4
```

After starting the IMAP services, they will listen on two ports: TCP 993 for connections using SSL and TCP 143 for connections using either Transport Layer Security (TLS) or no security at all. You usually don't have to make any configuration changes after the service is started. However, clients won't know where to connect. Outlook Web App includes the ability to show users what IMAP, POP, and SMTP settings they should plug in to their clients, but this advertisement is turned off by default. To make these settings visible in Outlook Web App for users who are IMAP-enabled or POP-enabled, you need to do the following:

- To advertise IMAP or POP settings, use Set-IMAPSettings or Set-POPSettings with the –ExternalConnectionSettings parameter, which should include the FQDN of the machine and the port number and encryption scheme.
- To advertise SMTP settings, use Set-ReceiveConnector –AdvertiseClientSettings:$true –FQDN fqdn.

Therefore, to publish mail.betabasement.com as the server name for a server named pao-ex01.betabasement.com, you could do the following:

```
Set-IMAPSettings –ExternalConnectionSettings mail.betabasement.com:995:SSL
Set-ReceiveConnector "Client Frontend PAO-EX01" –AdvertiseClientSettings:$true
–FQDN mail.betabasement.com
```

After this change, you must run iisreset before Outlook Web App will update. After you've done so, users will see a new link labeled Settings For POP or IMAP Access on the Account tab of their Outlook Web App settings page. Clicking that link displays a window similar to the one shown in Figure 3-23. Keep in mind that Exchange will show whatever FQDN you specify; if it's wrong, clients will see the wrong information.

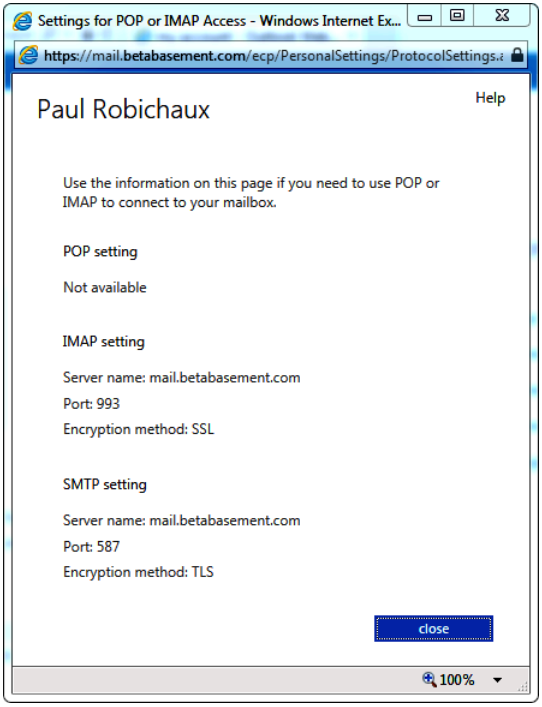

Figure 3-23 Displaying client configuration information for an IMAP user

Figure 3-24 shows the properties that are usually of most interest to administrators when they configure IMAP connectivity for Exchange 2013. The default configuration is for the IMAP service to listen on all available IPv4 and IPv6 addresses using TCP ports 143 and 993; you can change these bindings by using the two lists of IP addresses in the settings dialog box.

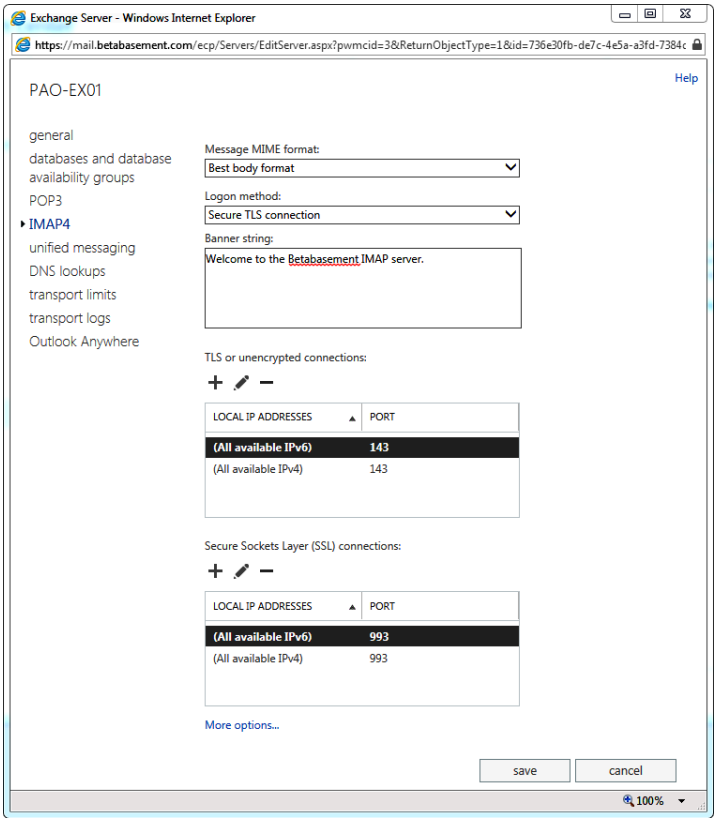

Figure 3-24 Viewing the basic properties of the Exchange IMAP4 server

The More Options link below the SSL bindings list expands the settings list to include time-out values and connection limits.

See TechNet for detailed information on these settings.

The Get-IMAPSettings and Set-IMAPSettings cmdlets retrieve and set configuration settings for the IMAP4 server. The equivalent cmdlets for POP3 are Get-POPSettings and Set-POPSettings. For example, to retrieve the current configuration for a multirole server named PAO-EX01, use the following command:

```
Get-IMAPSettings -Server PAO-EX01
```

```
ProtocolName                      : IMAP4
Name                              : 1
MaxCommandSize                    : 10240
ShowHiddenFoldersEnabled          : False
UnencryptedOrTLSBindings          : {[::]:143, 0.0.0.0:143}
SSLBindings                       : {[::]:993, 0.0.0.0:993}
InternalConnectionSettings        : {PAO-EX01.betabasement.com:993:SSL,
PAO-EX01.betabasement.com:143:TLS}
ExternalConnectionSettings        : {mail.betabasement.com:993:SSL}
X509CertificateName               : PAO-EX01
Banner                            : The Microsoft Exchange IMAP4 service is
ready.
LoginType                         : SecureLogin
AuthenticatedConnectionTimeout    : 00:30:00
PreAuthenticatedConnectionTimeout : 00:01:00
MaxConnections                    : 2147483647
MaxConnectionFromSingleIP         : 2147483647
MaxConnectionsPerUser             : 16
MessageRetrievalMimeFormat        : BestBodyFormat
ProxyTargetPort                   : 9933
CalendarItemRetrievalOption       : iCalendar
OwaServerUrl                      :
EnableExactRFC822Size             : False
LiveIdBasicAuthReplacement        : False
SuppressReadReceipt               : False
ProtocolLogEnabled                : False
EnforceCertificateErrors          : False
LogFileLocation                   : C:\Program Files\Microsoft\Exchange Server
\V15\Logging\Imap4
LogFileRollOverSettings           : Daily
LogPerFileSizeQuota               : 0 B (0 bytes)
ExtendedProtectionPolicy          : None
EnableGSSAPIAndNTLMAuth           : True
Server                            : PAO-EX01
Identity                          : PAO-EX01\1
```

If you change any of the configuration settings for the IMAP4 server, you have to restart the Microsoft Exchange IMAP4 service. It's common to find that you want to turn on protocol logging to help debug connections from a particular client. To enable protocol logging for IMAP4 clients, you need to enable logging and tell Exchange where it should create the log. Enabling logging in Exchange 2007 requires you to edit a configuration file, but in Exchange 2010 and 2013, you can enable logging with the Set-IMAPSettings cmdlet. For example:

```
Set-IMAPSettings -Server PAO-EX01-ProtocolLogEnabled $True -LogFileLocation
'C:\Logs\'
```

Logging generates a mass of data on the server, some of which is fairly obtuse if you are not familiar with debugging IMAP connections. Clients can also generate logs, and if you need to provide data to help a support representative solve a problem, you should generate server and client logs to ensure that they have full knowledge of what the client is sending and how the server is responding.

Configuring IMAP4 client access

From a user perspective, it is easy to configure a POP3 or IMAP4 client to connect to Exchange 2013. For my example, I chose the Thunderbird free IMAP4 client that you can download from *http://www.mozillamessaging.com/en-US/thunderbird/*. Two connections must be configured before an IMAP4 client can download and send messages.

- An IMAP4 server hosted by an Exchange 2013 CAS or multirole server must be ready to accept client connections so that IMAP4 clients can access mailboxes and download folders and items.
- An SMTP receive connector must be available to accept client connections to allow IMAP4 clients to relay outgoing messages through SMTP.

The steps required to configure the client to connect to Exchange 2013 are as follows:

1. Set the authentication setting to Basic for the IMAP4 server on the CAS to which you want to connect the client. This is sufficient for testing purposes because it ensures that just about any IMAP4 client can connect. When you have established that connections work freely, you can increase the level of security by moving to Integrated Windows Authentication or Secure Logon, depending on which authentication mechanisms the client supports.
2. Restart the IMAP4 server to effect the change in the authentication setting.
3. Configure the client with the name of the CAS server and the user name in domain name\account name format. In the case of the Mac OS X version of Thunderbird, this requires filling out the dialog box shown in Figure 3-25.

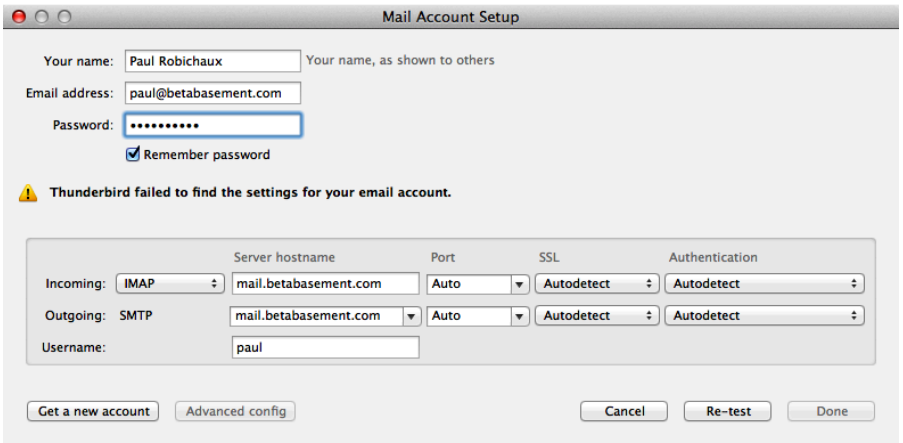

Figure 3-25 Configuring basic IMAP account settings in Thunderbird

4. Connect the client to prove that messages can be downloaded.

5. Check that the Permission Groups assigned to the default client receive connector on the server that you want to use for sending outbound messages allows anonymous connections. Again, this is the easiest setting to use to test outgoing message connectivity and should ensure that all types of client can connect to send messages. When you know that messages are flowing, you can increase the security. Thunderbird (and most other IMAP clients) supports STARTTLS security with user name and password credentials, so the receive connector doesn't need to allow anonymous connections because these authenticated connections will be regarded as Exchange users.

After messages are being downloaded and sent freely, the next step is to configure LDAP access to Active Directory so that you can use Active Directory as an address book. The details of how to configure a connection to Active Directory vary from client to client, as does the ability of the client to use the data fetched from Active Directory. Some clients can only browse Active Directory, whereas others, such as Thunderbird, can validate email addresses against Active Directory as they are entered into message headers.

You can use the following process to configure Thunderbird or a similar client:

1. Set the name of the connection to Active Directory or whatever else you like as an illustrative name.

2. Set the directory server hostname to the FQDN of a global catalog server that is reachable from the client. Ideally, this should be a global catalog server in the same site as the CAS server.

3. The base distinguished name (DN) provides a starting point for LDAP searches in the directory. You will probably want to use the root of the directory tree for your domain.

4. The port number is set to 3268 rather than the standard port (389) LDAP uses.

5. The bind DN should be set to the user's SMTP address.

To test the connection, open the client address book and search for some mailboxes that you know exist. You should be able to see mailboxes, contacts, and distribution groups.

INSIDE OUT **Only minor issues**

Two small issues are the following:

- **The LDAP searches executed by a client might ignore Exchange-specific filters. For example, if you select the Hide From Exchange Address Lists check box for an object, it stops Outlook and Outlook Web App users from seeing that object through the GAL. However, this block means nothing to other clients, and the hidden objects will probably be revealed to users.**

- **Along the same lines, an LDAP search against Active Directory doesn't impose any filters to eliminate objects that are not mail-enabled, so you'll probably be able to see security groups such as Enterprise Admins. However, you won't be able to send email to these objects because they don't have email addresses.**

These are small hiccups along the road, and because users have read-only access to directory information that reveals some objects that other clients don't show isn't really very serious.

Client throttling

Clients can occasionally create an excessive load on an Exchange server. The reasons this happens are many and varied but usually involve some form of software bug that causes the client to communicate in an unpredictable manner and so create an out-of-the-ordinary load. The usual corrective action taken in previous versions is first to identify the errant client with the Exchange User Monitor (ExMon) utility and then terminate its connection to relieve the strain on the server and restore normal levels of responsiveness to other clients. You can then figure out what action the client was taking to cause the problem and resolve the situation. However, this can be a labor-intensive process, and it doesn't scale well. To help protect against misbehaving clients in Office 365, Microsoft introduced a feature in

Exchange 2010 known as client throttling, which enables administrators to set limits proactively on what clients can do; more precisely, it enables them to establish policies that control what resources, and how many of them, clients may consume for various kinds of connections. When a client exceeds the limit for resource usage, it's not usually blocked altogether; instead, its requests are delayed. The net effect is that clients who are using more than their fair share of any particular resource will have their access to that resource reduced, with a gradual reduction in the amount of slowdown that allows the client to recharge.

Exchange 2013 client throttling policies can be applied to several distinct applications and protocols:

- Exchange ActiveSync (EAS)
- Outlook Web App
- POP3 and IMAP4
- Exchange Web Services (EWS; this category includes unified messaging users and users running Entourage or Outlook for Mac OS X)
- Discovery searches
- Cross-forest access (including hybrid access between on-premises and Office 365 tenants)
- Message sending using SMTP
- Windows PowerShell and PowerShell Web Services

A default policy is automatically created and enforced within the organization when you install Exchange 2013. This policy, named GlobalThrottlingPolicy_GUID, is intended to be the baseline policy for any user who doesn't have a more specific policy applied. Although you can change the limits enforced by the default policy, Microsoft doesn't recommend doing so (and you may not remove or rename the default policy). That's because you can create additional policies that provide more granular control over resource usage and then assign those policies to users. Any throttling setting not explicitly specified in a policy is inherited from the global policy, so you can quickly build policies that control exactly the resources you want while allowing other resources' usage to be governed by the global policy.

The policy comes into effect when the percentage of CPU usage by Exchange exceeds the threshold defined in the CPUStartPercent property of the default policy. This setting is applied on a per-service basis. The default value for CPUStartPercent is 75, so when one of the Exchange services monitored for client throttling reaches this threshold, Exchange

begins to apply any throttling restrictions that are defined in the default policy or on a per-mailbox basis to ensure that the server can continue to provide a reasonably smooth service to all clients.

In general, four common parameters are associated with each type of resource usage:

- CutoffBalance controls the level at which Exchange starts denying access to a resource. Think of this as a hard maximum limit for using the resource; after the client hits it, it will be blocked from that resource until it recharges.
- RechargeRate is the speed at which the user's resource budget recharges or refills. For example, a client that sends a large number of messages and hits the recipient rate limit will fall below the limit and regain full access as time passes.
- MaxBurst controls how far above the standard resource limit a client may go in short bursts.
- MaxConcurrency sets the limit for how many concurrent connections or actions a single client may take.

Individual protocols or applications might have additional settings, too; see the TechNet documentation for the New-ThrottlingPolicy cmdlet for a complete list.

Throttling policies can be managed only through EMS. You can view details of the default policy with this command:

```
Get-ThrottlingPolicy | Where {$_.IsDefault –eq $True} | Format-Table
```

A lot of data is output when you examine the attributes of a throttling policy. However, you can break them down into the categories listed earlier. Thus, you can retrieve the settings that govern EWS clients with:

```
Get-ThrottlingPolicy | Select Ews* | Format-List
```

```
EwsMaxConcurrency                    : 27
EwsMaxBurst                          : 300000
EwsRechargeRate                      : 900000
EwsCutoffBalance                     : 3000000
EwsMaxSubscriptions                  : 5000
```

The output for the EWS parameters indicates that multiple thresholds are currently in place to control user workload: the maximum concurrency for any user is set to 27 (the range is from 0 to 100), meaning that a user can have up to 20 active EWS sessions. A connection is maintained from the time a request is made to establish it until the connection is closed or otherwise disconnected by a user action (logging off). If a user attempts to establish

more than the allowed maximum, that connection attempt will fail. The EwsMaxBurst, EwsRechargeRate, and EwsCutoffBalance limits are set, too, indicating that limits for specific resource usage are in place. The max burst and recharge rate are both expressed in milliseconds; the user can have a burst of up to five minutes of heavy EWS activity before being blocked.

Similar groups of settings are available for the other client categories. For example, you can find those applying to Outlook Web App with:

```
Get-ThrottlingPolicy | Select OWA* | Format-List
```

TROUBLESHOOTING

Exchange is throttling BlackBerry Enterprise Server activities

Introducing client throttling had an unfortunate side effect on some applications that impose heavy demands on Exchange. The BlackBerry Enterprise Server (BES) provided the best example because the account it uses essentially mimics a hyperactive user who accesses multiple mailboxes to fetch and send messages to mobile devices. The usual problem was that Exchange throttled BES activities because it exceeded the RCA maximum concurrency threshold. The solution was to create a new throttling policy that set the value of the RCAMaxConcurrency parameter to $Null and then assigned the new policy to the BES account. This is a step the administrator can perform after installing BES.

A number of specific parameters are available to control workload generated through Windows PowerShell:

- **PowerShellMaxConcurrency (default value 18)** This constraint is applied in two ways. It defines the maximum number of remote Windows PowerShell sessions a user can have open on a server at one time. It also defines the maximum number of cmdlets EMS can execute concurrently.
- **PowerShellMaxCmdlets (default no limit)** Sets the number of cmdlets a user can execute within the time period specified by PowerShellMaxCmdletsTimePeriod. After the value is exceeded, no future cmdlets can be run until the period expires.
- **PowerShellMaxCmdletsTimePeriod (default no limit)** The period in seconds Exchange uses to determine whether the maximum number of cmdlets constraint has been exceeded.

- **ExchangeMaxCmdlets (default no limit)** Specifies the number of cmdlets a user can execute within the time period set by PowerShellMaxCmdletsTimePeriod. After the constraint is exceeded, Exchange slows down the execution of other cmdlets.

- **PowerShellMaxCmdletQueueDepth (default no limit)** Specifies the number of operations Exchange allows a user to execute. Operations are consumed by cmdlets as they run. They are also consumed by internal operations. (For example, the PowerShellMaxCurrency operation uses two operations.) Microsoft recommends that, if set, the value of PowerShellMaxCmdletQueueDepth should be set to three times the value of PowerShellMaxConcurrency. Exchange does not apply this constraint to the code run by ECP or EWS.

Three additional settings can constrain the consumption of general resources:

- **MessageRateLimit (default no limit)** Governs the number of messages per minute that a user can submit to the transport system for processing. Messages over the limit are placed in the user's Outbox until the server can accept them. The exception is for clients such as POP3 and IMAP4 that submit directly to the transport system using SMTP. If these clients attempt to submit too many messages, their request is declined, and they are forced to reattempt later.

- **RecipientRateLimit (default no limit)** Specifies the number of recipients that can be addressed in a 24-hour period. For example, if this value is set to 1,000, the user may address messages to up to 1,000 recipients daily. Messages that exceed this limit are rejected.

- **ForwardeeLimit (default no limit)** Specifies a limit for the number of recipients that can be configured in Inbox Rules for the forward or redirect action.

INSIDE OUT Storing the default throttling identifier in a variable

You'll note that the default throttling policy is given a value such as DefaultThrottlingPolicy_dade6c60-e9cc-4692-bc6a-71771158a82f as its name and identifier. I suspect that this is a joke played on us by the Microsoft engineers because no sensible human being could think that such a name is understandable. If you plan to work with a policy, you might want to store the identifier in a variable so you can use it to refer to the policy with which you want to work. For example:

```
$TP = (Get-ThrottlingPolicy).Identifier
Set-ThrottlingPolicy –Identity $TP -DiscoveryMaxKeywords 15
```

If you create a new policy with the New-ThrottlingPolicy cmdlet, the values from the default policy are inherited. All you have to do is state values for the settings you want to change. Thus, you can do:

```
New-ThrottlingPolicy –Name 'Restricted CAS Access' –RCAMaxConcurrency 10
```

To apply the new policy, you can either make it the default:

```
Set-ThrottlingPolicy –Identity 'Restricted CAS Access' –IsDefault $True
```

or apply it selectively to users:

```
Set-Mailbox –Identity 'David Jones' –ThrottlingPolicy 'Restricted CAS Access'
```