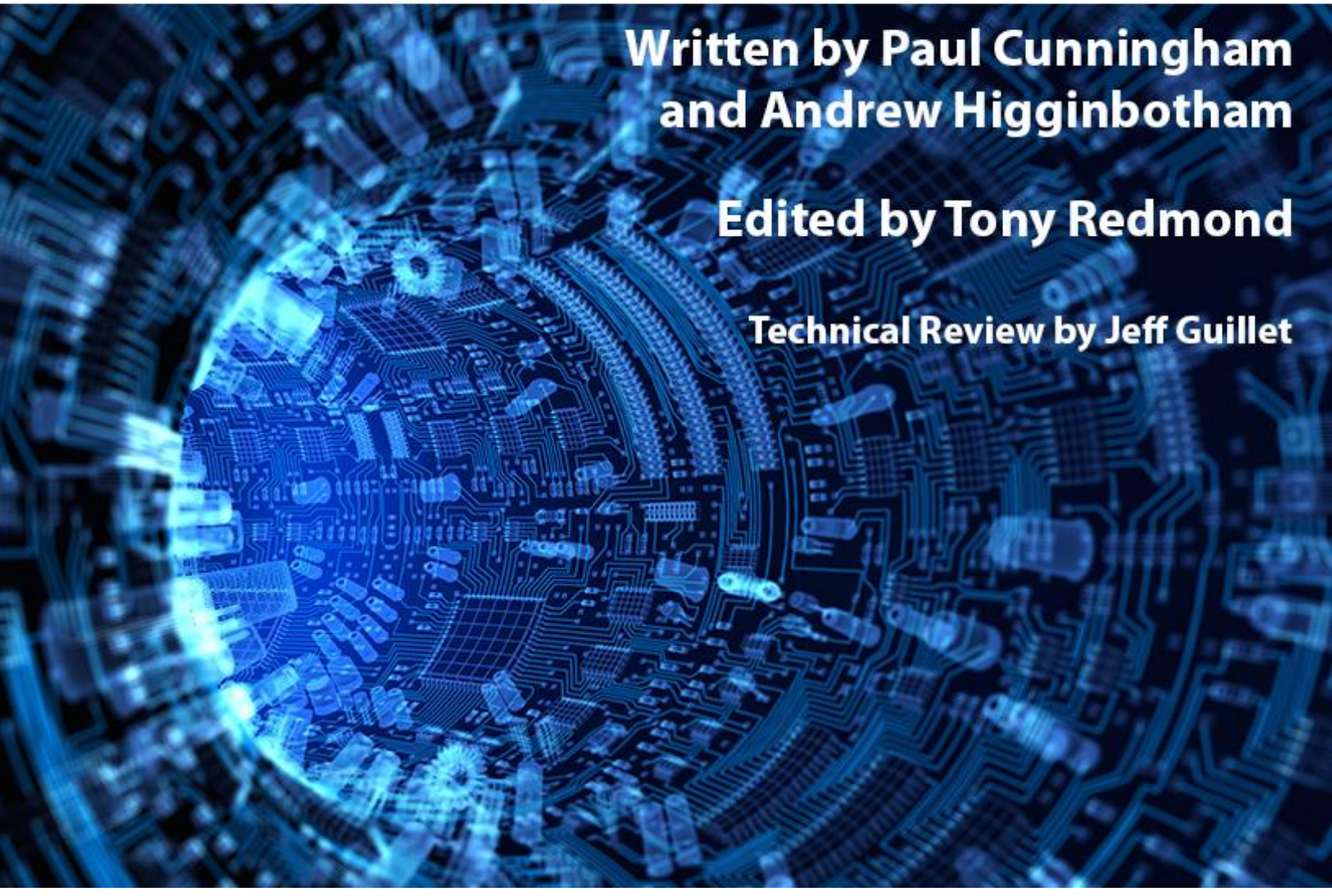
Exchange Server Troubleshooting Companion



Written by Paul Cunningham and Andrew Higginbotham

Edited by Tony Redmond

Technical Review by Jeff Guillet

The Legal Stuff

Published by Paul Cunningham and Andrew Higginbotham

The example companies, organizations, products, domain names, email addresses, logos, people, places and event depicted herein are fictitious. No association with any real company, organization, people, domain name, email address, logo, person, place, or event is intended or should be inferred. The book expresses the views and opinions of the authors. The information presented in the book is provided without any express, statutory, or implied warranties. The authors cannot be held liable for any damages caused or alleged to be caused either directly or indirectly by this book.

Although the two authors are members of Microsoft's Most Valuable Professional (MVP) program, the content of this book solely represents their views and opinions about Office 365 and any other technologies mentioned in the text and is not endorsed in any way by Microsoft Corporation.

Please be respectful of the rights of the authors and do not make copies of this eBook available to others.

First edition. Originally published in April 2016.

Updates and corrections for this eBook are issued regularly until a new edition is published. This version was last updated on 21 April 2016. You can find information about the changes included in each update through posts to the exchangeserverpro.com blog.

Table of Contents

Foreword

Microsoft shipped Exchange 4.0 in 1996. The first version of the server was pretty rudimentary and few components apart from the ESEUTIL program have survived to be included in Exchange 2016. Even though the software was simpler, the complexity of support remained high at times. This was partially due to a lack of knowledge as everyone was finding out how Exchange worked (or didn't, at times) and it was partially due to the email world as it existed at that time. Migrations were everywhere as people moved off PC-LAN based systems such as Microsoft Mail and Lotus cc:Mail, and interoperability was horrible because SMTP had not become ubiquitous then.

Even if these projects remain painful, we have become better at migrations since 1996 and interoperability between email systems is now a given. The Exchange software has improved too and is more automatic and generally easier to manage. There's a lot more documentation to read and a mass of knowledge to be absorbed from blogs, webinars, conferences, and so on. Some of the blogs are extraordinarily helpful; some are much less so. We've all read posts that are badly written, inaccurate, and based on outdated information.

Even with all the additional information available to us, I think the support challenge remains high. Configurations scale to unheard-of heights using hardware that no one anticipated in 1996. Virtualization creates more complexity for administrators to master. The spread of mobile devices and more functional clients creates new demands on infrastructures and the software is much more complex. Just mastering a single topic such as Autodiscover (especially in a hybrid cloud environment) can take more time than you'd imagine.

All of which means that we still need help to master Exchange, if that is even possible for a single human being. And it's not just Exchange; it's a complete ecosystem of network, hardware, software, clients, and people that has to be understood and managed. Books like this that are founded on the experience gained in the school of hard knocks are especially valuable because they strip away the veneer that often disguises bad administrative practices to approach the topic in a very practical sense. I rather like that and I think you will too.

No book is ever complete. Software evolves. Humans learn. People make mistakes and come up with new solutions and writers suddenly realize that they have missed something important. Don't treat this book as the definitive word on troubleshooting Exchange because it can never be. No author understands your environment or business goals or even the unique brand of "office politics" that exists in your company. The best way to use the advice given here is to put it into context. Question the guidance and make sure it fits your needs and you'll benefit from the exercise.

Above all, remember that Exchange evolves on an ongoing basis. Every update shipped by Microsoft brings its own unique quirks, so be prepared to continue to learn. It's the only way to succeed.

Tony Redmond

April 2016

Preface

Andrew Higginbotham

When Paul Cunningham approached me 2 ½ years ago about writing an Exchange troubleshooting book, I was a bit surprised. When a man with over twenty-times the Twitter followers as you and who runs one of the most popular Exchange Server websites in the world asks to collaborate with you on a project, it's a bit of a shocker. I had published a blog post about Troubleshooting Issues with Client Access Servers and Paul wanted to expand the concept into a full book's worth of troubleshooting information. Due to other projects we each had, it actually took another year for the project to begin. The original idea started off as just a large dump of useful OneNote data acquired over combined years' experience, to be used by those looking for a place to begin when troubleshooting relative Exchange components.

However, as writing began we realized the true value in this book would be the message of WHY the product breaks in the way it does and HOW to go about fixing it. The beauty of an eBook is we can easily add as many hyperlinks to blogs, tools, and Support articles as we feel necessary. This means we can still provide the value of a "OneNote knowledge dump" while simultaneously explaining why the product behaves the way it does, giving the reader valuable knowledge to be used in diagnosing an issue. If you know how a product is supposed to function, you're able to accurately identify when it is malfunctioning and whether the advice you're receiving (either from the internet or another source) is good or poor advice. What we hope to accomplish in this Exchange Server Troubleshooting Companion is to give the reader both an understanding of the product and a tool to be used as a reference when troubleshooting the many complex features of the Exchange Server product.

Whether you're an experienced Exchange professional looking to uncover a nugget of product knowledge you may not have encountered before, or someone entirely new to the product who wishes to become a capable Exchange troubleshooter, we hope this book proves useful. Here is a list of those we feel may benefit from this book:

- New IT Professional expected to manage or support Exchange Server
- Seasoned IT Professional who may be tasked with supporting Exchange Server, though they may not have extensive experience with the product
- Helpdesk staff required to support Exchange Server or various clients used to connect to Exchange Server or Office 365
- IT Professional looking for practical, real-world scenario training material for Exchange Server

Whatever your goals, we hope you enjoy the book and find it useful to your tasks as an Exchange Administrator, Support Engineer, or Consultant. If even one outage or call to Microsoft can be avoided as a result of reading this book, the return on your investment will be instantly rewarded.

We'd love to get your feedback on the book, including any topics you feel we should cover. Please send all feedback to feedback@exchangeserverpro.com.

Paul Cunningham

Early in my career I received some good advice from a senior colleague of mine – when trouble strikes, the most valuable person in the room is the one who can also confidently tackle the unknown. Being an effective troubleshooter is more important than being a walking encyclopaedia of solutions to known issues.

Over 15 years later, that advice has proven itself time and time again. Working for support organizations and in consulting roles continually threw me into the unknown, facing previously unseen problems in unfamiliar customer environments. Applying an effective troubleshooting process and methodology, combined with product knowledge, got me through those situations and enabled me to solve the customer's problem.

With the Exchange Server Troubleshooting Companion I firmly believe we've written an eBook that will help you to solve problems faster, understand why the problem occurred, and learn how to prevent the same problems from occurring in the future.

Thanks to my co-author Andrew for bringing his deep technical experience to this book, Tony Redmond for his editorial assistance, Jeff Guillet for his technical reviews, and to David Wedrat and Chris Brown for checking my work.

Being an MVP, writing the Exchange Server Pro blog, and writing eBooks like this one is very rewarding, but also takes a lot of time and energy away from my family and personal life. I am endlessly appreciative of the ongoing support from my wife Hayley and our children as I spend many late nights and weekends contributing to the Exchange Server community.

Follow Paul @ExchServPro or read his Exchange Server Pro blog.

Andrew Higginbotham

Spending almost a dozen years troubleshooting IT issues has taught me a few things. Empathy, patience, perseverance, overcoming stubbornness, and most importantly respect. Respect for the technology I'm working with and respect for others who have blazed the trails before me. The phrase "standing on the shoulders of giants" comes to mind when thinking of how many times I've relied upon other's blogs and expertise. Before becoming a Microsoft MVP, I started blogging and assisting others online because I decided it was time to "pay my tab" and give back to the community after it had helped me so greatly in my career.

That goal was always at the foremost of my thoughts when contributing to this book. To help others by passing on lessons learned, tips, and knowledge that was strenuously acquired over a decade of working with Exchange. I feel we've written something which will not only help the reader in times of need but also help them grow their understanding of the Exchange product itself.

Thanks to my co-author Paul for getting me in involved in the project, Tony Redmond for helping me navigate the world of technical writing (and spending his time translating from Texan to English), and Jeff Guillet for his technical expertise in reviewing this book. It's been an honor to work with these professionals who are masters of their craft.

I'd also like to thank Timothy Heeney, Alessandro Goncalves, and Brian Day at Microsoft for lending their thoughts to my content and for all the help they've given me with Exchange since I've known them. I'd also like to thank my fellow Microsoft Certified Masters Jedidiah Hammond, Ron He, and Mark Henderson for being excellent Exchange mentors throughout the years. Most importantly I must thank my wife Lindsay for her support and patience for the many long hours spent working on this book and other IT endeavors.

Follow Andrew @Ashdrewness, checkout the Exchange Server community he moderates on Reddit, or read his blog posts at Exchangemaster.Wordpress.com and Ashdrewness.Wordpress.com.

Chapter 1: Introduction

Andrew Higginbotham

When things don't work as planned, how do we react? With surprise or panic or perhaps pessimistic doom and gloom? The optimists amongst us probably take a calmer approach to first scoping and then resolving what's going wrong but even they can be reduced to frustration at times. Working with technology gives us plenty of possibilities to exhibit different emotions and having worked in a support role for over a decade I've seen nearly all of them. Often it's a lack of experience that predisposes individuals to react poorly when a piece of technology breaks, but a lack of relevant knowledge can have the same effect.

You can expect that being armed with the experiences of others combined with relevant technical product knowledge can equip an individual to handle even the most complex troubleshooting scenarios. One can even hope that with practice, reactions become instinctual under duress. In other words, when something breaks, the proper questions to ask and knowing what is the right data to gather to help drive towards the resolution of a problem simply becomes second nature. In my opinion, the mark of a good troubleshooter is someone who can navigate a complex break-fix situation without necessarily knowing the technology itself, but rather trusting the instincts they've built over time.

In a way, that's what we hope to give you with this book. We're never going to be able to deliver a cheat sheet to resolve every possible thing that can go wrong with Microsoft Exchange. The product is too big and complex and the environments in which it is deployed create such a massive matrix of possibilities that it would be impossible to attempt to create such a document. But what we can do is tap into our years of experience troubleshooting Microsoft Exchange, describe some of the most common areas that cause problems, and list the tools that experience have proven to be useful in resolving those problems.

I hope that you find the content of the book not only a useful read but also a tool in of itself to help you troubleshooting Exchange issues in different environments. A decade of troubleshooting Exchange and training others to do the same, including the opportunity to speak on the topic at various conferences, has made me realize that the more knowledge you have when approaching a broken Exchange server, the better. Spending years gathering blog posts, amassing a sizeable collection of tips and techniques in OneNote, and facing all manner of broken messaging systems have hopefully supplied sufficient useful content for readers to keep coming back to use this text as a reference.

But before we dive into analyzing what might happen to different areas of Exchange, we should first discuss the topic of troubleshooting itself. How we obtain and analyze data relevant to the problem that has to be resolved to restore normal operations, how we handle stressful situations, and the steps that can be taken to avoid similar situations in the future.

Know the environment and have a baseline

Sometimes, IT professionals have great control over the IT systems that they manage. We know how many users are connecting to each server, how many disk IOPS are being consumed by different applications, how many iPhones vs Windows Phones are in use, and the software revision of Windows, Exchange, Outlook, and all the other clients and software products that are in use across the environment. We should all be so lucky.

Other times we're working with an unknown environment on a contract/project basis. Maybe to upgrade systems to newer software or maybe to migrate mailboxes or other data to new systems. Or maybe we act in a

support role after a problem has been reported and the extent of our environmental knowledge is what we've learned since picking up the phone.

In all cases, gathering intelligence about the target environment is a vital step to take before attempting to troubleshoot any issue. Much like reading and understanding the assembly instructions for a piece of IKEA furniture is a must do to ensure successful construction (and to avoid a costly argument with your significant other), assembling information about the IT systems and their use before you consider what might be going wrong is a fundamental first step in the troubleshooting process.

This is especially important for performance-related issues. Someone may say "X is slow". Well, "slow" isn't a measurable thing because it's a very subjective assessment that could be influenced by many factors, including that person's experience of how an application works. And it's certainly not a term that is consistently defined among users. Slow compared to what? Yesterday? When I used Outlook 2010 opposed to Outlook 2013? When my mailbox is on an Exchange 2013 server opposed to when the mailbox was on an Exchange 2007 server? When I was connected by Wi-Fi compared to a LAN connection at my desk? We'll cover how to handle many of these variables when I talk about the right questions to ask, but for now you can see that a term like "slow" can mean different things to different people. It is therefore important to establish a performance baseline, as well as a means to measure it against a currently reported behavior. A colleague once said "how will you ever know what bad performance is without knowing what your good performance is?"

The need to have a performance baseline introduces a requirement for monitoring, not necessarily from a particular vendor or solution, but as an ongoing practice. Some sports coaches use the term "winning is a habit" to refer to the culmination of all the little things you do along the way, every day, that determines your success. Those are the things that get you ready for game day. Of course in our case, game day is when the messaging environment goes down due to an unforeseen issue. Being properly prepared can make or break your reputation/performance rating/job security.

Every organization will have its own way to monitor performance. It doesn't really matter what tools you use, as long as you have an understanding of the historical load and performance of your Exchange servers. I make sure that I know the basic characteristics of what's happening to Exchange, including;

- Number of mailboxes
- Messages Sent/Received daily
- Average Message Size
- Expected IOPS per Database/Disk/Server
- Average Disk Latency
- Average Memory consumption
- Average CPU Utilization
- Average connections per client type at the Client Access Layer (typically the Load Balancer)

You probably have your own ideas about good essential signs to monitor and we will certainly cover most topics in the Performance chapter.

Knowing all of this information at the start of a support call would make life much easier. In fact, I would be surprised to receive a support call because the people running the environment are so well equipped with data, I'd wonder why they hadn't resolved the issue on their own yet. But more often I find myself asking for information I can reasonably expect any administrator to have at their fingertips, such as:

- The software version running on servers (including the Cumulative Update/Service Pack/Update Rollup)
- The number of servers/users in the environment

- The timeline of when the symptoms appeared
- Hardware specifications of the affected servers

Most of this information should be already known or is easily obtained.

The software revision level matters because it will determine whether the server is currently supported by Microsoft (more on this shortly when we discuss Supportability) and also because many issues are known and have been fixed by available updates.

Knowing the number of servers/users in the environment determines both the impact of the problem as well as the complexity of the environment. Although I feel that Microsoft's Preferred Architecture yields the most stable and simple deployment at scale, it can be argued that a single virtual Exchange server with 100 mailboxes is simpler or less complex than a 60k seat global deployment based on physical servers running with JBOD storage. I haven't mentioned client type yet but you can assume that environments where all users run Outlook in Cached Mode or Outlook Web App are much easier to troubleshoot than environments with ActiveSync, BES, VDI/thin clients (Outlook Online Mode), Outlook for Mac, and the various mobile Apps.

Understanding the environment ties directly into understanding the scope of an issue, which is one of the more important aspects of troubleshooting anything in life. Will a load balancer outage bring down services globally or only in a regional datacenter? Is a database failure impacting 20 mailboxes or 200? Does that database also house the only Public Folder Mailbox? If so then the outage could have a global impact. As you can see, scope and impact are often directly linked. Therefore, it's vital upon initial issue analysis to determine the scope of an issue.

The timeline of events is one of the most critical pieces of information to know when analyzing an issue. Did the issue happen today? Has it been getting progressively worse over a matter of months? Did it start after a change was made to a component such as a storage controller? Did it start after changing datacenters? Did it start after the servers were last rebooted? What was the reason you rebooted (updates or some other reason)? Did anything else change other than that hardware update you ran? No, nothing else changed? Oh, never mind, looks like there were 20 pending Windows Updates waiting for a reboot…. Unfortunately, I have been in situations where fundamental issues like servers waiting to be rebooted to complete the installation of Windows updates have been present, and more than once sadly. So you can see why having an accurate (making no assumptions) timeline of events can give you the kind of essential information you need to start down the right path of troubleshooting.

Finally, having the hardware specifications can give you an idea of what skill set you'll need to diagnose the issue. If the problem lies with virtualized Exchange servers running on VMware with attached SAN-based storage using Cisco switches, you could potentially need assistance from five different vendors: Microsoft, Cisco, VMware, the server vendor and the SAN vendor. Depending on your role in the company, the coordination of support requests to all of those vendors and the reconciliation of the answers that come back might rest on your shoulders (I hope they're paying you well). Alternatively, you will at the very least need the knowledge and cooperation of several individuals.

Having as much background data about the environment at the ready is extremely useful to resolving issues between intertwined teams. I always love to see customers with detailed network diagrams, SAN/Switch/Server update levels, performance data (historical and current), and configuration data. Having this information almost always results in faster results.

No matter whether you're the individual responsible for all aspects of the Exchange environment or a member of a vast team including coworkers as well as vendors, having a detailed and well-documented understanding of the environment will dramatically increase your chances of a fast resolution and make you look like a rock star. Far too often I see customers not investing the time to properly document their environments. They

either don't allocate time, don't prioritize it, or don't see it as forward movement and so cast the task aside. Yet when sirens are blaring, it's the biggest thing they wish for. Well, maybe right behind more thoroughly tested backups.

Similarly, another common mistake is to think, "well we have Bob, and Bob manages Exchange and Active Directory. He knows the environment like the back of his hand." I call this a knowledge vacuum, because when Bob isn't there his coworkers are left clueless. Some individuals like it that way as it gives them a sense of job security, but it's the job of the IT Director/CIO to ensure that no one person holds the keys to the kingdom. So a properly documented environment, along with performance/monitoring baselines will go a very long way towards keeping you well able to handle any incident that comes along.

Testing

Everyone who has tested something by deploying it into production please raise their hand. It's OK, we've likely all done it in some form or fashion. In fact, if nobody raised their hand then I probably wouldn't have a job and books like this would not be needed. People have a tendency to remember the bad more than the good when it comes to product quality. This is true for any product.

Many Exchange administrators could tell you about an Exchange update that broke search, or an Apple update that filled up their drives with transaction logs because calendar items were processed 100k times, or when a Windows Update broke applications that depended on the .NET Framework. However, they probably don't remember all the instances where an update went smoothly, or added stability to the environment (I've seen more Exchange updates do just that rather than break things). Still, having trust in your vendors won't mean much when an outage occurs that could have been avoided.

So how do we avoid Exchange disasters? Well we can't avoid them all but we can tackle the factors that introduce risk into the environment and change is one of the biggest factors. IT change management is worthy of its own book, but speaking purely from an Exchange perspective, we could still spend chapters discussing the perils of haphazard Exchange changes.

Everyone talks about the horrors of Active Directory schema updates, but since Microsoft launched frequent cumulative updates for Exchange 2013, these updates are now expected every three months. Issues arising from schema updates are actually extremely rare. I've spoken with a Microsoft Active Directory Premier Field Engineer and a Support Escalation Engineer about the topic. Between both, they had seen maybe half a dozen catastrophic failures as a result of failed schema updates. To put that figure in context, that's in almost 25 combined years of experience dealing with the worst of the worst Active Directory support cases. Even so, people continue to fear schema updates not for their failure rate, but rather because of the ramifications of such a failure.

It's much like flying in a plane. Statistics tell us it's the safest way to travel, but more people fear flying compared to riding in a car, even if it is statistically a more dangerous means of travel. However, while many people have survived car crashes, far fewer have survived plane crashes. When dealing with schema updates, the ramifications of failure are much higher as a complete Active Directory Forest recovery might be required. The need to avoid the consequence of failure within production environments is why people build test environments, and it is why Microsoft recommends customers test all updates and changes to Exchange in a lab. Although the employment status of IT professionals are seldom threatened when lab environments are broken, the consequences are more serious when flaws emerge in production.

I commonly hear customers say that they don't have the money, resources, or time to completely duplicate their production environment in a lab. To that I say, few have the money to create a perfect duplication, but it doesn't take a lot of effort and money to back up a Domain Controller and a couple Exchange servers and restore them into a virtual environment. While Microsoft tests every Exchange build in Office 365 as well as

their on-premises environments, they have zero knowledge of how YOUR environment is configured and what software is running in it, including custom security policies, group policies, custom access control lists, anti-virus software, third-party plugins, and that one custom script that has to run or else nobody gets paid on Friday.

Any customization or deviation from using software in the way Microsoft expects you to (and therefore tests for) is a potential liability that must be tested. This is especially important with Exchange 2013 onwards because fewer rollback options exist in comparison to Exchange 2010. In Exchange 2010 a Service Pack was essentially an entire fresh set of bits, which could not be uninstalled. If you install Exchange 2010 Service Pack 2, there is no way of uninstalling it to revert to Exchange 2010 Service Pack 1. Your only option is to install a new server with Service Pack 1 and migrate mailboxes to it. Conversely, Update Rollups were pushed down with Windows Updates and added new features/functionality/fixes. These Update Rollups could be uninstalled (for example, updating from Service Pack 3 Update Rollup 7 to Update Rollup 8, and back to Update Rollup 7 again) which gave administrators the ability to roll back changes that introduced functionality that they didn't desire. For example, a number of Exchange 2010 Update Rollups changed things like cross-site client redirection or RPC client connectivity. In certain environments, the changes led to undesired behavior. Luckily, customers had an exit strategy of uninstalling the rollup. Of course, testing the updates first in a lab would've been ideal as they never had to discover the problem when it appeared in production.

Along came Exchange 2013 to change the way in which Microsoft services Exchange with the introduction of Cumulative Updates. Cumulative Updates behave much like Service Packs in that they're full and complete installations of a brand-new version of the product. They cannot be uninstalled and are expected to require schema updates. This means an Exchange administrator's exit strategy is near non-existent if they discover problems after an update is deployed.

Because Cumulative Updates cannot be uninstalled, you have to install a new server if you did not like the introduced behavior. And if that behavior flows from a schema update, you would find yourself performing an Active Directory Authoritative Restore. The way the new servicing model works highlights the importance of testing in a lab environment especially if your production environment has certain quirks or customizations that Microsoft cannot anticipate. For example, you might have a business process that relies on mail-enabled public folders with customized rules/categories. Or all 50 developers have access to each other's mailboxes, with all of those mailboxes open in Outlook Online mode in VDI because you feel it helps drive collaboration? Or perhaps the marketing department has a frequent need to send 10,000 messages to customers in 10 minute spans, with customized encoding, using a custom transport agent? Congratulations on being unique and creating conditions that Microsoft will likely never test. Every new Cumulative Update will be an adventure for your configuration and you will have to test each update thoroughly to ensure that it works for you.

The importance of ensuring a new update will not break either Exchange or third party functionality is vital. Remember to work closely with your vendors to ensure they have tested their products against the Cumulative Update you plan to deploy. For some reason, transport agents seem to be commonly affected by changes introduced in updates. This is possibly because any change to the code within the transport pipeline will directly affect a transport agent's logic. So third party agents for anti-spam, anti-virus, archiving, disclaimers, or routing must be validated against the new code before an Exchange update is introduced into production.

Let's move on to what you should do when the time comes to pull the trigger and start an update. Here are a few tips.

Disable anti-virus software for both file and process scanning. I've dealt with many customers who had anti-virus crash the update process and leave their Exchange server in an inconsistent state. In theory, with a correctly behaving anti-virus process that has all the proper exclusions, you should have nothing to worry about. However, many people do not create the proper exclusions (please see this page) or the anti-virus

software behaves unexpectedly. In many cases where anti-virus proved to be the root cause for an issue, customers adamantly proclaimed that "nothing had changed" or "this software has been running fine for years." However, I usually ask if the anti-virus process received definition or engine updates in that time? Or has the underlying application or operating system changed in that time? Often anti-virus has both file-level filter drivers (seen using the *fltmc* command) as well as network-level filter drivers that live on the TCP/IP stack, so changes to Windows or NIC drivers can affect the behavior of the anti-virus software. In fact, in many troubleshooting scenarios I've known Microsoft to ask that a customer not only disable but also completely uninstall all anti-virus software. This is because even if the software is disabled, these filter drivers still sit on the file/network stack and can cause issues if they misbehave. Sometimes it takes gathering a full memory dump of the system to prove to the customer that even though it's disabled, those drivers are still using/affecting system resources.

Know where logs are located. Issues can still happen during the setup process and when they do you need to know what to do. The first thing I recommend is looking through the setup logs on the system root which typically provides some clues to point you in the right direction. Historically, if your system has crashed/hung during setup, you'll find yourself in what I call an in-between "Exchange Purgatory" state. Meaning some components will either be installed or updated, while the rest are at the original version. In the old days you would either have to start fresh or delete the installation watermark entries from the registry to force Exchange to continue with the install. While that still may be required, Exchange setup has become pretty sophisticated over the years and Exchange 2013 will typically just let you run the program again to a successful completion.

Make sure your permissions are correct. Another common culprit that causes Exchange to crash during updates are missing or inaccurate permissions. Personally, I like to open an elevated command prompt (Right-Click>Run as Administrator) to run Setup.exe and update Exchange. Usually you'll get an immediate failure if permissions are the problem but getting 80% through an update only to have it fail because you (or the process you are running) doesn't have the necessary permissions is beyond frustrating.

Put the server in the right state. As part of Managed Availability, Microsoft introduced Server Component States in Exchange 2013. Managed Availability is composed of a large number of components that aim to ensure your Exchange environment stays available, functional, and healthy. Specifically, Server Component States are groupings of the key features of an Exchange server, which are either in an Active or Inactive state. These states can be altered by various requestors, such as Maintenance, Deployment, or the HealthAPI. When an Exchange update process begins, it requests all components go into an Inactive state so the server transfers work to other servers and won't take any new workload on. Eventually, the services are fully taken offline to allow the update process to update files. The requestor is typically Maintenance or Deployment. Unfortunately, when an update fails for any reason, these components are not always restored from an inactive state and you'll find yourself having to run *Get-ServerComponentState <ServerName>* to verify all states are "Active". If they are not, you may have to manually enable them either via the Exchange Management Shell or via the registry. The most important thing to understand here is that the same requester must be used to set a component state as Active as the one that originally set it as "Inactive". For example, if you run:

Set-ServerComponentState <ServerName> -Component hubtransport -State inactive -Requestor maintenance

to disable Hub Transport functionality, then you would have to run:

Set-ServerComponentState <ServerName> -Component hubtransport -State active -Requestor maintenance

to re-enable it. So typically, re-enabling these states and restarting the relevant Exchange services is all that's needed to restore functionality after a failed update.

Protect yourself. My last piece of advice, as I would hope would be common knowledge, is to ensure that a properly tested full Exchange-aware backup is taken before you begin to install an update or make any critical changes to the system. This will be covered in greater detail in the Backup/Disaster Recovery chapter.

Supportability and Staying Up to Date

Microsoft changed the rules of the serviceability game by introducing the cumulative update servicing model with Exchange 2013 to replace service packs and rollup updates. Cumulative Updates are effectively like Service Packs in that they cannot be uninstalled and usually include Active Directory schema updates (though they may not always contain them, you should expect that they will).

The support lifetime for a cumulative update is as follows: "*A CU will be supported for a period of three (3) months after the release date of the next CU. For example, if CU1 is released on 3/1 and CU2 is released on 6/1, CU1 support will end on 9/1.*" At first reading, this statement shocked many Exchange administrators. In their minds, the new model meant that unless they stayed within 3 months of NewVersion-1, they would be unable to call into Microsoft Support for issues encountered in their production environment. I often say you should never have a production workload for which you don't have an escalation path (a statement I feel many IT Directors would agree with), so you could imagine the state of panic this induced in some.

However, this is not necessarily what this statement meant and in my opinion it could've been much better worded as it caused much confusion, mostly because Microsoft didn't clearly define what "Supported" means. First off, I recommend you watch this extremely helpful Ignite 2015 session from Exchange MVP Paul Robichaux and Microsoft Program Manager Brent Alinger "Servicing Microsoft Exchange Server: Update Your Knowledge". In it, they clearly call out what Microsoft really means when they say a product is "Supported". Let's look at it from each angle.

First off, just because something works does not mean it's supported. Also, just because Microsoft hasn't explicitly come out saying something is not supported, does not mean it is supported. As Exchange MVP Nathan O'Bryan once told me, "*There's no warning signage on a sports car that says not to drive it into a brick wall going 100MPH, but that doesn't mean you should do it*!" So just because Microsoft hasn't come out saying they don't support Exchange, SharePoint, SQL, and Lync on the same server it doesn't mean it's a supported (or wise) configuration. Sadly, I've seen several customers who were used to running an integrated environment on older Small Business Server configurations decide that similar integration was possible with newer software and attempted to do this with disastrous results. Next, Microsoft supporting something means that they have actually tested it and validated that everything works. So if you're unable to get something to work that Microsoft officially supports, it means that an environmental, configuration, or extremely oddball issue exists that is likely to be classified as a bug. But the fact that they support it means Microsoft will help you drive to a resolution.

Lastly, the Microsoft Product Lifecycle is generally considered the bible of Microsoft support timelines. As a general rule you'll find that Microsoft provides ten years support for a product after it's released, but if a subsequent Service Pack is released then the RTM code is only fully supported for a year after the Service Pack release date. This is because RTM is treated as if it were the first service pack, so it's supported for 1 year after the next service pack appears. This logic is fairly easy to understand for service packs, but where do Cumulative Updates and the N-1 Support strategy fit? As it turns out, this is the big difference between Serviceable and Supportable (Serviceable not actually being an official term by Microsoft). Being within a product's "Supported" lifecycle doesn't necessarily mean you'll be issued a fix for a bug or undesirable behavior. For that you need to be within the N-1 window, assuming the fix isn't already in the latest release (N).

To clear up what I think is the biggest misconception that customers have about Microsoft's support policy, if you're not within the N-1 window, or even if you're not within a product's lifecycle, it doesn't necessarily mean they're going to hang the phone up on you. This all really depends on both the scenario that you've encountered as well as whether you have a Premier Support contract.

There's two main ways in which you can engage Microsoft Support; by picking up the phone as a regular customer and using a credit card for a one-time incident based fee or, by having a Microsoft Premier contract where you have purchased a pre-determined number of support hours for use at your discretion. It's widely understood that the level of support you receive via Premier Support is higher than normal phone incident-based support and the length to which the support engineers will go to investigate and research a problem is greater than exists with "normal" support. In some cases, this is because you have already paid for the hours that the Microsoft support engineers are using and they are happy to let you spend those hours how you wish. In my experience, I've found that if you're a Premier customer, you'll never be told something isn't supported and then have the engineer abruptly terminate the call. Microsoft may tell you an update is required or that the software you have lies outside a products support timeline, but they will still attempt to get the issue to resolution. A regular customer without a Premier contract calling in paying a one-time fee to get an issue resolved is going to have much less success, mainly since by simply calling in, Microsoft is likely already losing money. That one-time fee really doesn't go that far in terms of Microsoft profitability.

As I said, it also depends on the situation itself. If your database will not mount due to it being in a Dirty Shutdown state, Microsoft isn't going to drop the call just because you're running software that is three Cumulative Updates behind. After all, the level of the software has nothing to do with your database being dismounted. On the other hand, if you call in with performance issues or problems related to client connectivity, they're probably not going to spend much time looking at the issue before asking that you get to the latest update. Some customers will then look at the public Knowledge Base articles for the updates and discover that the issues they're experiencing are not specifically called out. It's important for customers to know that Microsoft does not publish every fix included in a cumulative update. Instead, only fixes as a result of higher profile customer cases are usually publicized. In many situations, Microsoft Support has insider knowledge (as you would expect) as to which fixes are included in an update. However, at times even they don't know about every fix or enhancement that development put into the code. Being at the latest update will ensure you have all the latest fixes and enhancements the Product Team has to offer, which could very well resolve your issue.

Knowing how the support experience will unfold after you contact Microsoft support will not only set your expectations but may ultimately dictate your organization's update cadence. Although Microsoft won't give you the cold shoulder for being a few updates behind, the change management policies operated by some companies prevent them from deploying an update in a reasonable amount of time just to resolve a support incident; so being as close to N as possible will be ideal.

Speaking of factors controlling update cadence, you have to consider third party applications. Not all applications will support every Cumulative Update, so before moving Exchange forward, verify supportability with your vendors. This applies to Transport Agents, backup software, anti-malware, stubbing/archiving solutions, and so on. Another caution against staying too far out of date is that the updates provided for third party software may not necessarily be tested against older updates of Exchange. So updates to a vendor package could introduce instability in an outdated environment. Even with dated Microsoft software, there's certainly no guarantee they're testing new code against old. For example, on more than one occasion an Exchange update has broken XP clients (also this example) because Microsoft didn't perform the same level of testing against a legacy product as they do against currently supported code. Of course we should be reasonable in what we expect of Microsoft as they can't test against every single use case for products, some of which have never been heard of by the Microsoft testers.

Being an effective troubleshooter

In some ways effective troubleshooting is an art form, and not just in the realm of IT. Doctors taking input, determining whether that input is trustworthy they analyze data and compare it against a knowledge base as well as expertise they've spent years building. Constantly, they're learning new things in an effort to stay relevant and a valued asset to their team. Mechanics are much the same and a lot of faith is put into their ability to correct a problem and leave our family riding safely at 60MPH. In both professions, a simple mistake or careless act has an extremely high risk factor associated with it. As IT professionals, we can probably rest easier knowing the stakes aren't quite as high for us, but many of these other characteristics also apply.

When something goes wrong there's basically three paths forward; things get better, things get worse, or they remain at their current state. We want to improve our troubleshooting skills so the path to improvement is quicker. If we don't succeed we'd like to make sure we don't make things worse. Making another parallel to doctors, the phrase "first, do no harm" are words many of us should live by.

In college, some friends and I joked about shoot-from-the-hip troubleshooting and came up with the phrase "troubleblasting." As in, "Andrew doesn't troubleshoot, he troubleblasts!" While it's a pretty campy/lame joke, I use it all the time with coworkers and trainees because it conveys an important message. Don't be a troubleblaster by throwing every fix you can think of at a problem, not documenting changes, or failing to note what worked and what didn't. Don't dive down a deep-dark rabbit hole where you're desperate because nothing is working, your boss is yelling at you and before long you're just throwing any half-baked solution against the server just to see what sticks. Taking suggestions and purposed solutions from the dark corners of the internet, thinking that just because someone has a website that means what they posted was supported or correct. After all, anyone can start up a free blog on various platforms and start to share their experience with the world. No one tests bloggers for their intelligence, so don't believe everything you see on the internet. At the very least, take a backup before trying a solution you found in a blog, even one from a Microsoft or other trusted source. Otherwise, you run the risk of doing something foolish like deleting your transaction logs because you're full on disk space, enabling circular logging, and then running an ESEUTIL /P against your database (more on this scenario in the Backup and Disaster Recovery chapter). All because the internet said so and it was the first suggestion that came up for a search on Bing/Google.

The problem with this swashbuckling approach to tackling an issue is that it usually results in two things: fixes that aren't really fixes but more like Band-Aids or making so many changes in a short timeframe that you're unsure what actually resolved the issue. Often, things get enabled/disabled/installed/uninstalled that actually had nothing to do with the problem with the result that instability or some other weakness is introduced into the environment. A famous example of this is the Windows Scalable Networking Pack.

Because the pack was a common culprit for network/performance issues for a few years (mostly due to poor code from Microsoft and NIC vendors) it became common practice to blindly disable these features as a first step. Unfortunately, these steps were often taken in ignorance of the actual issue and because it's what was always done. This behavior led to many performance issues in Windows because the disabled features weren't able to fulfill the purpose they were designed to. As a Microsoft Escalation Engineer once asked me, "*All these customers that disable ToE, RSS, Offloading, etc. How many of them notice it didn't resolve their issue so they re-enabled it afterwards?*" This is a perfect example of leaving a system in worse shape than when you started. For those interested, Exchange really should have Receive Side Scaling enabled if you want the best performance.

Take your time when troubleshooting an issue, document each change you make and note whether it had an effect. In addition, ensure you allow sufficient time for the change to take effect (AD replication, service restart, application pool recycle, etc.). When you've finished, you can work backwards to remove any changes that you felt did not resolve the issue, In that way you're not introducing unnecessary changes to the environment (remember; change=risk).

Lastly, probably the most important tip with troubleshooting or debugging any issue. Don't let your assumptions immediately become your conclusions. Or to put it another way, don't make your mind up before actually looking at the facts. People have a way of contorting the facts to suit their narrative when they're already so strongly convicted. Working for a hardware vendor most of my career, I've often seen the customer who is convinced the performance issue is with hardware. Of course, we deal with hardware every day and understand that components break, so we're certainly not opposed to the idea of it being the root cause, but we need to see some proof first. In many instances, a customer will demand replacement hardware in the certain knowledge that it has to be the cause and will reject any input from us or Microsoft with hostility because in this customer's mind it can be only one thing; hardware. Sometimes a broken component certainly is a problem, but often it's a configuration issue, or a sizing issue, or a software bug. The point is that an effective troubleshooter will let the facts lead them to the answer, not emotion, not anecdotal evidence, and certainly not preexisting prejudice ("X product was the culprit last time, so it must be this time as well"). You can certainly have a theory, but it can't be baseless. The biggest problem with anecdotal evidence is that it's sometimes right, and it leads people to think that their method is sound. You can't win at the craps tables every night, so go with the safe bet; actual evidence.

Additional reading

Testing

- Microsoft Anti-Virus Exclusion List
- Exchange Setup Logs - Best Practices
- How does Exchange 2007 setup know to resume a failed setup?
- Exploring Exchange Server Component States

Supportability and Staying up to Date

- Servicing Exchange 2013
- Servicing Microsoft Exchange Server: Update Your Knowledge
- Microsoft Support Lifecycle
- Exchange 2013 SP1 Transport Agent Fix (updated)
- Exchange 2013 CU3 causes headaches for OWA on Windows XP
- Shock! Outlook 2010 users on Windows XP experience problems with Exchange 2013 CU7

Being an effective troubleshooter

- Cluster Network Thresholds – A Good Read
- Give Microsoft's Scalable Networking Pack Another Look
- A reminder on real life performance impact of Windows SNP features

Chapter 2: Troubleshooting Active Directory

Andrew Higginbotham

An unhealthy Active Directory environment is an unhealthy Exchange environment. You'd be hard-pressed to find an Exchange expert who would disagree with that statement. Since Exchange 2000 replaced its own directory with the Windows Active Directory infrastructure, an Active Directory skillset has been a prerequisite for those who design, deploy, and administer an Exchange environment. Before we look at the various tools and techniques available for troubleshooting Active Directory issues that affect Exchange, to gain an insight into the issues that might occur, let's first discuss how Exchange interacts with Active Directory.

An Active Directory database is organized into Partitions (sometimes referred to as Naming Contexts), each replicating between Domain Controllers and containing various kinds of data. The partitions are the Schema partition, Domain Partition, Configuration Partition, and Application Partition. Let's consider what Exchange data is contained in each partition.

The Active Directory schema contains the definitions of object classes and attributes used within the Forest. Out of the box (before Exchange has been installed), Active Directory has absolutely no idea what an Exchange Server, Mailbox Database, Database Availability Group (DAG), or Mailbox is because the schema does not define these objects. When the Exchange installer performs a schema update (*Setup /PrepareSchema*), it adds the definitions of the various objects used by Exchange so that the objects become known to the directory and can be manipulated by the management tools. When it is said that Exchange requires a schema update, it means that the Active Directory schema must be updated to allow existing objects to take on new functionality or to accommodate completely new objects.

Because the schema is essentially just a big repository of definitions, it's extremely rare to experience a failure during the update process. From a troubleshooting perspective, I've never found myself spending much time looking at the schema, except perhaps to verify the current schema version after an update (see this post for more information). The most common issues with the schema are permissions or reachability issues encountered while attempting an update. These problems are usually resolved by ensuring that:

- Your account is a member of the Schema Admins group,
- You can communicate with the Domain Controller holding the Schema Master FSMO role
- The Active Directory Remote Server Administration Tools are installed on the workstation where the update runs

The Domain Partition contains all objects created in a single domain and replicates its data only within that domain. This is where you'll find User Objects, Distribution Groups, Computer Objects, and Organizational Units. When an Exchange client (Outlook, OWA, etc.) performs a query against Address Lists, this is the part of Active Directory queried by the client. When an additional SMTP address is added to a Recipient, it's stamped onto the user object in this partition (using the ProxyAddresses Attribute). This Exchange attribute, along with many others are populated on the Active Directory objects when they are either Mail-Enabled or Mailbox-Enabled. While these Attributes are present on all user objects, they are not populated until the user objects are used with Exchange in some way.

From a troubleshooting perspective, Exchange needs access to this data regardless of which Active Directory domain the user objects exist in. This means all domains containing Exchange-enabled objects must have *Setup /PrepareAD* run against it and all objects need to have Inheritable Permissions set to Allowed. To satisfy this requirement, at least one writable Global Catalog server must be deployed in each Active Directory Site where Exchange is installed.

A Global Catalog server is a Domain Controller that contains a subset of all domain objects in the Forest. Because the Global Catalog holds this data, Exchange can perform queries and make changes to objects regardless of which domain the object belongs. If a writable Global Catalog server is not available, Exchange queries will fail and services may not start (more on that shortly). Often in a multi-site environment, when a change to an Exchange Recipient is made, but not picked up by another Active Directory Site, it's this partition that requires updating (I typically use *repadmin /syncall /Aed* in a small lab environment. In a larger environment the /Aed switch may saturate WAN bandwidth). Updates to other Global Catalog servers occur via Active Directory Replication, which is a multi-master replication process designed to propagate changes to Domain Controllers. When Domain Controllers exist in separate Active Directory Sites, this replication normally occurs according to a scheduled interval, unless an administrator forces replication using the *repadmin* command. However, enabling Change Notification on your Active Directory Site Links can speed up this process.

Creations/Modifications that require replication of the Domain Partition include:

- Adding a new Mailbox/Resource/Distribution Group
- Adding a ProxyAddress (SMTP Address) to a Mailbox or other mail-enabled object
- Moving a mailbox to a new Database
- Updating User passwords (Depending upon Urgent Replication behavior)
- Updating the membership of a distribution group
- Enabling the archive for a mailbox
- Updating the quotas assigned to a mailbox or an archive mailbox
- Updating Exchange policies - Retention/OWA/ActiveSync/Unified Messaging/RBAC Policies
- Updating mailbox permissions (Full Mailbox/Send As/Send on Behalf)

The Configuration Partition contains information on the physical structure, services, and configuration of the Forest. This is where the objects that constitute the Exchange organization are held, such as Exchange Server objects, Connectors, Virtual Directories, Mailbox Databases, Public Folders, Database Availability Groups, Active Directory Sites, and various policies. When you make a change to the Exchange configuration, the command you execute manipulates the attributes for objects in this container. Often in a multi-site environment, when a change to Exchange Configuration is made but not yet detected in another Active Directory Site, it's this partition that requires updating (I typically use *repadmin /syncall /Aed*).

Creation/Modifications that require replication of the Configuration Partition include:

- Adding or removing Exchange servers
- Exchange Virtual Directories
- Retention/OWA/ActiveSync/Unified Messaging/RBAC Policies
- Unified Messaging Dial Plans
- Role-Based Access Control (RBAC) setting/policy
- Public Folders (also dependent on Public Folder Hierarchy replication)
- Accepted Domains
- Email Address Policies
- Transport Rules
- Receive/Send Connectors

- Mailbox Databases
- Database Availability Groups
- Enabled Certificates
- Hybrid Configuration

The last partition type is an Application Partition. When talking of Exchange, the most important type of Application Partition is the DNS Application Partition; which is created when you use Active Directory-Integrated DNS Zones. An unhealthy DNS infrastructure means an unhealthy Active Directory environment and usually doesn't bode well for Exchange, especially when the time comes to route messages to other servers or external domains.

DNS issues are broken up into two types; client-side and server-side. Examples of client-side issues include configuring an incorrect DNS server address for the NIC of an Exchange server, a mistake that will lead to failed DNS queries. I often see smaller customers make this mistake by adding public DNS servers on a NIC for redundancy without realizing that this can cause queuing and service communication issues. Server-side issues, for example, include the DNS server not answering queries or not having the proper SRV records Exchange needs to be able to contact a Global Catalog server. Whether it's answering external mail flow queries or providing needed service records for Active Directory communication, a properly functioning DNS infrastructure is critical for a healthy Active Directory and Exchange environment.

Having broadly covered what Exchange data is stored in Active Directory and how it interacts with it, let's now cover various failure points and the tools/techniques used to troubleshoot them.

Using Event Viewer to diagnose Exchange Active Directory Communication Issues

The article of mine I find that I most often provide to others when they're having Active Directory communications issues covers MSExchange ADAccess Application Event ID 2080. When the Microsoft Exchange Active Directory Topology Service starts up, it queries DNS for SRV records to create a list of available and reachable Domain Controllers in both the local and remote Active Directory sites. Figure 2-1 illustrates some sample SRV records held in DNS.

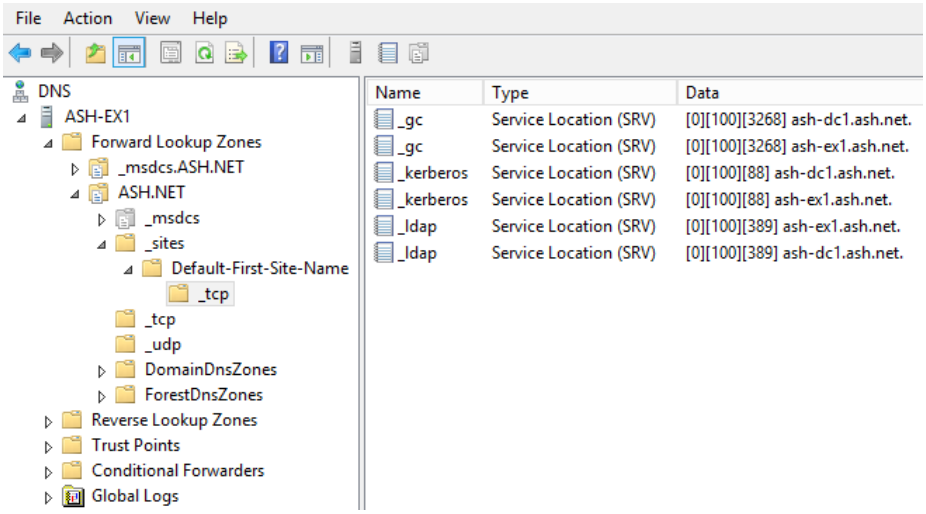


Figure 2-1: SRV records in DNS

After the initial gathering of SRV data, every 15 minutes, MSExchange ADAccess Event ID 2080 is logged (Figure 2-2). This event provides important details about the Active Directory servers available to Exchange.

You can get information about the information reported in the event from Microsoft Knowledge Base article 316300. Much of the content below is taken from that Knowledge Base article. More information about the DSAccess component functionality is available on this page.

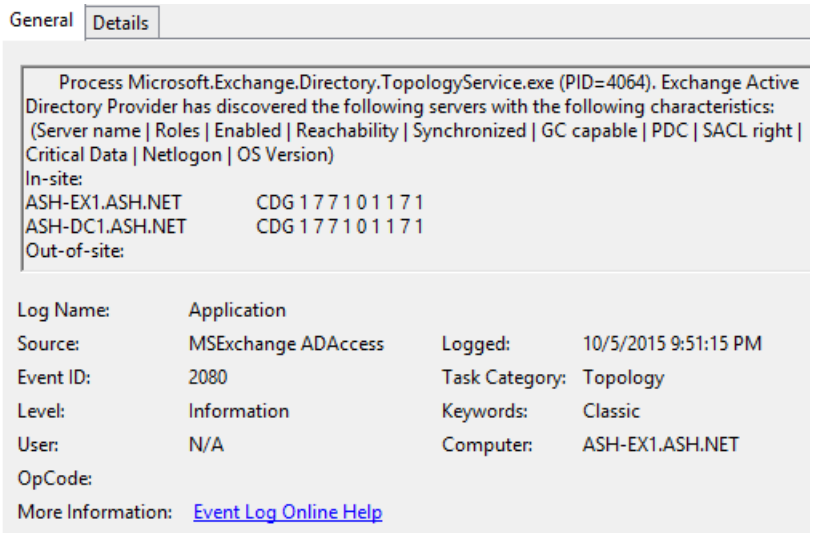


Figure 2-2: Event 2080

In Figure 2-2, we see that two Active Directory servers have been detected in the site. Taking the first (ASH-EX1.ASH.NET) as an example, we can determine:

Server name: Indicates the name of the domain controller that the rest of the data in the row corresponds to. (ASH-EX1.ASH.NET)

Roles: The second column shows whether or not the particular server can be used as a configuration domain controller (column value C), a domain controller (column value D), or a global catalog server (column value G) for this particular Exchange server. A letter in this column means that the server can be used for the designated function, and a hyphen (-) means that the server cannot be used for that function. CDG means that the server can be used for all roles.

Enabled: Either 1 for yes or 0 for no. In this example, we see 1, so we know that the server is enabled.

Reachability: The fourth column shows whether the server is reachable by a Transmission Control Protocol (TCP) connection. These bit flags are connected by an OR value. 0x1 means the server is reachable as a global catalog server (port 3268), 0x2 means the server is reachable as a domain controller (port 389), and 0x4 means the server is reachable as a configuration domain controller (port 389). In other words, if a server is reachable as a global catalog server and as a domain controller but not as a configuration domain controller, the value is 3. In the example shown above, the value 7 in the fourth column means that the server is reachable as a global catalog server, as a domain controller, and as a configuration domain controller (0x1 | 0x2 | 0x4 = 0x7).

Synchronized: The fifth column shows whether the "isSynchronized" flag on the rootDSE of the domain controller is set to TRUE. These values use the same bit flags connected by an OR value as the flags that are used in the Reachability column. The ideal situation is that the values shown in the fourth and fifth columns match.

GC capable: The sixth column is a Boolean expression that states whether the domain controller is a global catalog server. The value (1) indicates that the server is a global catalog.

PDC: The seventh column is a Boolean expression that states whether the domain controller is a primary domain controller for its domain. The value (0) indicates that this is not true.

SACL right: The eighth column is a Boolean expression that states whether DSAccess has the correct permissions to read the SACL (part of the NTSecurityDescriptor) against that directory service. The value (1) means that this is true.

Critical Data: The ninth column is a Boolean expression that states whether DSAccess found this Exchange server in the configuration container of the domain controller listed in Server name column. The value (1) means that the server was found.

Netlogon Check: The tenth column states whether DSAccess successfully connected to a domain controller's Net Logon service. This requires the use of Remote Procedure Call (RPC), and this call may fail for reasons other than a server that is down. For example, firewalls may block this call. So, if there is a 7 in the tenth column (as is the case here), it means that the Net Logon service check was successful for each role (domain controller, configuration domain controller, and global catalog).

OS Version: The eleventh column states whether the operating system of the listed domain controller is running the minimum supported Operating System. Exchange 2007 only uses domain controllers or global catalog servers that are running Windows 2003 SP1 or later. Exchange 2010 and 2013 support Windows 2003 SP2 or later, while Exchange 2016 supports Windows 2008 SP2 or later. A Boolean expression of 1 means the domain controller satisfied the operating system requirements for use by DSAccess.

The value of this data comes from knowing if Exchange can properly utilize the Domain Controllers available in your environment, and is therefore able to communicate with Active Directory. Symptoms that would cause you to look at this counter include:

- Issues starting Exchange services
- Intermittent failures in Exchange Management Shell and the Exchange Admin Console
- Events related to communication issues with Active Directory

If you discover Domain Controllers that Exchange cannot communicate with or that have the improper settings, chances are that the problem lies with a firewall setting, lack of exclusion from Anti-Virus scanning, some setting overridden by Group Policy, a Deny security entry in Access Control Lists, or other permissions issues with the server.

As an example, restrictive Group Policies can result in Exchange being unable to communicate with Domain Controllers since Exchange 2000 (also see KB896703). In the reported scenarios, customers had improperly configured Group Policies which blocked the Manage Auditing and Security log permission to the Exchange Servers group. This is just one example of a permission required by Exchange that is configured during setup when *Setup /PrepareAD* is run, but errant Group Policy settings can overwrite this configuration.

Using Exchange Setup as a troubleshooting tool

When the Exchange Setup program is run for the first time it executes many actions against Active Directory to extend and update the schema, create the Exchange configuration in the configuration partition, create organizational units, a number of security groups, system mailboxes, and to configure the necessary permissions to allow Exchange to use Active Directory on an ongoing basis. Settings must be properly configured to have a fully functioning Exchange environment. You can find a list of the changes made by Setup on TechNet (and this page).

The key modifications made to Active Directory by Exchange setup are:

- The Microsoft Exchange container is created under CN=Services,CN=Configuration,DC=<root domain> if it doesn't already exist
- Permissions are set throughout the configuration partition in Active Directory.
- The Microsoft Exchange Security Groups organizational unit (OU) is created in the root domain of the forest, and permissions are assigned to it.
- Default groups are created within the Microsoft Exchange Security Groups OU if they don't already exist
- Permissions are set on the Microsoft Exchange System Objects container for the Exchange Servers, Organization Management, and Authenticated Users security groups.
- The Exchange Install Domain Servers domain global group is created in the current domain and placed in the Microsoft Exchange System Objects container.
- The Exchange Install Domain Servers group is added to the Exchange Servers USG in the root domain.
- Permissions are assigned at the domain level for the Exchange Servers USG and the Organization Management USG.

The last point is key because many permissions and Access Control Lists are created in the domain that need to propagate to every Active Directory object used by Exchange. If problems occur (for example, if permissions inheritance is blocked to an object) then Exchange functionality may be impaired.

Instead of installing Exchange itself, you have the option to prepare Active Directory beforehand by using command-line setup. The schema can be prepared by using *Setup /PrepareSchema* and Active Directory can be prepared by running *Setup /PrepareActiveDirectory* (instructions for each can be found in the links provided above).

Many are aware that the */PrepareActiveDirectory* option exists and how it is used in deployment scenarios, but few are aware of its use for troubleshooting. If you're experiencing any of the symptoms I described in the "Using Event Viewer to diagnose Exchange Active Directory Communication Issues" section (probably due to Active Directory permissions issues) then running *Setup /PrepareActiveDirectory* or its shortened version *Setup /PrepareAD*, can be a useful troubleshooting step or even provide the resolution itself.

Because customers associate *Setup /PrepareAD* with deployment, changes, outage windows, or any number of other scary words, they might be hesitant about running this command again in production. I've never seen or heard any reports that running *Setup /PrepareAD* breaks anyone's environment or has caused an outage and personally believe that if Setup was to cause an issue, it's a good indication that the environment had serious faults to begin with. In any case, reconfiguring needed Access Control Lists and recreation of Universal Security Groups can often resolve issues caused by that restrictive Group Policies or misguided permissions changes.

A common scenario I see occurs when *Setup /PrepareAD* is run to resolve a suspected Active Directory permissions issue, is successful, yet the issue returns within minutes or a couple of days. This is almost always a sign of a Group Policy which is breaking a required Exchange permission after being reapplied when the machine is rebooted or the policy has refreshed (whichever comes first).

Another common time when running *Setup /PrepareAD* solves problems is when you need to recreate the Arbitration mailboxes after these objects have become corrupt or have been deleted for some reason.

It's also important to note that when running *Setup /PrepareAD*, you should use the same version of the setup binaries as the newest version of Exchange that is installed in the organization.

Lastly, it's important to realize that running Setup/PrepareAD to repopulate Active Directory with Exchange objects might not be able to correct all permissions issues if something exists to block the correct permissions being applied to objects. For instance, if an object or tree has Permission Inheritance Disabled then the needed Exchange permissions will continue to not propagate to those objects until this problem is corrected.

Figure 2-3 shows how you might detect the issue, as the figure shows that no Exchange security groups have permissions for objects (when they're expected to be present) within an organizational unit. Enabling inheritance should typically resolve this.

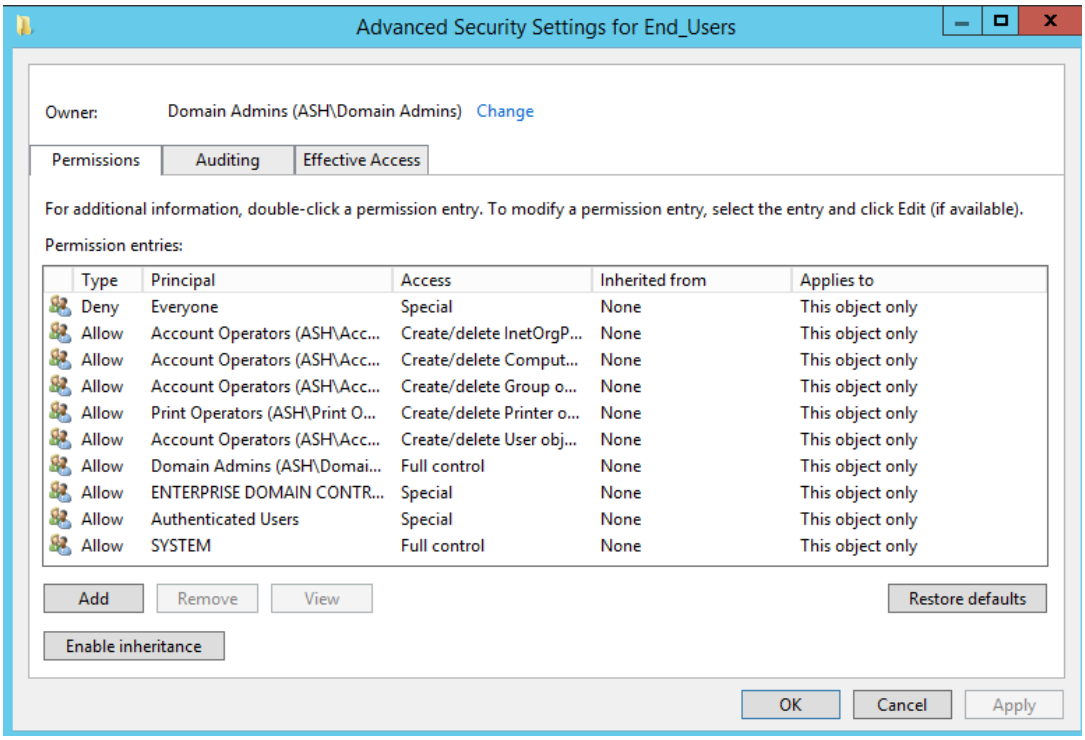


Figure 2-3: No Exchange Security Groups can access an object

Using Active Directory Sites and Services

The Active Directory Sites and Services console is the de facto standard tool to manage your Active Directory Site topology. It is commonly used to define Active Directory Sites by associating them with IP subnets, to create Active Directory Site Links to determine replication paths, and to configure replication frequency. I often find myself using this tool in troubleshooting scenarios for two primary purposes: to force replication to occur when an Active Directory change has not been updated in other sites (Figure 2-4) and when Exchange has communications issues with Domain Controllers.

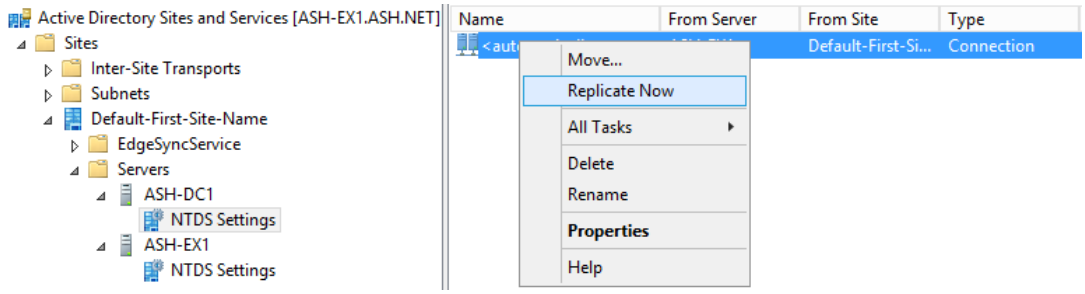


Figure 2-4: Forcing replication using Active Directory Sites and Services

My preferred method to force Active Directory replication is the command line *Repadmin* utility. An Exchange change applied to Active Directory in one site that is not detected in other sites after 15 minutes or so is the typical justification to force Active Directory replication. If replication does not happen, it will lead to situations such as new mailboxes or mailbox databases being invisible to Exchange servers located in other sites, updates applied to objects such as virtual directories not showing up, or schema changes not being available.

When you troubleshoot Exchange Active Directory communications issues, I use AD Sites and Services to verify that Domain Controllers are Global Catalog servers and to ensure that the NTDS settings for all Domain Controllers are being displayed. Figure 2-5 shows what you should see for a healthy domain controller whereas Figure 2-6 illustrates a domain controller whose NTDS settings have been missing for some reason. This scenario would result in replication, authentication, and general Active Directory access issues to occur.

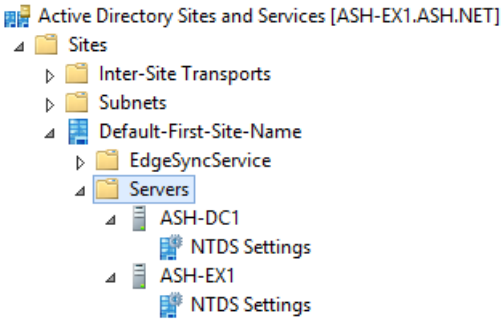


Figure 2-5: Healthy Domain Controllers with NTDS settings being displayed

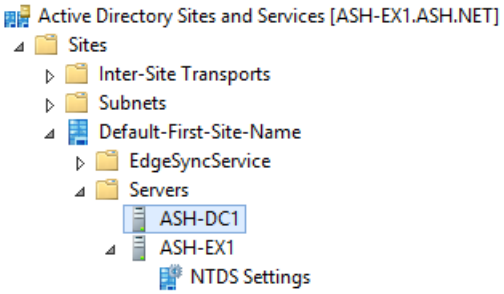


Figure 2-6: Unhealthy Domain Controller with missing NTDS Settings

Another issue I have often come across is where an administrator decommissions Domain Controllers incorrectly. Instead of running DCPROMO to demote a Domain Controller before it is removed, they will simply reformat the operating system. This action leaves remnants of the Domain Controller behind in Active Directory, usually in the form of DNS SRV records as well as a missing NTDS container in AD Sites and Services.

At one time, all you could do to correct the problem was to run a Metadata Cleanup using NTDSUTIL (a low-level AD tool), delete the server object from AD Sites and Services, and manually remove any lingering records of the server that remain in Active Directory. However, starting in Windows Server 2008, all you needed to do is to delete the server object from AD Sites and Services and then clean out DNS. However, this method didn't work in all cases. In these circumstances, you would still find yourself running NTDSUTIL.

Another scenario that requires NTDSUTIL is when an improperly decommissioned Domain Controller holds one of the Flexible Single Master Operation (FSMO) roles. In this case, you also have to seize the FSMO roles using NTDSUTIL or PowerShell.

The common symptoms that show up in Exchange when these issues are present include:

- Occasional failures when running cmdlets or queries
    - EX: "Object not found"
- Exchange service startup issues
    - Services do not start or take several attempts to successfully start
- Slowness of directory queries from Outlook clients
    - Delays of several seconds when querying mailboxes, calendars, etc.
- Mail flow delays

- Several minutes between mail leaving the Outbox and arriving in the recipient's mailbox

Note: Another common cause of service startup and logon failure is Kerberos Time Skew. By default, Active Directory and Kerberos requires a discrepancy of no greater than five minutes between parties participating in Kerberos Authentication. Ensure all devices have a consistent time keeping source and are always within five minutes of each other. In some virtualization scenarios, if proper time synchronization is not being performed, Exchange components and Management Tools will encounter failures.

Using the DCDIAG Tool

The Domain Controller Diagnostic Tool, also known as DCDIAG, is a useful tool for both Active Directory and Exchange administrators. The TechNet article for the command details the various tests that it can perform. Those tests include:

- Verify required DNS records exist for the Domain Controllers in Active Directory
- Verify the Domain Controllers respond to ICMP requests
- Verify the Domain Controllers allow LDAP connectivity by binding to the instance on each server
- Verify the Domain Controllers are advertising themselves properly (as Global Catalog, as Time Servers, etc.)
- Verify Active Directory replication is healthy
- Parse through meaningful Event Logs related to Active Directory health and identify concerns
- Verify SYSVOL and NETLOGON shares are present and able to service domain clients

For a full listing of possible tests that DCDIAG can run, see this Microsoft Blog Post.

In its simplest form, you can just run *DCDIAG* from an elevated command prompt and get useful output. However, if more detailed output is required, here are just a few parameters can be used with the command:

dcdiag /test:dns	*Validate DNS health*
dcdiag /test:testnetlogons	*Validate accessibility/permissions for NETLOGON/SYSVOL shares*
dcdiag /test:replications	*Checks whether last replication attempt was successful*
dcdiag /test:services	*Validates AD-dependent services are running and accessible*

If you suspect Active Directory issues are adversely affecting the Exchange environment, DCDIAG is a quick tool to get a health report of each Domain Controller.

Using the REPADMIN Tool

The Replication Administration tool, also known as REPADMIN, is used to view and troubleshoot Active Directory replication between Domain Controllers. The TechNet article for the command lists the set of supported parameters and functions. Some of the more common parameters used for troubleshooting are as follows:

repadmin /showrepl	*Displays replication status of most recent inbound replication neighbors*
repadmin /replsummary	*Summarizes replication state and health of forest*
repadmin /syncall	*Synchronizes with all replication partners*

The parameter */syncall* is the most commonly used option to force Active Directory replication between domain controllers. I personally find it extremely useful for a multi-site lab environment when I don't want to wait for replication to occur after making a configuration change. The parameter even has its own TechNet article. I typically use the command *repadmin /syncall /Aed* to synchronize all naming contexts across all Domain Controllers in all sites. This step ensures all Exchange changes are replicated, but use it with care in larger environments as the replication activity that results may saturate network bandwidth, especially to domain controllers that serve smaller sites in hub-and-spoke networks.

Note: For additional reading on the topic of Active Directory Replication, especially the differences between FRS and DFSR replication, I recommend this Microsoft post detailing the pros/cons of each.

Active Directory User Token Bloat

Every Active Directory user must be issued an Access Token to be used to access resources in the domain. An access token is a protected object that contains information about the identity and privileges associated with a user account.

When a user logs on interactively or attempts to make a network connection to a computer running Windows, the logon process authenticates the user's logon credentials. If authentication is successful, the logon process returns a Security Identifier (SID) for the user and a list of SIDs for the security groups to which the user belongs. The Local Security Authority (LSA) on the computer uses this information to create an access token — in this case, the primary access token — that includes the SIDs returned by the logon process as well as a list of privileges assigned by local security policy to the user and to the security groups to which the user belongs.

After LSA creates the primary access token, a copy of the access token is attached to every process and thread that executes on the user's behalf. Whenever a thread or process interacts with a securable object or tries to perform a system task that requires privileges, the operating system checks the access token associated with the thread to determine the level of authorization for the thread. This includes Exchange client and server processes which authenticate the user account against a resource.

Token Bloat happens when a user's access token becomes so large that either certain data gets excluded from the token or certain applications cannot handle authentication using the token provided. One Microsoft employee described the situation as follows, "Picture a suitcase filled to overflowing. You managed to close it but some stuff had to get left out." Token Bloat is most often caused by one of the following:

- Users are migrated from one Active Directory domain to another. The Security Identifier history (SIDHistory) is retained from the previous domain to preserve seamless access to resources for the user.
- Users are added to many security groups. The issue is made exponentially worse when those groups are nested into other group memberships.

Among the tools that are available to combat this issue are scripts to detect user token size, formulas to determine token size, as well as improvements in the latest operating systems to better handle this situation. However, it should also be known how this can adversely affect Exchange environments.

A common issue seen with proxying is during migrations to Exchange 2013. The same issue can be expected with Exchange 2016 migrations as it involves the same proxy action. The article "Failures when proxying HTTP requests from Exchange 2013 to a previous Exchange version" describes a scenario where HTTP traffic must be sent from the newer Exchange version (2013 in this case) to a legacy version. As previously mentioned, access

tokens can be used by application processes for authentication purposes and should the tokens be too large, failures can occur.

For one large organization I worked with, after moving their Exchange Outlook Web App (OWA) and Outlook Anywhere namespace from Exchange 2010 to Exchange 2013, some users experienced failures. Upon further investigation we determined that only users with large tokens were experiencing the issue. Useful Exchange logs to help diagnose this issue can be found in the *<Exchange Server Install Path>\Logging\HttpProxy\<Http resource>* logs on the Exchange Server 2013 Client Access server. In this case, the resolution was to either reduce the number of groups the affected users were members of (and therefore reducing their token size) or to make the changes suggested in Microsoft Knowledge Base article KB298844 on the legacy Exchange servers. The relevant extract is as follows:

Increase the MaxFieldLength and MaxRequestBytes entries to the following values. This change requires a restart of the Client Access servers. The recommended value for Exchange 2010 coexistence is 65536.

HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\HTTP\Parameters

MaxRequestBytes DWORD 65536 (Decimal)

MaxFieldLength DWORD 65536 (Decimal)

Because Exchange 2016 will also proxy connections to previous versions of Exchange, and since this issue is caused by Active Directory and not Exchange, I expect to continue to see this issue in the future. Since the proxying of HTTP connections is the key factor here, the problem affects Exchange connections related to Outlook Anywhere, Outlook Web App (OWA), Offline Address Book (OAB), Exchange Web Services (EWS), and ActiveSync.

User Principal Names and Exchange Authentication

A User Principal Name, or UPN, is an Active Directory login format for user identification that resembles an SMTP address. For example, if I had an Active Directory environment where my username is "Andrew" and my domain name is "AndrewLab.local", my UPN would be Andrew@AndrewLab.local. If I install Exchange in this environment, the default SMTP address for my recipients would be Alias@AndrewLab.local. Because ".Local" suffixes are invalid for public email communications, to get around the problem, we can create a new Accepted Domain of "AndrewLab.com" (for example) and apply it to all recipients using an Email Address Policy. With this configuration, I could send emails externally as Andrew@AndrewLab.com, but my default or "Implicit" UPN remains Andrew@AndrewLab.Local (more details here on Explicit vs Implicit UPNs).

This is important because when a user connects to Autodiscover using an Outlook or ActiveSync client, they must provide their Primary SMTP address for Autodiscover to be successful. If this fails, they might be presented the option to use their Domain\Username format (also known as the pre-Windows 2000 User Logon Name) but this is not an ideal user experience. In this scenario, because our primary SMTP address is not a valid UPN, we will not have a seamless logon experience.

This scenario is often encountered, especially inside organizations who deployed Active Directory in the 1999-2004 period when less attention was paid to the goodness of having matching UPNs and primary email addresses. Although the solution is simple, it can also involve some tedious administration effort. The solution is to first define an additional UPN Suffix in Active Directory Domains and Trusts (Figure 1-7).

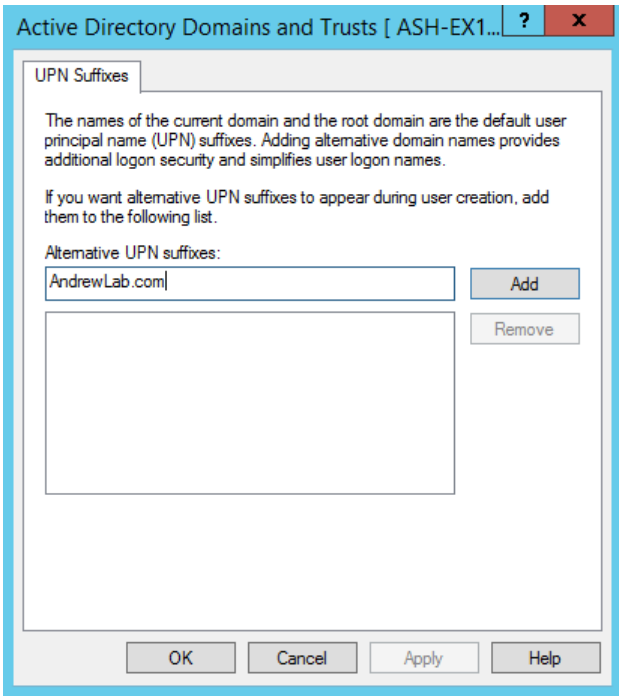


Figure 2-7: Adding a UPN Suffix to a domain in Active Directory Domains and Trusts

After defining the additional UPN, the Explicit UPN for each user account can be changed via a drop-down menu as shown in Figure 2-8. In this case, the user's UPN becomes Andrew@AndrewLab.com and so matches the primary SMTP address to resolve the Autodiscover authentication issue.

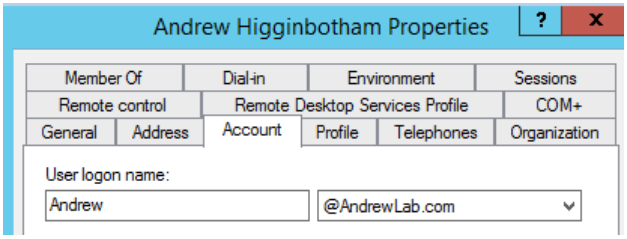


Figure 2-8: Defining the Explicit UPN for a user account in Active Directory Users and Computers

Of course, this action can be scripted to bulk-process a set of user accounts. Several useful scripts exist (Reference 1 Reference 2 Reference 3) that can serve as a basis for this procedure. A common favorite among Active Directory and Exchange professionals is ADModify which offers a GUI and an easy "undo" feature. Note that making sure that the UPNs match primary SMTP addresses is vital for Office 365 migrations. It is also a best practice for on-premises Exchange. If I hear reports of authentication pop-ups in Outlook or ActiveSync clients, my first step is to ensure the user account's primary SMTP address aligns with either their Explicit or Implicit UPN as this ensures both a consistent and seamless login experience.

Active Directory Performance Counters and MaxConcurrentAPI

While relevant Windows performance counters and Performance Monitor will be covered extensively in our "Performance" chapter, let's discuss a few important Active Directory counters which are relevant to Exchange Server health. Table 2-1 lists the relevant Active Directory counters for Exchange called out in TechNet (more information is available on these counters here).

Counter	***Description***	***Threshold***
MSExchange ADAccess Domain Controllers(*)\LDAP Read Time	Shows the time in milliseconds (ms) to send an LDAP read request to the specified domain controller and receive a response.	Should be below 50 ms on average. Spikes (maximum values) shouldn't be higher than 100 ms.
MSExchange ADAccess Domain Controllers(*)\LDAP Search Time	Shows the time (in ms) to send an LDAP search request and receive a response.	Should be below 50 ms on average. Spikes (maximum values) shouldn't be higher than 100 ms.
MSExchange ADAccess Processes(*)\LDAP Read Time	Shows the time (in ms) to send an LDAP read request to the specified domain controller and receive a response.	Should be below 50 ms on average. Spikes (maximum values) shouldn't be higher than 100 ms.
MSExchange ADAccess Processes(*)\LDAP Search Time	Shows the time (in ms) to send an LDAP search request and receive a response.	Should be below 50 ms on average. Spikes (maximum values) shouldn't be higher than 100 ms.
\Netlogon\Semaphore Waiters	The average number of waiters waiting on the semaphore	See Microsoft Knowledge Base article 2688798 How to do performance tuning for NTLM authentication by using the MaxConcurrentAPI setting
\Netlogon\Semaphore Holders	How many threads on average are holding the client semaphore. This is the number of threads trying to get a Netlogon session to a Domain Controller that are Blocked. Blocked could be locked open by a process, network down, etc. When semaphore waiters is non 0, some local process is waiting on lsass for a response and the lsass thread is blocked. This correlates to the MaxConcurrentApi setting.	<2 with Windows Server 2003/2008 Domain Controllers. <10 on Windows Server 2012 or newer Domain Controllers
\Netlogon\Semaphore Acquires	The total number of times that the semaphore has been obtained over the lifetime of the security channel connection, or since system startup for _Total.	Not applicable

\Netlogon\Semaphore Timeouts	The total number of times that a thread has timed out while it waited for the semaphore over the lifetime of the security channel connection, or since system startup for _Total.	Not applicable. Note: Default timeout is 45 seconds.
\Netlogon\Average Semaphore Hold Time	The average wait time for a thread to acquire the semaphore	These values should normally be very quick. Longer hold times mean that a potential bottleneck is occurring.

Table 2-1: Important Exchange Server performance counters

Some of the terms used above may be unfamiliar to most, so let's define them. A semaphore in programming logic is an apparatus for controlling access by multiple processes to a common resource in a concurrent system. Think of this like a microphone in a panel Q&A session. If only two microphones are distributed for Q&A, then only two concurrent individuals can be conversing at once. In Windows Domain Controllers (from Windows Server 2003 to Windows Server 2008) the default number of semaphores is 2. This means a server could handle no more than 2 authentication requests at a time. Each request has a timeout value of 45 seconds, so a server can quickly become overloaded should requests become hung. This behavior is controlled by the MaxConcurrentAPI setting.

Armed with this information, we begin to understand how to interpret the above counters. MaxConcurrentAPI is a setting that applies to both the client and server sides of a NetLogon authentication session, which means that the bottleneck could exist either on the client-side (Exchange Server) or server-side (Domain Controller) of the connection. The symptom to the end user may be repeated authentication prompts, but the root cause is probably that not enough NetLogon authentication channels exist to handle the authentication traffic between the Exchange Server and its reachable Domain Controllers. In most scenarios, it is sensible to increase the MaxConcurrentAPI setting on both the Exchange Server and Domain Controller operating system. This is done by adding (or adjusting) the following registry changes on both systems.

Subkey	Type	Value
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Netlogon\Parameters\MaxConcurrentAPI	REG_DWORD	Higher than the default (2)

After making the change, at a command prompt, run *net stop NetLogon*, and then run *net start NetLogon*.

This Microsoft blog provides some information to help you determine what value is appropriate for your environment. For more detail on how to troubleshoot and diagnose issues related to MaxConcurrentAPI, see this "Ask Premier Field Engineering" blog post. As described in the post, here are the most common symptoms of MaxConcurrentAPI issues:

- Users may be prompted for authentication even though correct credentials are used.
- Slow authentication (may be intermittent or consistent); this may mean authentication slows through the day, or is slow from the hours of 8AM-9:30AM but fine the rest of the time, or any number of scenarios here.
- Authentication may be sporadic (10 users sitting next to each other may work fine but 3 other users sitting in the same area may not be able to authenticate; or vice versa)
- Microsoft and/or 3rd party applications fail to authenticate users
- Authentication fails for all users
- Restarting the NetLogon service on the application server or domain controllers may temporarily resolve the issue

Note: this should not be done as a workaround as you are merely pushing the problem off to another machine!

- Any authentication handled by NetLogon (or Kerberos PAC validation) may experience the same or similar behavior
- Synchronization is not being performed
- Exchange components and Management Tools will encounter failures

Fortunately, this problem should be less common as customers deploy Exchange 2013/2016 and Domain Controllers onto Windows Server 2012 or newer operating systems. This is because the default value of MaxConcurrentAPI is 10 instead of 2 on Server 2012. The five-fold increase in authentication channels should alleviate these issues in most environments.

Note: The most common scenario for performance issues in modern operating systems is if several Exchange Servers in a load balanced array were to fail and the remaining servers had to bear the load of all authentication requests.

However, each environment is different and requires tuning according to the observed workload. The risk of setting these values extremely high (150 for example) is unnecessarily high resource utilization on the systems, so the change should be handled with care.

As one last step to aid in diagnosing NetLogon issues, you can enable NetLogon debug logging. Personally, I find the performance counters listed in Table 2-1 are sufficient to tell me when there's an Active Directory bottleneck but these logs may prove useful in confirming a diagnosis. Another alternative for Exchange servers is to enable Kerberos authentication for client connections, which scales much better for larger or more AD authentication intensive environments.

Having the foundational knowledge of troubleshooting Active Directory will aid us in maintaining a stable and resilient foundation for Exchange. We now move on to Client Access Services and the means in which users access Exchange once they're authenticated by Active Directory.

Additional reading

Overview

- Verify Exchange Server Schema Version
- FSMO Roles
- Preparing Active Directory and Schema for Exchange 2013 Release Preview
- Sites overview
- How Active Directory Replication Topology Works
- Enabling Change Notification on your Active Directory Site Links
- Understanding Urgent Replication
- Are you storing your AD-Integrated DNS Zones in the DNS Application Partitions
- Bad NIC Settings Cause Internal Messages to Queue with 451 4.4.0 DNS query failed (nonexistent domain)

Using Event Viewer to diagnose Exchange Active Directory Communication Issues

- Quick method to diagnose Exchange Active Directory Access and Service Startup Issues
- Event ID 2080 from MSExchangeDSAccess
- Exchange DSAccess / ADAccess and AD site Link
- Exchange Server Supportability Matrix
- MSExchange ADAccess (DSAccess) errors and the "Manage auditing and security" right
- Issues that may occur when the "Manage auditing and security log" permission is removed from the Exchange Enterprise Servers group in Exchange 2000 Server

Using Exchange Setup as a troubleshooting tool

- Prepare Active Directory and domains
- What changes in Active Directory when Exchange 2016 is installed?
- How to recreate System Mailbox, Federated Email and DiscoverySearchMailbox in Exchange 2010

Using Active Directory Sites and Services

- Repadmin
- How to promote a domain controller to a global catalog server
- How to remove data in Active Directory after an unsuccessful domain controller demotion
- Using Ntdsutil.exe to transfer or seize FSMO roles to a domain controller
- Move (Transferring or Seizing) FSMO Roles with AD-PowerShell Command to Another Domain Controller
- Kerberos Time Skew
- Authentication Errors are Caused by Unsynchronized Clocks
- Kerberos Authentication
- Time Synchronization issues and Exchange Management Tools

Using the DCDIAG Tool

- Domain Controller Diagnostics Tool (dcdiag.exe)
- What does DCDIAG actually… do?

Using the REPADMIN Tool

- Repadmin
- Repadmin /syncall
- The Case for Migrating SYSVOL to DFSR

Active Directory User Token Bloat

- What Are Access Tokens?
- Token Bloat Apparently Not Caused By Spicy Food
- Check for MaxTokenSize Problems (Updated)
- Problems with Kerberos authentication when a user belongs to many groups
- What improvements has Microsoft made in Windows 8 and Windows Server 2012 to reduce the number of Kerberos authentication errors due to token bloat and too-large Kerberos tickets?
- Failures when proxying HTTP requests from Exchange 2013 to a previous Exchange version

- "HTTP 400 Bad Request" error when proxying HTTP requests from an Exchange Server to a previous version of Exchange Server

User Principal Names and Exchange Authentication

- Understanding Active Directory Naming Formats
- User Principal Names In AD
- Script: Set-UPN-O365-PSMTP.ps1 – Sets UPNs on-premises and in Office 365
- Changing the User Principal Name (UPN) on Users in Bulk using a GUI tool
- Change UPN
- ADModify

Active Directory Performance Counters and MaxConcurrentAPI

- Exchange 2013 Performance Counters
- The Case of the Mysterious Exchange Server Hang
- How to do performance tuning for NTLM authentication by using the MaxConcurrentApi setting
- Semaphore (programming)
- The Curious Case of MaxConcurrentAPI
- Configuring MaxConcurrentAPI for NTLM Pass-Through Authentication
- Quick Reference: Troubleshooting, Diagnosing, and Tuning MaxConcurrentApi Issues
- Enabling debug logging for the Netlogon service
- Configuring Kerberos authentication for load-balanced Client Access servers
- Recommendation: Enabling Kerberos Authentication for MAPI Clients

Chapter 3: Troubleshooting Client Access Services

Andrew Higginbotham

The role of client access services in Exchange has changed over the years. By "client access services" I'm referring to the server-side components which enable users to connect to Exchange via various clients, including Outlook, Outlook for Mac, Outlook Web App (recently changed to "Outlook on the Web" but commonly known as OWA), and Exchange ActiveSync mobile clients. Based on hardware and operating system limitations over time, the Exchange product team has made several architectural changes to the product to influence how the client access components should be deployed and supported for each release of Exchange. To begin, let's discuss how client access has changed in Exchange Server to aid us in understanding how to properly troubleshoot the current version.

Of front and back ends

There was only a single Exchange Server role in Exchange 2000 and 2003, meaning each Exchange Server installation handled all Exchange workloads. This implementation proved satisfactory for the deployments of the time, but as more types of client devices became available, the need gradually arose to scale-out Exchange servers to be able to handle an increasing number of connections at the client access layer. External Outlook Anywhere (introduced as RPC over HTTP in Exchange 2003 SP1), Outlook Web App (then called Outlook Web Access), and ActiveSync connections came into an environment through shared or dedicated namespaces (namespace being synonymous with a URL resolving to an Exchange resource). These connections would then be routed to their destination mailbox in the environment, which might or might not exist in the same physical location as the ingress point. Depending on the design of an environment or the number of inbound connections, individual Exchange servers needed to be highly scalable at the client access layer.

This need (as well as security concerns) drove the advent of the Front-End/Back-End topology configuration for Exchange 2000 and Exchange 2003 environments. The configuration enabled customers to dedicate Exchange servers for the purpose of handling inbound and outbound client connections (as well as SMTP traffic), so they could scale as needed by adding additional Front-End servers. This ability to split functionality and scale independently was critical at this time because, as x86 servers, Exchange 2000/2003 servers were limited to 4 GB of RAM and CPU processing power was limited. The platform limitations mandated a scale-out approach (in contrast to scale-up by adding more memory/CPU) and the Front-End/Back-End architecture allowed Exchange Administrators to dedicate hardware solely for client access (and SMTP) traffic, freeing up Back-End server resources for mailbox workloads. Common client access services issues encountered in Exchange 2000/2003 were:

- Configuring the Front-End/Back-End topology
- Load balancing the Front-End servers for HTTPS and SMTP traffic
- Requesting and configuring certificates for secured OWA and RPC over HTTP traffic
- Namespace and URL redirection configuration
- Updating Exchange servers in a Front-End/Back-End configuration

Exchange 2007 introduced 64-bit support (removing the 4 GB RAM limitation) as well as the ability to install individual server roles. The various server role installation options were as follows:

Minimum Required Exchange Server Roles for an Exchange Organization

- Client Access Server Role
- Hub Transport Server Role
- Mailbox Server Role

Optional Exchange Server Roles

- Unified Messaging Server Role
- Edge Transport Server Role

At the time when Exchange 2007 was released, the ability and desire to scale-out at the client access layer was fresh in people's minds, so many administrators were extremely pleased at the ability to install only the client access services on a dedicated server.

In an effort to provide similar functionality to a Front-End/Back-End Exchange 2003 configuration (handling both HTTP and SMTP traffic), it was common to collocate the Exchange 2007 Client Access (CAS) and Hub Transport (HT) server roles. Also, because the Mailbox role was considered the "heavy-lifting" server role, these servers were usually deployed with high performing storage and failover clustering. In contrast, CAS and Hub Transport were typically seen as good candidates for virtualization due to the impression that these servers did not require significant resources. This trend continued in Exchange 2010 as the server role and client access architecture remained mostly unchanged. However, this led many Exchange customers to underestimate the importance of the Client Access Server role. Correct Client Access Server sizing in relation to Memory and CPU processing power was still vital to a properly functioning Exchange environment, and failure to provide sufficient resources was a very common issue.

Improperly sized Client Access Servers can provoke many symptoms, including

- Periods of high user activity, such as start of work day or peak season, would result in connection drops and pop-ups in Outlook clients.
- ActiveSync devices are unable to send/receive messages
- Client Access Server system hangs or lockups due to lack of RAM/CPU
- Unsupported virtualization settings used with Exchange Client Access Server Role. For example, using dynamic Memory or no memory reservation and a vCPU:Physical Core ratio greater than 2:1. These scenarios all lead to serious performance issues.

Note: Many of these issues can still occur in Exchange 2013/2016 environments

Along with the option for dedicated server role installations, Exchange 2007 introduced a requirement for Exchange Administrators to become well-versed in certificates/PKI (Public Key Infrastructure). Along with the introduction of Autodiscover, a web-based framework for automatic client configuration, came a requirement for HTTPS-secured Exchange client connections to enable successful Outlook and ActiveSync profile creation.

In addition, Outlook Public Folder-dependent features such as accessing the Offline Address Book (OAB), gathering Free/Busy information, and modifying Out of Office settings were moved to web-based services that also required an HTTPS connection. Although the OAB is now downloaded from its own URL (HTTPS://Mail.Contoso/OAB), the Availability Service (Free/Busy) and Out of Office settings are controlled from Outlook by accessing Exchange Web Services (EWS) via its own URL (HTTPS://Mail.Contoso.com/EWS). This EWS URL is also used when Outlook for Mac connects to Exchange as this client uses EWS instead of MAPI-

based communication used by the Windows version of Outlook. It's important to know that EWS relies entirely on the Autodiscover service for configuration. Even in an Exchange 2007/2010 environment where Outlook users connect to Exchange using MAPI/RPC (not using Outlook Anywhere), access to these web services requires a trusted connection to Exchange using HTTPS. Of course, this means that a certificate that the client trusts must be installed and enabled in Exchange (we'll discuss what's required for a client to trust a certificate later in this chapter).

If I were asked to list the #1 Exchange support call seen in relation to Exchange 2007/2010/2013 client access services, certificates come high up on the list and perhaps even #1. Maybe this is due to a lack of understanding PKI in the industry, or perhaps due to a lack of certificate management features in the Exchange product. For whatever reason, the results were many avoidable Exchange outages. Examples of certificate-related issues in Exchange 2007/2010/2013 include:

- Issues generating a certificate request
- Issues completing a certificate request
- Issues assigning a certificate to a service
- Improper names on a certificate
- Expired certificate
- Incorrect or missing Intermediate certificates
- Autodiscover failures for ActiveSync and Outlook clients due to improper certificate
- Outlook pop-ups related to untrusted certificates
- Free/Busy and Out of Office failures due to untrusted certificate

While the certificate-related tools in the Exchange Admin Center have streamlined certificate management in Exchange 2013/2016, many of the issues listed above can still occur when the fundamentals of certificate issuance and usage are not properly understood. It is for this reason we'll begin this chapter discussing PKI, certificates, and their relation to Exchange.

The basics of Certificates

What Defines Trust

By far, the incorrect configuration and use of certificates is the most common CAS/IIS-related support issue I see from customers. This is odd considering the core concepts are not all that difficult to comprehend. Much like understanding core TCP/IP functionality is critical for any Exchange administrator, I feel core PKI and SSL knowledge is something every IT professional should learn early in their careers. You don't have to be a certificate expert, yet you should understand the 3 golden rules of trust:

- Do I trust the issuer of this certificate?
- Is the certificate expired?
- Is the name I'm using to connect to this service listed on the certificate?

When teaching, I often use the example of a government-issued driver's license as an analogy to certificate trust. If I enter a government building and am asked to provide identification, what would the requesting party be looking for in that identification? By presenting my state-issued Texas driver's license, the requestor has decided to trust the state of Texas. If they do not trust the state of Texas, the ID would be of no use to them for validation purposes. To them, a note from my Mother stating my credibility would be just as unreliable, as it's not a trusted and unbiased source. The next thing they would look for is whether my license was expired, as an expired license is not considered valid. Lastly, the name I'm presenting to them (my name) must match the name listed on the license. Similarly, the picture on the license must be an accurate representation of my appearance to them at that time. So when describing what is important in regards to certificates, the three

golden rules of trust (as I call them) are the same as when describing a valid form of identification. Do I trust the issuer? Is it expired? Is the name I'm using listed on the identification?

Naming

Knowing the requirements for trust will help us to understand which names we need to put onto our Exchange certificate when requesting it. Namespace planning in Exchange 2010, Exchange 2013, and Exchange 2016 are extremely important to ensure successful traffic flow as well as a good end-user experience in an Exchange environment. Part of this is determined when you decide which names you want to include in your certificates. You can technically have a functional solution with only one name on your certificate in a simplistic environment with limited requirements (which also seem to be the environments where less experienced customers are unsure of their options). This is usually the case when a customer does not wish to pay the extra cost for a multi-named (UCC) certificate. For instance:

Example A – Single Namespace

Name on certificate: Mail.Contoso.com

Split DNS Enabled=Yes (Mail.Contoso.com resolves to the Exchange server both internally and externally)

Outlook Anywhere Internal and External Namespace=Mail.Contoso.com

OWA/EWS/OAB/ActiveSync Internal and/or External URL's=Mail.Contoso.com

AutodiscoverServiceInternalUri=HTTPS://Mail.Contoso.com/AutoDiscover/AutoDiscover.xml

External DNS Record for AutoDiscover=None

Internal DNS Record for AutoDiscover=None

In this example, internal Outlook Connectivity and AutoConfiguration will function but external Outlook and ActiveSync AutoConfiguration will fail. This is due to the lack of an external DNS record for AutoDiscover which is present on an installed trusted certificate. Similarly, non-domain-joined Outlook clients connecting internally will also fail. This is due to the lack of an internal DNS record for AutoDiscover which is present on an installed trusted certificate. This is because you don't have AutoDiscover.Contoso.com listed for your certificate, so the process will not be seamless. You will either be greeted with certificate warnings or the connection just would not work (the client AutoDiscover process is further covered in the Clients chapter). Technically speaking, you can get external and non-domain joined Outlook clients to work if you create an AutoDiscover SRV record in DNS for the AutoDiscover service but there's no workaround for ActiveSync clients. Mobile users would be required to manually input the ActiveSync server when creating ActiveSync profiles on their devices. Also, depending on how your device handles certificates, you may or may not be able to connect at all.

Example B – Single namespace using Autodiscover

(I've never actually seen this configuration in production but it would technically work without issue)

Name on certificate: AutoDiscover.Contoso.com

Split DNS Enabled=Yes (AutoDiscover.Contoso.com resolves to the Exchange server both internally and externally)

Outlook Anywhere Internal and External Namespace=AutoDiscover.Contoso.com

OWA/EWS/OAB/ActiveSync Internal and/or External URL's=AutoDiscover.Contoso.com

AutodiscoverServiceInternalUri=HTTPS://AutoDiscover.Contoso.com/AutoDiscover/AutoDiscover.xml

External DNS Record for AutoDiscover=AutoDiscover.Contoso.com

Internal DNS Record for AutoDiscover= AutoDiscover.Contoso.com

Of course, the downside of this configuration is your users have to use HTTPS://Autodiscover.contoso.com/owa to access OWA and I've yet to find a customer who was willing to do that. However, all services would technically function, including Outlook and ActiveSync profile AutoConfiguration.

Example C – Multiple Namespaces with a Subject Alternate Name (SAN) certificate

Names on certificate: Mail.Contoso.com,AutoDiscover.Contoso.com

Split DNS Enabled=Yes (AutoDiscover.Contoso.com and Mail.Contoso.com resolve to the Exchange server both internally and externally)

Outlook Anywhere Internal and External Namespace=Mail.Contoso.com

OWA/EWS/OAB/ActiveSync Internal and/or External URL's=Mail.Contoso.com

Either

AutodiscoverServiceInternalUri=HTTPS://AutoDiscover.Contoso.com/AutoDiscover/AutoDiscover.xml

Or

AutodiscoverServiceInternalUri=HTTPS://Mail.Contoso.com/AutoDiscover/AutoDiscover.xml

External DNS Record for AutoDiscover=AutoDiscover.Contoso.com

Internal DNS Record for AutoDiscover= AutoDiscover.Contoso.com

This is the most commonly used configuration. It allows for full AutoConfiguration of clients via the AutoDiscover service. All necessary records are present internally and externally and the names are on a trusted certificate.

Example D – Wildcard certificate

*Names on certificate: *.Contoso.com*

Split DNS Enabled=Yes (AutoDiscover.Contoso.com and Mail.Contoso.com resolve to the Exchange server both internally and externally)

Outlook Anywhere Internal and External Namespace=Mail.Contoso.com

OWA/EWS/OAB/ActiveSync Internal and/or External URL's=Mail.Contoso.com

Either

AutodiscoverServiceInternalUri=HTTPS://AutoDiscover.Contoso.com/AutoDiscover/AutoDiscover.xml

Or

AutodiscoverServiceInternalUri=HTTPS://Mail.Contoso.com/AutoDiscover/AutoDiscover.xml

External DNS Record for AutoDiscover=AutoDiscover.Contoso.com

Internal DNS Record for AutoDiscover= AutoDiscover.Contoso.com

This method accomplishes all the same goals as Example C, but adds the flexibility (and added cost of the wildcard certificate) of publishing any name prefix you wish to access Exchange resources. Email.Contoso.com, Europe.Contoso.com, or SMTP.Contoso.com could all be used to access Exchange if they were resolvable. However, aside from the added cost, some security professionals have security concerns regarding wildcard certificates. In my personal experience, I've never encountered an Exchange environment that was compromised as a result of using wildcard certificates.

I mention these examples not to tell you how to deploy Exchange (by all means, get a multi-name or wildcard certificate) but instead to explain that in the end, all that matters is that the names you configure in Exchange must be resolvable via DNS to Exchange and are listed on the certificate. You could literally make your Outlook Anywhere namespace "randomseriesofcharacters.contoso.com" and as long as it was on your certificate, it hasn't expired, and the name resolved to Exchange, it would function.

Private Key

Public-Key Cryptography can be a complicated and confusing topic. It involves math, ciphers, and many complex calculations happening extremely quickly. While many books have been written on this topic, an effective Exchange troubleshooter only needs to understand a few key concepts. Let's discuss them at a high level.

Certificates have both private and public keys that are mathematically connected to each other by an algorithm. In practice this means that if a public key is used to convert plaintext to ciphertext, only the private key can be used to convert the ciphertext back to plaintext. However, the same cannot be said of using the public key to decipher information encrypted with the private key. Otherwise, anyone who intercepts the conversation could decipher the data using the easily available public key. So while the public key can be shared freely, the private key must be held secret to everyone but the server where the certificate is installed.

What this all means to Exchange is that an installed certificate is only useful to an Exchange server if it includes the private key. When you export a certificate with the intent of importing it on another Exchange server (which is a common scenario as there's no issue with reusing an Exchange certificate), you must include the private key. When using the Certificates MMC Snap-in (Figure 3-1) to export a certificate, the option to export the private key must be selected. Due to the security implications of the private key falling into the wrong hands, you are required to provide a password when exporting the private key.

Follow these steps to open the Certificates MMC Snap-in in order to export a certificate:

- Open PowerShell
- Type "MMC" <Enter>
- File>Add/Remove Snap-in
- Select "Certificates" and click Add
- Select "Computer Account" and click Next
- Click Finish
- Click Ok
- Certificates for Exchange usage will be in the Personal>Certificates container

Conversely, when using the Exchange Admin Center to export a certificate (Figure 3-2) there is no choice but to export the private key and provide a password, as this is the only way the certificate will be useful for an Exchange server.

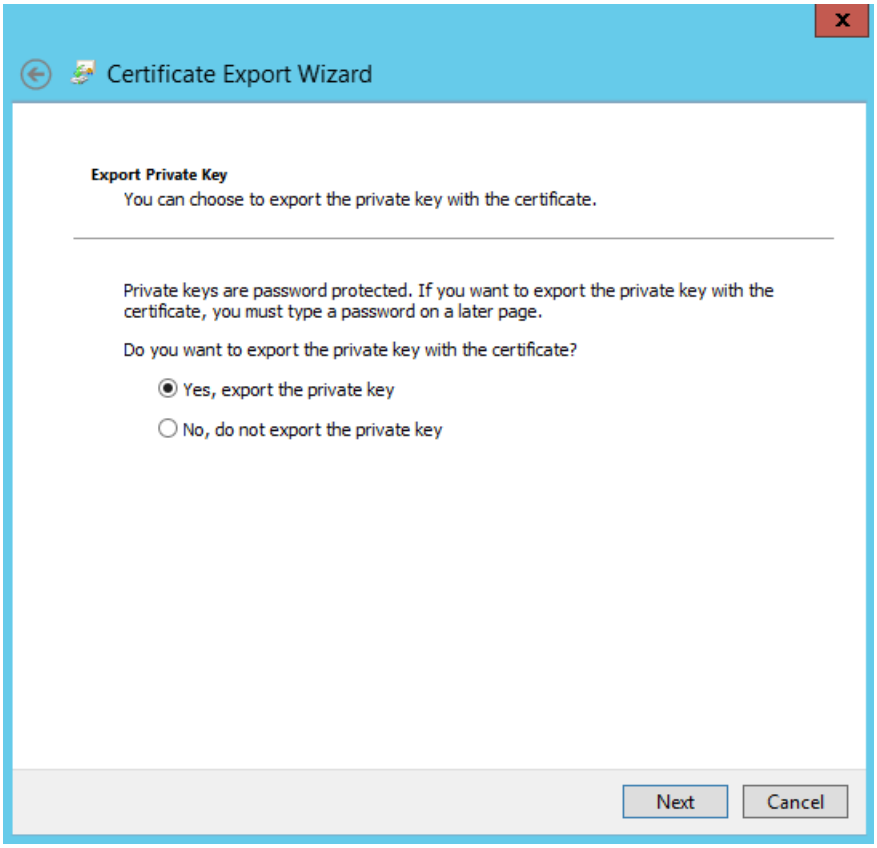


Figure 3-1: Exporting a certificate with the private key using the MMC Snap-in

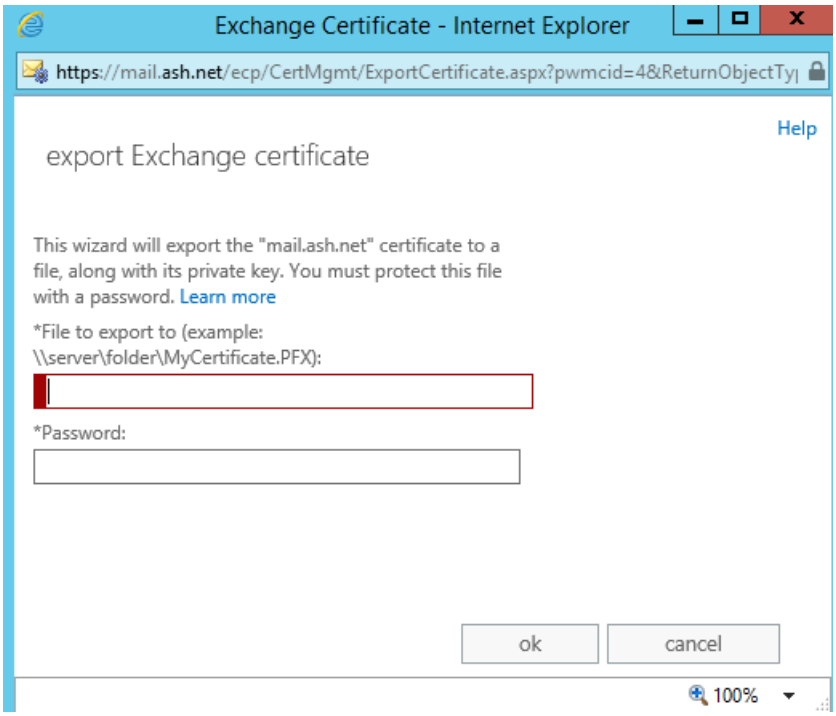


Figure 3-2: Exporting a certificate with the private key using the Exchange Admin Center

A common support issue where the private key (or lack thereof) comes into play is when a certificate is installed without the private key and you're unable to assign Exchange services to it. Depending on the version

of Exchange, you will either be greeted with a "The Certificate is Invalid for Exchange Server Usage" error or the certificate itself will not show up in the Exchange Admin Center. This typically happens as a result of:

- Importing a certificate that does not contain the private key
- Creating a certificate request on one server and either completing the request on a different server or removing the request before completing it on the same server

The process to request an Exchange certificate follows these steps:

- Request an Exchange certificate with the desired names. The Exchange Certificate Wizard will guide through this process.
- Submit the request file to a Certificate Authority
- Receive the certificate file from the Certificate Authority
- Complete the pending certificate request on the same Exchange Server by associating the certificate file with the pending request (this action creates the private key)
- Assign Exchange services to the newly created certificate

Note: Requesting an Exchange Certificate using Exchange Management Shell

Once this process is complete, a valid Exchange certificate that contains the private key will be installed and enabled for Exchange server usage.

Types of Certificates

When discussing the origin of a certificate, you can place them in one of three categories:

- Self-Signed Certificate
- Internal Certificate Authority Certificate
- Trusted Third-Party Certificate

A Self-Signed Certificate is a certificate generated by a server for its own use. This means that server is the only entity that trusts the certificate. A self-signed certificate is typically used when a certificate is required for encryption but not necessarily authentication. In this situation, the Public/Private Key pairs can be used for SSL encrypted communications but client systems will not trust the issuer of the certificate. When Exchange is installed, it creates a self-signed SAN certificate with a Subject Name of <*ServerName*> and the Subject Alternative Name of <*servername.contoso.com*> that is used for encrypted communications between other Exchange servers and client systems. Although an Outlook or ActiveSync client might not connect because they do not trust this certificate, other clients such as Internet Explorer may give a warning, but allow the user to proceed by accepting the warning (Figure 3-3).

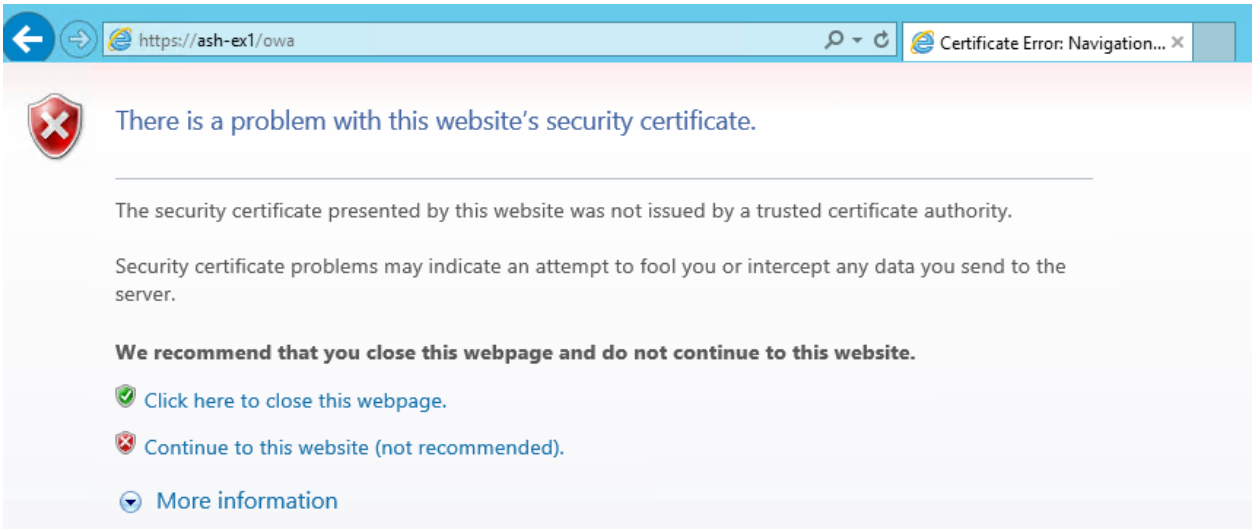


Figure 3-3: Certificate warning from Internet Explorer when using an untrusted certificate

It is important to understand that while the self-signed certificate is not trusted by clients and will fail authentication, the connection is still secured as encryption is still used.

If the self-signed certificate installed by Exchange expires, all you need to do to generate a new self-signed certificate is to run the *New-ExchangeCertificate* cmdlet in an EMS session:

```
[PS] C:\> New-ExchangeCertificate
```

Running the cmdlet with no parameters generates a self-signed certificate with the server's name as the *SubjectName*. This step is typically required when either the default certificate has expired or when someone has erroneously removed the default certificate as an ill-advised way to "clean-up" the installed certificates on the server. When this happens, you will see Warnings/Errors in the Application logs from the Transport Service (Figure 3-4):

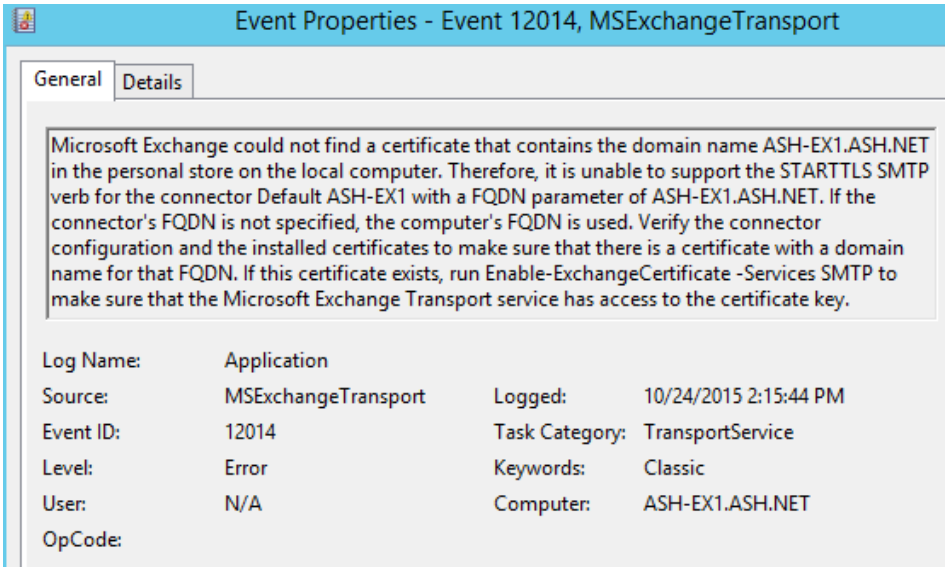


Figure 3-4: Common error when a certificate with the server's name is not present

In this instance, the default Hub Transport receive connector which is used for Exchange Server-to-Exchange Server communication requires a certificate with the server's name to be present. The default self-signed certificate satisfies this requirement but as it has been removed, so TLS is not possible for this receive connector.

Internal Certificate Authority Certificates are requested from a Certificate Authority (CA) controlled by the same entity that requests them (internal), and are typically trusted by all domain-joined clients. This CA has had its certificate chain pushed down to all domain-joined clients via Active Directory and therefore has any certificates issued by it trusted by those clients. While cost (essentially free) is a benefit to using Internal CA certificates, the downside is that clients which are not joined to the domain (such as ActiveSync devices or non-domain joined Outlook clients) do not trust certificates issued by the CA. Instead, the Certificate Authority's root certificate must be manually imported into the local client's certificate store (Figure 3-5).

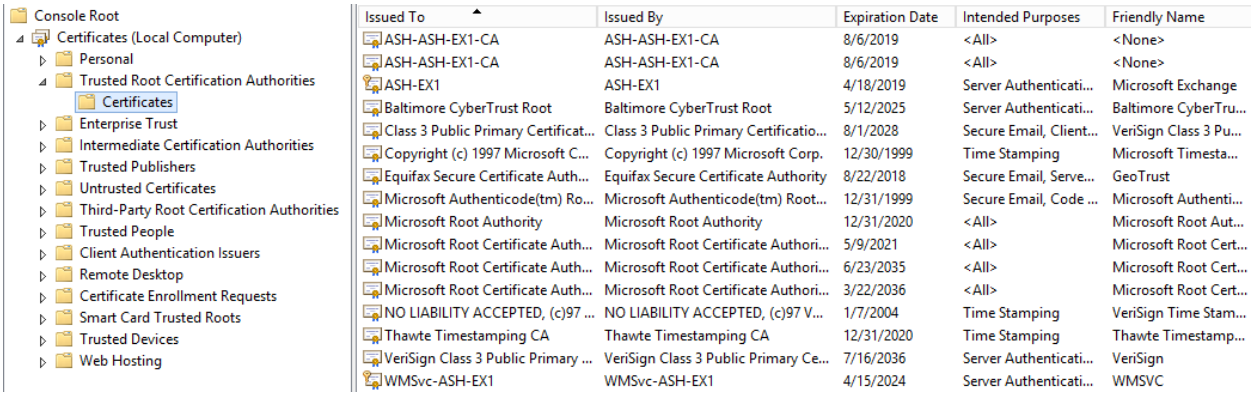


Figure 3-5: Trusted Root Certification Authorities container of a Windows machine

Take note of the Internal CA (ASH-ASH-EX1-CA) root certificate listed in the Trusted Root Certification Authorities container in Figure 3-5.

Finally, a Trusted Third-Party Certificate is issued by a third-party certificate authority which is trusted by most if not all IT systems. Examples of such third-party Certificate Authorities are DigiCert, GoDaddy, VeriSign, Entrust and others. Although these certificates are much simpler to deploy (as all clients should trust automatically them), they can easily cost several hundred dollars per year. The cost is usually determined by the number of names/domains listed on the certificate as well as whether it is a Wildcard Certificate. Windows operating systems trust the Root Certification Authorities (CAs) based on the contents of the Trusted Root Certification Authorities container (Figure 3-5). This container is populated partially upon operating system install, but is later updated either via Windows Updates or manual installation files (additional reference). While Root Certificate Authorities are certainly important, Intermediate Certificate Authorities are equally important. A client must trust not only the Root but also any Intermediate CA's in the certificate's chain. This "Certificate Chain" is often provided when an issuer distributes a certificate to a customer. It's recommended to install this chain not only on your Windows Systems, but when installing a certificate on a load balancer as well.

While I feel we've covered the basics of certificates which will help you be an effective troubleshooter, if you remember nothing else about certificates, just remember:

- Do I trust the issuer of this certificate?
- Is the certificate not expired?
- Is the name I'm using to connect to this service listed on the certificate?

If the answer to any of these questions is "No", that should be the first issue you address. Also, see this great blog post from co-author Paul Cunningham on the topic of Exchange Certificates. It also references several additional articles covering the basics of planning certificates and naming for Exchange 2013.

Note: Wildcard certificates can be used with multiple subdomains of a domain. These certificates will have a Subject of *.domain.com. As an example, https://plus.google.com uses such a certificate. They

are typically more expensive than other types of certificates but have the added flexibility of using the certificate with as many subdomains as you choose. If using a Wildcard Certificate with Exchange, you may be required to modify the Outlook Provider to ensure proper Outlook Anywhere functionality.

IIS Basics and Exchange

The first step to troubleshooting any technology is to understand how its core components function during normal operation. People are often given a set of tools to be used in troubleshooting, but never truly understand how to interpret the data they're looking at. Similar to how using Microsoft's Network Monitor tool will be of little use to someone who doesn't have a solid understanding of TCP/IP, looking at Exchange Client Access logs or IIS data will not prove useful if you do not understand how each of the components interact with each other. Let's begin by looking at IIS (Figure 3-6).

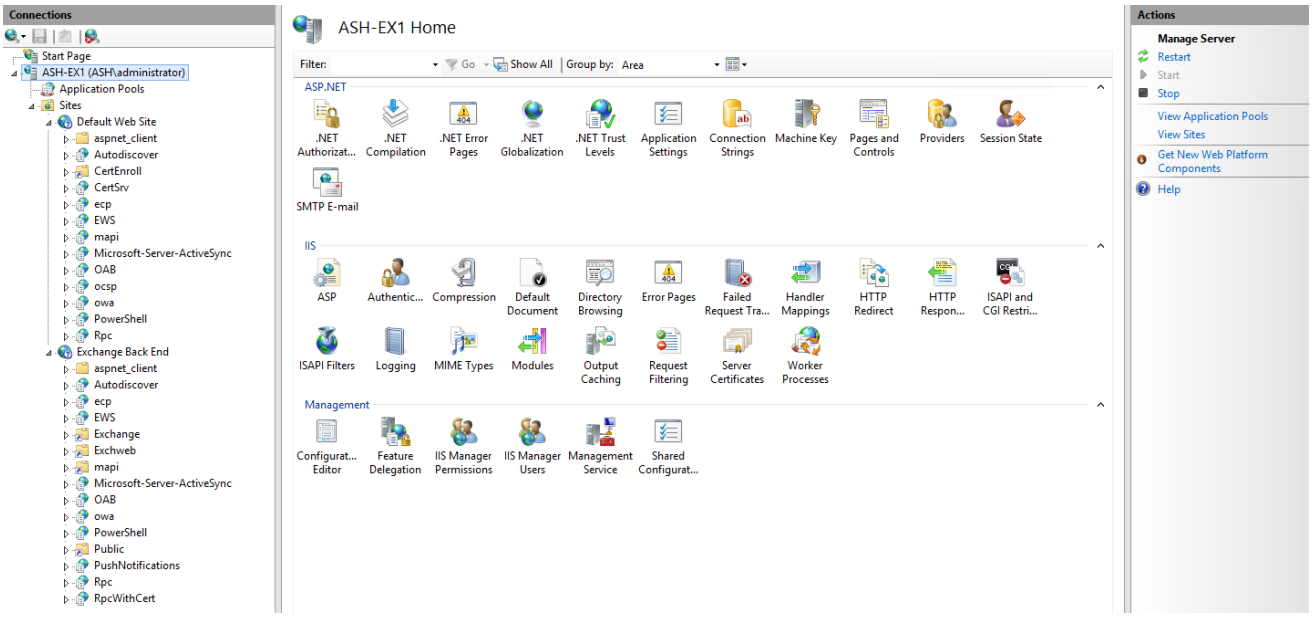


Figure 3-6: Expanded IIS Manager on Server 2012 R2 Exchange 2013 multi-role server

Figure 3-6 shows the IIS Manager on my Windows Server 2012 R2 Exchange 2013 multi-role server. We find the various web sites as well as the Application Pools that correspond to each application such as ActiveSync, PowerShell, or OWA. Because this server is multi-role (it has both CAS and Mailbox Roles installed) you will see two separate Exchange web sites:

- Default Web Site = Client Access Server Role
- Exchange Back End = Mailbox Server Role

The two primary services associated with IIS are the IIS Admin Service (inetinfo.exe) and the World Wide Web Publishing Service (w3wp.exe). To explain it simply, inetinfo.exe corresponds to IIS configuration information whereas w3wp.exe corresponds to each of the various Application Pools. After changing IIS configuration information (such as authentication settings, etc.), the IIS Admin Service will typically need to be restarted. Whereas, if a particular application still isn't updating after you've made a change (like OWA or ActiveSync) then you may need to recycle that Application Pool and at worst, restart the World Wide Web Publishing Service.

However, in many cases it's recommended to simply stop then start the website or recycle the application pool rather than restarting the services or using *iisreset* (Reference-1 Reference-2 Reference-3). This is because it's

possible IIS has not saved the necessary changes in time and those changes could be lost by a forcible service restart. Starting then stopping the websites, recycling the application pools, or using the "*/noforce*" switch for *iisreset* is preferred. However, sometimes killing a service using Task Manager is all you can do as a last resort in a troubleshooting scenario.

Web Sites and Application Pools in Exchange 2013

When troubleshooting IIS, I commonly find myself looking at the Web Site Bindings. These are what "bind" an IP Address, Port Number, Host Name, and (potentially) a Certificate to the web site. Let's look at the bindings using both PowerShell as well as the GUI.

```
PS C:\> Import-Module WebAdministration
PS C:\> Set-Location IIS:\
PS IIS:\> Get-Website

Name             ID   State      Physical Path                  Bindings
----             --   -----      -------------                  --------
Default Web Site 1    Started    %SystemDrive%\inetpub\wwwroot  http *:80:
                                                                net.tcp 808:*
                                                                net.msmq localhost
                                                                msmq.formatname localhost
                                                                net.pipe *
                                                                https :443: sslFlags=0
                                                                http 127.0.0.1:80:
                                                                https 127.0.0.1:443: sslFlags=0
Exchange Back    2    Started    C:\inetpub\wwwroot             http *:81:
End                                                             https *:444: sslFlags=0
                                                                net.pipe *
```

Figure 3-7: Viewing IIS Web Site settings with the Web Administration PowerShell module

Using the series of commands shown in Figure 3-7, I imported the IIS PowerShell Module and queried the bindings of my two web Sites in IIS. I've found that using PowerShell is a very useful way to query this data fairly quickly. It's also useful for when you need to send a customer a set of commands they can run and send the data back to you for analysis. Figure 3-8 shows a few of my preferred information gathering command in action:

```
PS IIS:\> Get-ChildItem

Name
----
AppPools
Sites
SslBindings


PS IIS:\> set-location sites
PS IIS:\sites> Get-ChildItem

Name             ID   State      Physical Path                  Bindings
----             --   -----      -------------                  --------
Default Web Site 1    Started    %SystemDrive%\inetpub\wwwroot  http *:80:
                                                                net.tcp 808:*
                                                                net.msmq localhost
                                                                msmq.formatname localhost
                                                                net.pipe *
                                                                https :443: sslFlags=0
                                                                http 127.0.0.1:80:
                                                                https 127.0.0.1:443: sslFlags=0
Exchange Back    2    Started    C:\inetpub\wwwroot             http *:81:
End                                                             https *:444: sslFlags=0
                                                                net.pipe *


PS IIS:\sites> set-location "Default Web Site"
PS IIS:\Sites\Default Web Site> dir

Type              Name                           Physical Path
----              ----                           -------------
directory         aspnet_client                  C:\inetpub\wwwroot\aspnet_client
application       Autodiscover                   C:\Program Files\Microsoft\Exchange
                                                 Server\V15\FrontEnd\HttpProxy\Autodiscover
virtualDirectory  CertEnroll                     C:\Windows\system32\CertSrv\CertEnroll
application       CertSrv                        C:\Windows\system32\CertSrv\en-US
application       ecp                            C:\Program Files\Microsoft\Exchange
                                                 Server\V15\FrontEnd\HttpProxy\ecp
application       EWS                            C:\Program Files\Microsoft\Exchange
                                                 Server\V15\FrontEnd\HttpProxy\EWS
file              iis-85.png                     C:\inetpub\wwwroot\iis-85.png
file              iisstart.htm                   C:\inetpub\wwwroot\iisstart.htm
application       mapi                           C:\Program Files\Microsoft\Exchange
                                                 Server\V15\FrontEnd\HttpProxy\mapi
application       Microsoft-Server-ActiveSync    C:\Program Files\Microsoft\Exchange
                                                 Server\V15\FrontEnd\HttpProxy\sync
application       OAB                            C:\Program Files\Microsoft\Exchange
                                                 Server\V15\FrontEnd\HttpProxy\OAB
application       ocsp                           C:\Windows\SystemData\ocsp
application       owa                            C:\Program Files\Microsoft\Exchange
                                                 Server\V15\FrontEnd\HttpProxy\owa
application       owa\Calendar                   C:\Program Files\Microsoft\Exchange
                                                 Server\V15\FrontEnd\HttpProxy\owa
application       owa\Integrated                 C:\Program Files\Microsoft\Exchange
                                                 Server\V15\FrontEnd\HttpProxy\owa
application       owa\oma                        C:\Program Files\Microsoft\Exchange
                                                 Server\V15\FrontEnd\HttpProxy\owa
application       PowerShell                     C:\Program Files\Microsoft\Exchange
                                                 Server\V15\FrontEnd\HttpProxy\PowerShell
application       Rpc                            C:\Program Files\Microsoft\Exchange
                                                 Server\V15\FrontEnd\HttpProxy\rpc
file              web.config                     C:\inetpub\wwwroot\web.config
file              web.config.bak                 C:\inetpub\wwwroot\web.config.bak
```

Figure 3-8: Querying IIS settings using PowerShell

The commands shown in Figure 3-8 are executed after navigating to the "Default Web Site" (already done in Figure 3-7) and expose the various Applications and Virtual directories underneath it. Notice how the commands work similar to navigating a folder structure. If I need to go back a level I can simply use "*cd ..*". Alternatively, if I wanted to export this to a text file I could repeat the last command but with a *Format-List* at the end and redirect it to a text file:

```
dir | Format-List > C:\IISOutput.txt
```

This can be useful when comparing information taken from a known working server to what you see on a problematic one. Of course there's also any number of ways this data can be scripted/manipulated/etc. to fit your needs. Now if I go back to the root I can see a list of all the Application Pools in IIS (Figure 3-9).

```
PS IIS:\> cd ..
PS IIS:\> Get-ChildItem

Name
----
AppPools
Sites
SslBindings


PS IIS:\> set-location apppools
PS IIS:\apppools> dir

Name                    State        Applications
----                    -----        ------------
.NET v4.5               Started
.NET v4.5 Classic       Started
DefaultAppPool          Started      Exchange Back End
MSExchangeAutodiscoverAp Started     /Autodiscover
pPool                                /Autodiscover
                                     /Autodiscover/bin
                                     /Autodiscover/help
MSExchangeECPAppPool    Started      /ecp
                                     /ecp
MSExchangeMapiAddressBoo Started     /mapi/nspi
kAppPool
MSExchangeMapiFrontEndAp Started     /mapi
pPool
MSExchangeMapiMailboxApp Started     /mapi/emsmdb
Pool
MSExchangeOABAppPool    Started      /OAB
                                     /OAB
MSExchangeOWAAppPool    Started      Default Web Site
                                     /owa
                                     /owa/Integrated
                                     /owa/oma
                                     /CertSrv
                                     /owa
MSExchangeOWACalendarApp Started     /owa/Calendar
Pool                                 /owa/Calendar
MSExchangePowerShellAppP Started     /PowerShell
ool
MSExchangePowerShellFron Started     /PowerShell
tEndAppPool
MSExchangePushNotificati Started     /PushNotifications
onsAppPool
MSExchangeRpcProxyAppPoo Started     /Rpc
l                                    /RpcWithCert
MSExchangeRpcProxyFrontE Started     /Rpc
ndAppPool
MSExchangeServicesAppPoo Started     /EWS
l                                    /EWS
                                     /EWS/bin
MSExchangeSyncAppPool   Started      /Microsoft-Server-ActiveSync
                                     /Microsoft-Server-ActiveSync
OCSPISAPIAppPool        Started      /ocsp
```

Figure 3-9: Listing Application Pools in IIS

Note: The Default Web Site has bindings of 80 and 443 for HTTP and HTTPS, respectively, while Exchange Back End has 81 and 444, respectively. When a client makes a connection to Exchange using HTTPS it's connecting to the Default Web Site which proxies the connection back to the Exchange Back End web site. Do not change the bindings on the Exchange Back End website lest you want to break all HTTP proxy functionality from the CAS to Mailbox role. Note that the Default Web Site will use the certificate which has been imported and enabled for the IIS service via the Exchange Management Tools. Conversely, the Back End website uses the self-signed Microsoft Exchange certificate which is installed upon Exchange installation.

Alternatively you could use the Exchange Management Shell for some of these commands but you might find that the PowerShell IIS Module gives you a bit more flexibility. Now to look at these settings in the GUI may

seem easier but it does require a bit more mouse clicks to get the same data. In Figure 3-10 we've expanded Sites and right-clicked the Default Web Site.

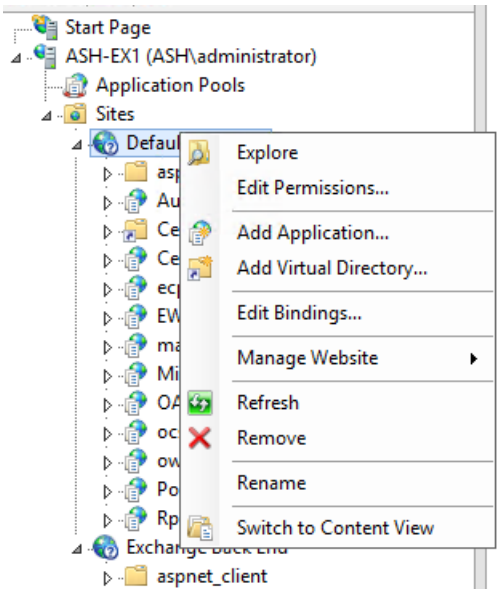


Figure 3-10: Viewing the properties of a web site in IIS Manager

After selecting Edit Bindings, we're presented with the IP address, port number, and Host Name binding information on this web site (Figure 3-11). By selecting HTTPS and clicking Edit, you can view the assigned SSL certificate.

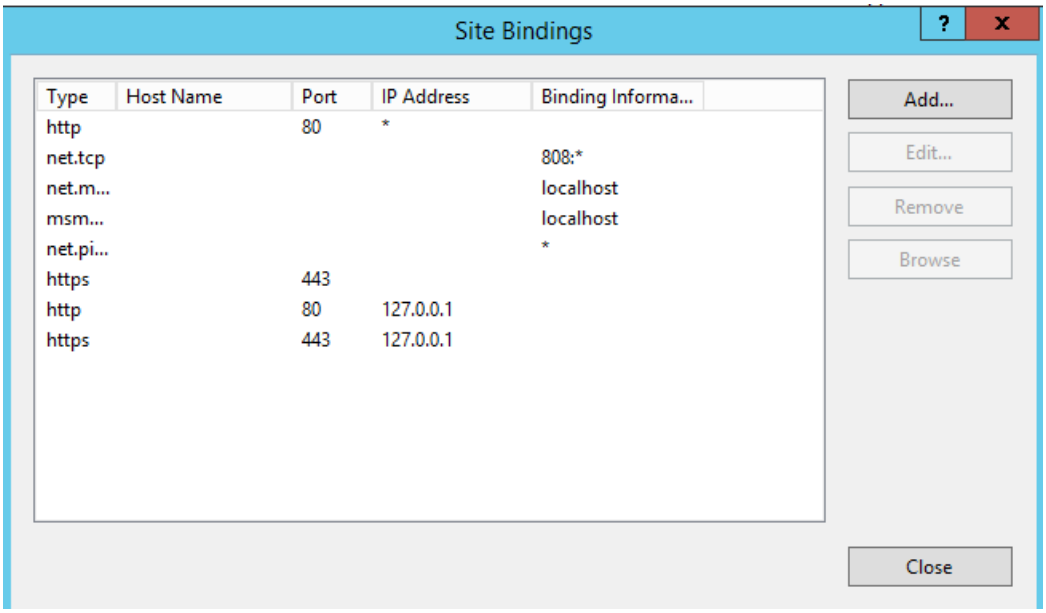


Figure 3-11: Viewing the site bindings in IIS Manager

To view Application Pool properties in the IIS Manager, navigate to Application Pools below the IIS Server object (Figure 3-12). Here you can view Application Pool state, the version of .NET it runs, and its identity.

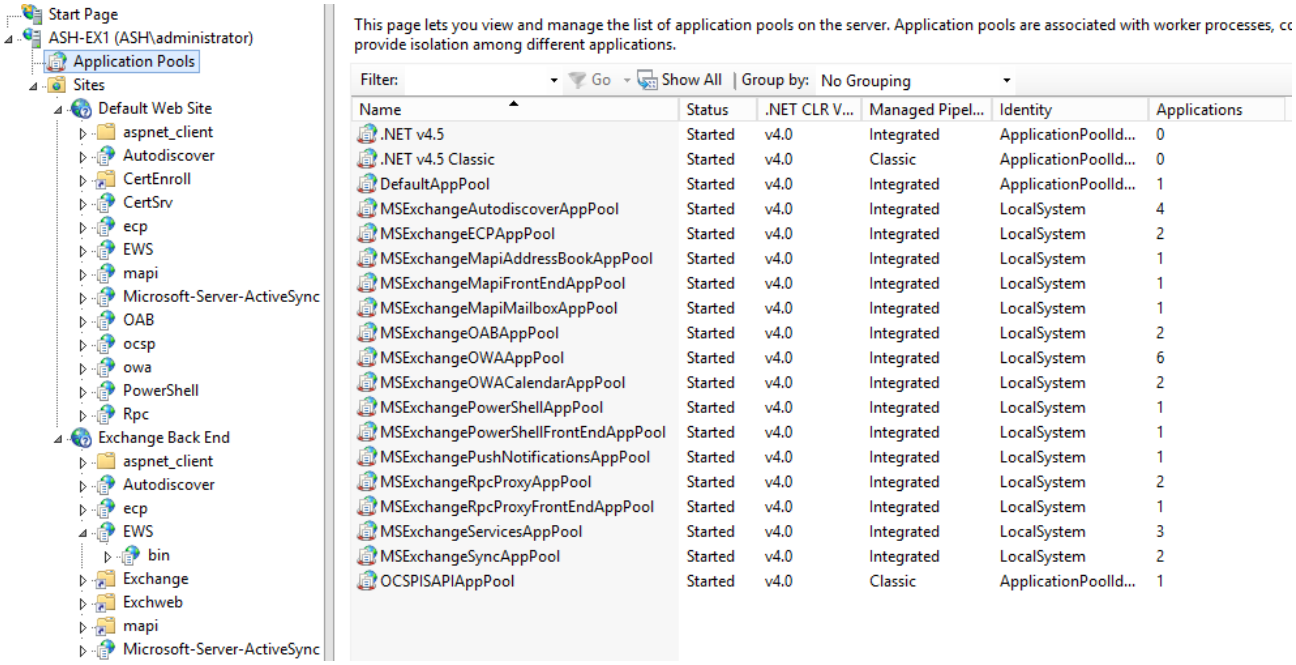


Figure 3-12: Viewing Application Pools in IIS Manager

How IIS can break

There's an idiom in troubleshooting that I hold as truth: you never truly understand something until you have to teach it to someone else. Second to that, I feel that configuring and then intentionally trying to break a piece of technology will drastically increase your knowledge of the product. The value of a lab environment for testing, experimentation, and comparison (to a broken environment) can't be overstated (if you're looking to get started with a lab server, I highly suggest Exchange MVP Jeff Guillet's series on building a home lab server). I mention this now because of all the Exchange support cases I've worked, the majority I've resolved as a result of using my lab environment for comparison were related to Client Access and IIS. Bindings, authentication settings, Application Pool configuration, IIS Modules, SSL settings, and certificates. The proper settings for each of these, with each version of Exchange, can be an unrealistic challenge for someone to commit to memory. Thus the need for a lab environment for comparison and testing. In that spirit, let's discuss the common ways these components can break and how to recover. However, Exchange-related changes should be made in Exchange Management Tools and not directly in IIS. Failure to do this could result in the change being reverted when the IIS Metabase queries the values defined in Active Directory.

Bindings and Firewalls

Excluding Certificates (which we previously discussed), the most common IIS-related issue I see is related to IIS Bindings. I'll commonly encounter customers trying to install a 3rd party application or a Microsoft application that is not explicitly supported on an Exchange 2013 Server (SharePoint, Remote Desktop Services, Lync, etc.) and in the process their bindings will become broken. Allow me to demonstrate. For example, assume that I'm logged into OWA on my multi-role 2013 server (Figure 3-13).

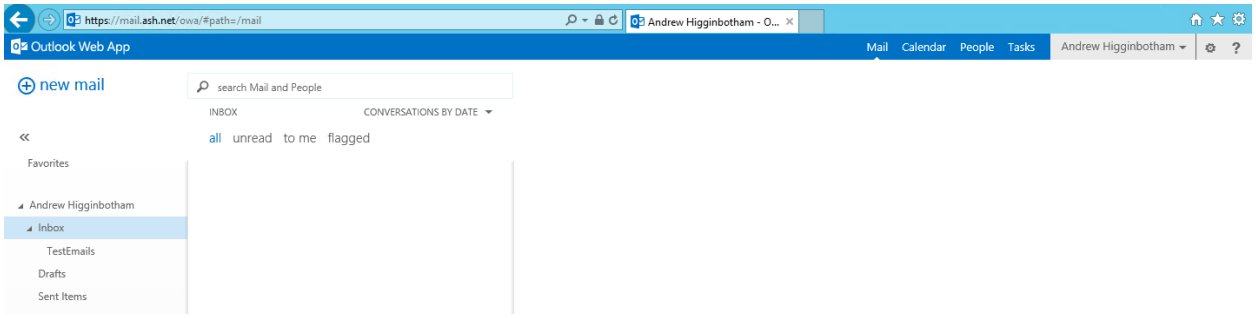


Figure 3-13: Exchange 2013 Outlook Web App

Now within IIS I right-click Exchange Back End>Edit Bindings and change the HTTPS binding from 444 to 445 (Figure 3-14).

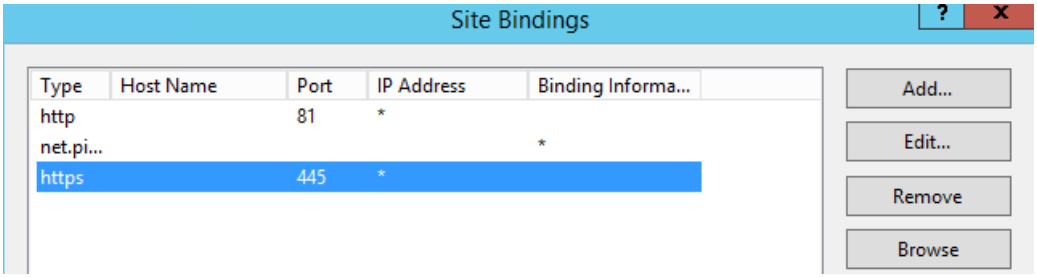


Figure 3-14: Changing the bindings of the Exchange Back End website in IIS Manager

If I now refresh my browser, I'll be greeted with a blank page (Figure 3-15).

Figure 3-15: Result of an incorrectly configured IIS binding

This is because, by design, the Default Web Site in Exchange 2013/2016 uses the traditional web server bindings for port 80 and 443, while the Exchange Back End website uses ports 81 and 444 for HTTP/HTTPS connectivity. When the Client Access Server role communicates with the Mailbox Server role for IIS–related functions, it proxies these connections via HTTPS using port 444 (port 81 for HTTP connections). So the expected flow for UserA logging into OWA on ServerA (single server environment for this first example) would be:

UserA using browser client
|
ServerA Default Web Site (over port 443)
|
ServerA Exchange Back End website (over port 444)
|
RPC/MAPI Communications to the local MSExchange Information Store Service

How would the traffic flow look if we were connecting to https://ServerA/owa with our browser but our mailbox (UserB) was on a database that was mounted on ServerB? Let's have a look:

UserB using browser client

|

ServerA Default Web Site (over port 443)

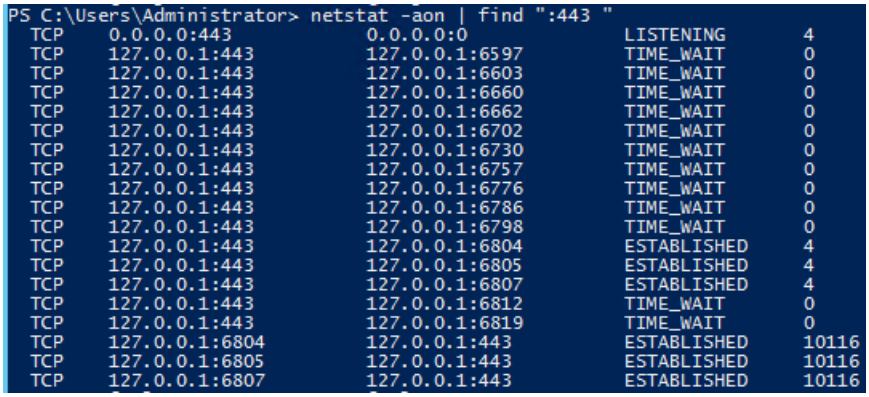
|

ServerB Exchange Back End website (over port 444)

|

RPC/MAPI Communications to the local MSExchange Information Store Service (on ServerB)

As you can see, in this scenario while the client connects to OWA using 443, CAS proxies that connection to the relevant Mailbox Server over 444 (over the network). If you want to see this in action, then you can use a tool like *NETSTAT* to view connections between your servers. In the example shown in Figure 3-16, I see a local connection to 443 and an associated Process ID (PID). I can use Task Manager to see that PID correlates to an instance of Internet Explorer (iexplore.exe), which I have open and connected locally to OWA.



```
PS C:\Users\Administrator> netstat -aon | find ":443 "
  TCP    0.0.0.0:443            0.0.0.0:0              LISTENING       4
  TCP    127.0.0.1:443          127.0.0.1:6597         TIME_WAIT       0
  TCP    127.0.0.1:443          127.0.0.1:6603         TIME_WAIT       0
  TCP    127.0.0.1:443          127.0.0.1:6660         TIME_WAIT       0
  TCP    127.0.0.1:443          127.0.0.1:6662         TIME_WAIT       0
  TCP    127.0.0.1:443          127.0.0.1:6702         TIME_WAIT       0
  TCP    127.0.0.1:443          127.0.0.1:6730         TIME_WAIT       0
  TCP    127.0.0.1:443          127.0.0.1:6757         TIME_WAIT       0
  TCP    127.0.0.1:443          127.0.0.1:6776         TIME_WAIT       0
  TCP    127.0.0.1:443          127.0.0.1:6786         TIME_WAIT       0
  TCP    127.0.0.1:443          127.0.0.1:6798         TIME_WAIT       0
  TCP    127.0.0.1:443          127.0.0.1:6804         ESTABLISHED     4
  TCP    127.0.0.1:443          127.0.0.1:6805         ESTABLISHED     4
  TCP    127.0.0.1:443          127.0.0.1:6807         ESTABLISHED     4
  TCP    127.0.0.1:443          127.0.0.1:6812         TIME_WAIT       0
  TCP    127.0.0.1:443          127.0.0.1:6819         TIME_WAIT       0
  TCP    127.0.0.1:6804         127.0.0.1:443          ESTABLISHED     10116
  TCP    127.0.0.1:6805         127.0.0.1:443          ESTABLISHED     10116
  TCP    127.0.0.1:6807         127.0.0.1:443          ESTABLISHED     10116
```

Figure 3-16: Local connections for port 443 viewed using the NETSTAT command

I now run a different command (Figure 3-17) from the same server but for port 444; this output is a bit busier. There's the connection to the local server for the OWA session that I'm logged into (the mailbox I'm logged in with is on a database that's mounted locally). However, you'll also find there's a connection to 10.180.62.191, which is one of my other Exchange servers. This is for another instance of OWA I have open for a mailbox that's currently mounted on that server. In that case the PID corresponds to an instance of w3wp.exe (World Wide Web Publishing Service). The other PIDs correspond to background processes like Microsoft.Exchange.ServiceHost.exe (MSEXchange Service Host Service), MSExchangeHMWorker.exe (MSExchange Health Manager Service), and MSExchangeMailboxAssistants.exe (MSExchange Mailbox Assistants Service). These are all background processes that are constantly running behind the curtains to keep Exchange up and running (managed availability, synthetic transactions, maintenance tasks, etc.).

It's possible that administrators accidentally change the bindings or delete them. Unfortunately, their attempts to repair the web sites typically result in their usage of the incorrect port numbers (like configuring 443 on the Exchange Back End site). Alternatively, customers (or their network security admins) may block port 444 traffic between servers and suddenly find their servers in a state of uselessness.

> **Note**: A common issue in Exchange 2013's timeline was the Exchange Back End site's HTTPS bindings having their certificate mapping removed. This could occur either after an Exchange update failure, a failure during a new certificate assignment, or the removal of the self-signed certificate which should be bound to this site. The resolution is to use IIS to re-assign a proper certificate.

```
PS C:\Users\Administrator> netstat -aon | find ":444 "
  TCP    0.0.0.0:444            0.0.0.0:0              LISTENING       4
  TCP    10.180.62.190:444      10.180.62.190:6194     ESTABLISHED     4
  TCP    10.180.62.190:444      10.180.62.190:6363     ESTABLISHED     4
  TCP    10.180.62.190:444      10.180.62.190:6808     ESTABLISHED     4
  TCP    10.180.62.190:444      10.180.62.190:6811     ESTABLISHED     4
  TCP    10.180.62.190:444      10.180.62.190:6862     ESTABLISHED     4
  TCP    10.180.62.190:444      10.180.62.192:6313     ESTABLISHED     4
  TCP    10.180.62.190:6194     10.180.62.190:444      ESTABLISHED     3584
  TCP    10.180.62.190:6363     10.180.62.190:444      ESTABLISHED     3584
  TCP    10.180.62.190:6801     10.180.62.191:444      ESTABLISHED     11156
  TCP    10.180.62.190:6808     10.180.62.190:444      ESTABLISHED     3600
  TCP    10.180.62.190:6811     10.180.62.190:444      ESTABLISHED     3600
  TCP    10.180.62.190:6840     10.180.62.191:444      ESTABLISHED     7080
  TCP    10.180.62.190:6844     10.180.62.191:444      ESTABLISHED     7080
  TCP    10.180.62.190:6862     10.180.62.190:444      ESTABLISHED     5224
  TCP    127.0.0.1:444          127.0.0.1:6651         ESTABLISHED     4
  TCP    127.0.0.1:444          127.0.0.1:6745         ESTABLISHED     4
  TCP    127.0.0.1:444          127.0.0.1:6876         ESTABLISHED     4
  TCP    127.0.0.1:444          127.0.0.1:6880         ESTABLISHED     4
  TCP    127.0.0.1:6651         127.0.0.1:444          ESTABLISHED     3600
  TCP    127.0.0.1:6717         127.0.0.1:444          TIME_WAIT       0
  TCP    127.0.0.1:6745         127.0.0.1:444          ESTABLISHED     3600
```

Figure 3-17: Local and remote connections for port 444 viewed using the NETSTAT command

Recreating Exchange Virtual directories

Ever since Exchange 2003, Microsoft has provided the ability to recreate Exchange virtual directories for troubleshooting purposes. Beginning with Exchange 2013 you have the option to reset/recreate the virtual directories either from within the Exchange Admin Center or Exchange Management Shell.

The recreating of an Exchange Virtual Directory (vDir) was often needed after corruption, misconfiguration, or unexplained failures were encountered. I've seen it resolve odd display issues in OWA as well as authentication failures. Recreating the various virtual directories was a useful troubleshooting step in the past, but I'll be honest when I say that it's usually done as a last ditch step whenever every other avenue of troubleshooting hasn't helped. In fact, if recreating the virtual directory doesn't resolve the issue I'm usually looking at a /RecoverServer install as the next step (this option is discussed in the Backup/Disaster Recovery chapter). However, recreating a virtual directory is useful when the components that depend on IIS (OWA/ECP/ActiveSync/EWS/OAB/PowerShell/AutoDiscover) aren't working as expected and you'd like to reset the relevant virtual directory to defaults.

> **Note**: Recreating the Virtual directories will remove any settings or customizations you have implemented, so I recommend running a "*Get-OWAVirtualDirectory* | *Format-List*" or similar command beforehand to record the existing settings. In fact, if you use the EAC to reset the virtual directories then you'll be prompted to save the configuration to a network path.

There are two ways to recreate a virtual directory: EAC (GUI) or EMS (Shell). Let's look at the EAC method first. Navigate to EAC>Servers>Virtual Directories, select the virtual directory you wish to reset and then click the Reset icon (Figure 3-18).

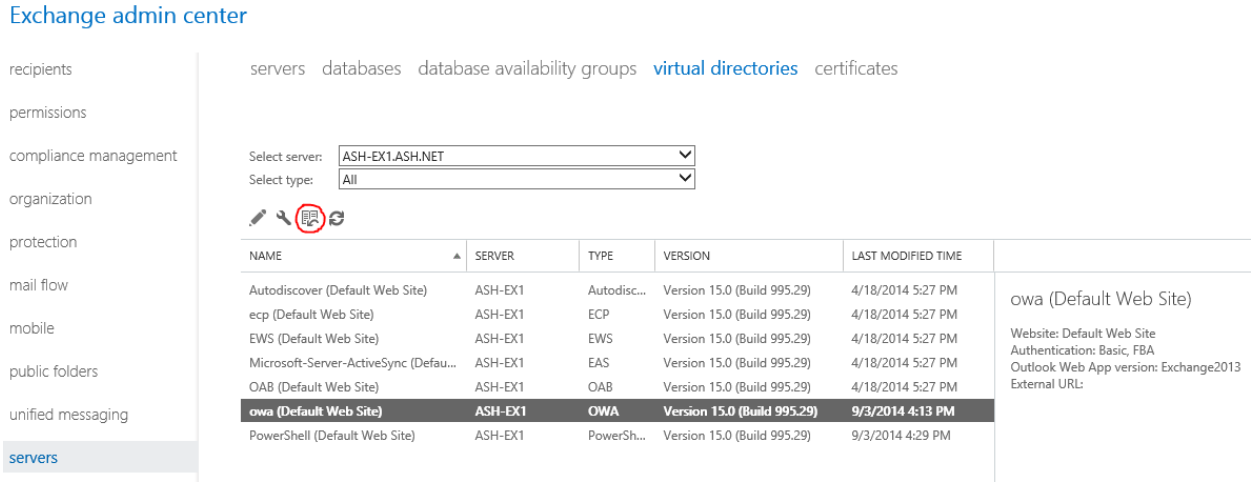


Figure 3-18: The Reset Virtual Directory icon in EAC

Figure 3-19 shows the prompt you'll receive to back up the current Virtual directory settings before resetting it.

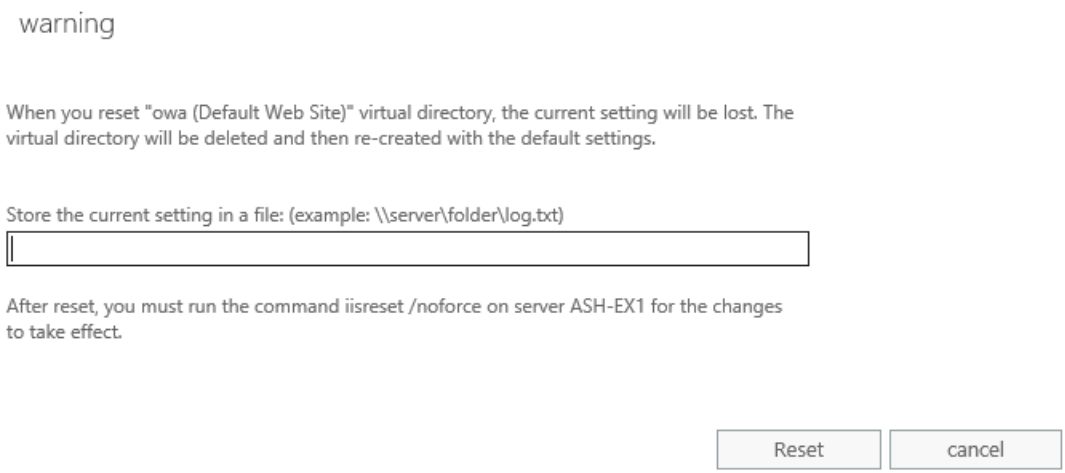


Figure 3-19: Prompt to backup virtual directory settings before recreating it

After clicking "Reset" the Virtual directory will be removed and then recreated. Afterwards you'll need to restart IIS (*iisreset /noforce*) and reconfigure any customized settings, such as authentication settings.

The steps to perform the same action through PowerShell are straightforward. Figure 3-20 shows the commands in action:

```
[PS] C:\Windows\system32>Get-OwaVirtualDirectory -Server ash-ex1

Name                                    Server                                  OwaVersion
----                                    ------                                  ----------
owa (Default Web Site)                  ASH-EX1                                 Exchange2013


[PS] C:\Windows\system32>Remove-OwaVirtualDirectory -Identity "ash-ex1\owa (Default Web Site)"

Confirm
Are you sure you want to perform this action?
Outlook Web App virtual directory "ash-ex1\owa (Default Web Site)" is being removed.
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [?] Help (default is "Y"): A
[PS] C:\Windows\system32>New-OwaVirtualDirectory -Server ash-ex1 -WebSiteName "Default Web Site"

Name                                    Server                                  OwaVersion
----                                    ------                                  ----------
owa (Default Web Site)                  ASH-EX1                                 Exchange2013


[PS] C:\Windows\system32>_
```

Figure 3-20: Exchange Management Shell commands used to reset a virtual directory

These commands work when we have an issue with the Default Web Site but I've actually encountered instances where it was required to recreate the OWA Virtual directory on the Exchange Back End site as well. To do this, run the commands shown in Figure 3-21:

```
PS C:\Users\Administrator> Add-PSSnapin Microsoft.Exchange.Management.PowerShell.SnapIn
PS C:\Users\Administrator> Remove-PowerShellVirtualDirectory -Identity "ash-ex1\PowerShell (Default Web Site)"

Confirm
Are you sure you want to perform this action?
Removing the Windows PowerShell virtual directory "PowerShell (Default Web Site)" on server "ASH-EX1".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is "Y"): A
PS C:\Users\Administrator> New-PowerShellVirtualDirectory -Name PowerShell -Server ash-ex1 -WebSiteName "Default Web Sit
e" -RequireSSL $False

Name                                                Server
----                                                ------
PowerShell (Default Web Site)                       ASH-EX1


PS C:\Users\Administrator> _
```

Figure 3-21: Removing and recreating the OWA virtual directory from the Exchange Back End web site

PowerShell

What do you do if you're having issues with the PowerShell virtual directory? You probably are unable to connect to the problem server to manage it via EMS or EAC (since both require a functional PowerShell virtual directory) so it will be required to load the local Exchange Management PowerShell snap-in using the commands shown in Figure 3-22:

```
[PS] C:\Windows\system32>Remove-OwaVirtualDirectory -Identity "ash-ex1\owa (Exchange Back End)"

Confirm
Are you sure you want to perform this action?
Outlook Web App virtual directory "ash-ex1\owa (Exchange Back End)" is being removed.
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [?] Help (default is "Y"): A
[PS] C:\Windows\system32>New-OwaVirtualDirectory -Server ash-ex1 -WebSiteName "Exchange Back End"

Name                                    Server                                  OwaVersion
----                                    ------                                  ----------
owa (Exchange Back End)                 ASH-EX1                                 Exchange2013


[PS] C:\Windows\system32>_
```

Figure 3-22: Recreating the PowerShell virtual directory using the local Exchange PowerShell snapin

Since we're on the topic of PowerShell, on occasion I've found myself having to verify all the proper IIS Modules are added for the PowerShell virtual directory (Figure 3-23).

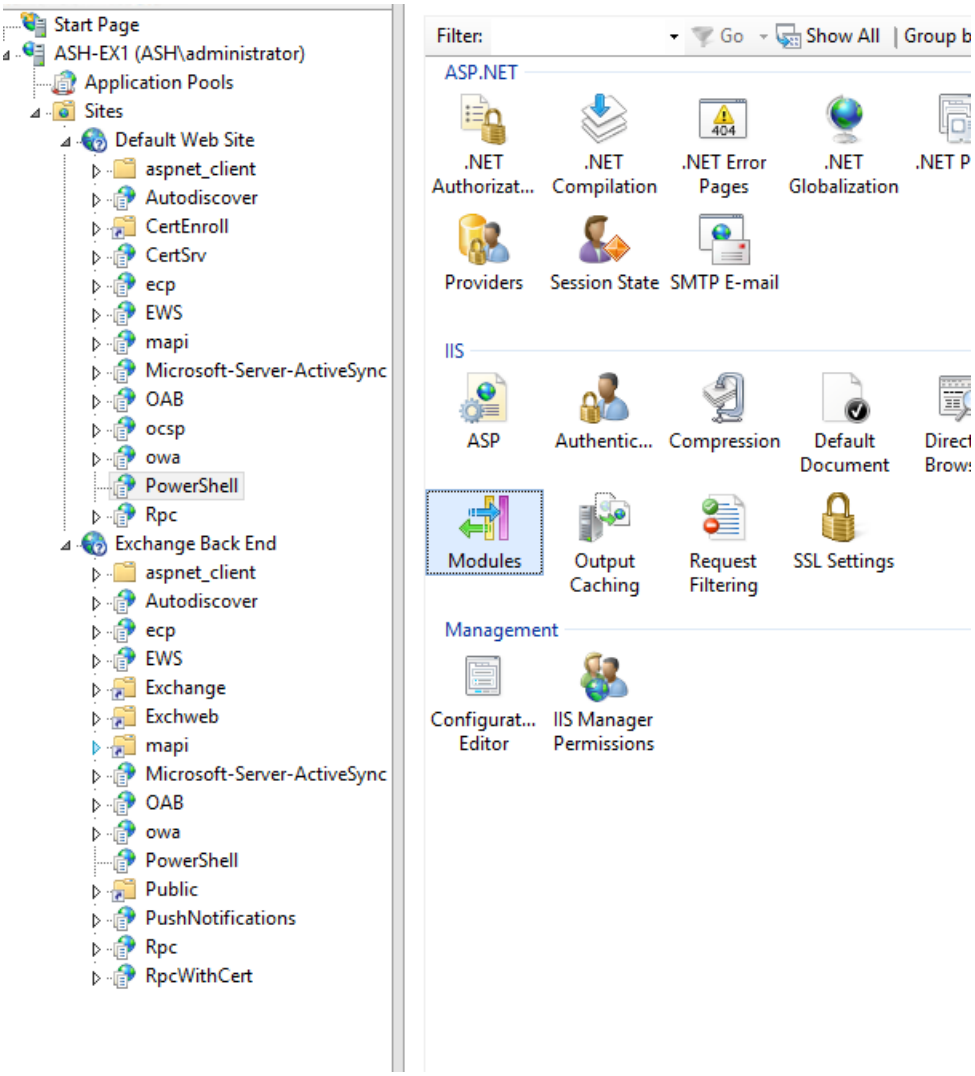


Figure 3-23: Managing the IIS Modules for the PowerShell virtual directory

I recommend comparing the loaded modules here to a known working server (or lab machine). On several occasions I've found the "kerbauth" module (Figure 3-24) to be missing and I've needed to re-add it. I've encountered missing modules with both Exchange 2010 and 2013 but with each version, the proper modules will be needed for proper functionality on any version of Exchange.

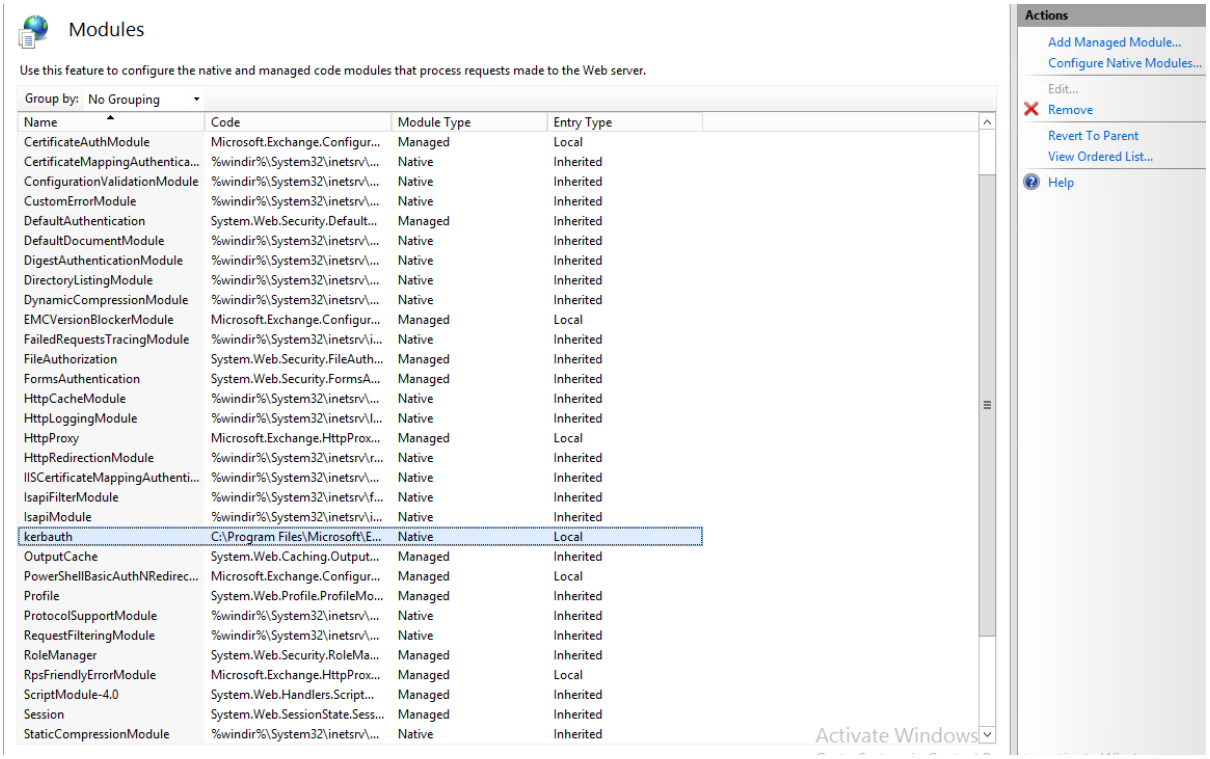


Figure 3-24: Listing of the various installed IIS modules

Note: Also make sure that any and all file directory paths have the proper permissions set on them. Again, it's helpful to have a known working server to use as a comparison. Also, be sure that all proper Anti-Virus Exclusions have been configured (an extremely common scenario).

Certificate Binding

Certificates are bound to both the Default Web Site as well as the Exchange Back End site in IIS. If you right-click on Default Web Site>Edit Bindings>Select HTTPS and click Edit you can see the current certificate bound to the site. When you run *Enable-ExchangeCertificate –Thumbprint <Thumbprint> -Services IIS*, this is what is configured within IIS. Figure 3-25 shows a certificate generated by my Internal Certificate Authority installed on the Default Web Site.

I often find the incorrect certificate listed here or that some certificates are missing. While the EAC and the *Get-ExchangeCertificate* shell command are extremely useful, many customers mistakenly think that the Exchange tools are the only way to Import/Export certificates. However, the Certificates MMC Snap-in is a very handy troubleshooting tool.

To open the Certificates MMC Snap-in:

- Open PowerShell
- Type "MMC" <Enter>
- File>Add/Remove Snap-in
- Select "Certificates" and click Add
- Select "Computer Account" and click Next
- Click Finish
- Click Ok
- Certificates for Exchange usage will be in the Personal>Certificates container

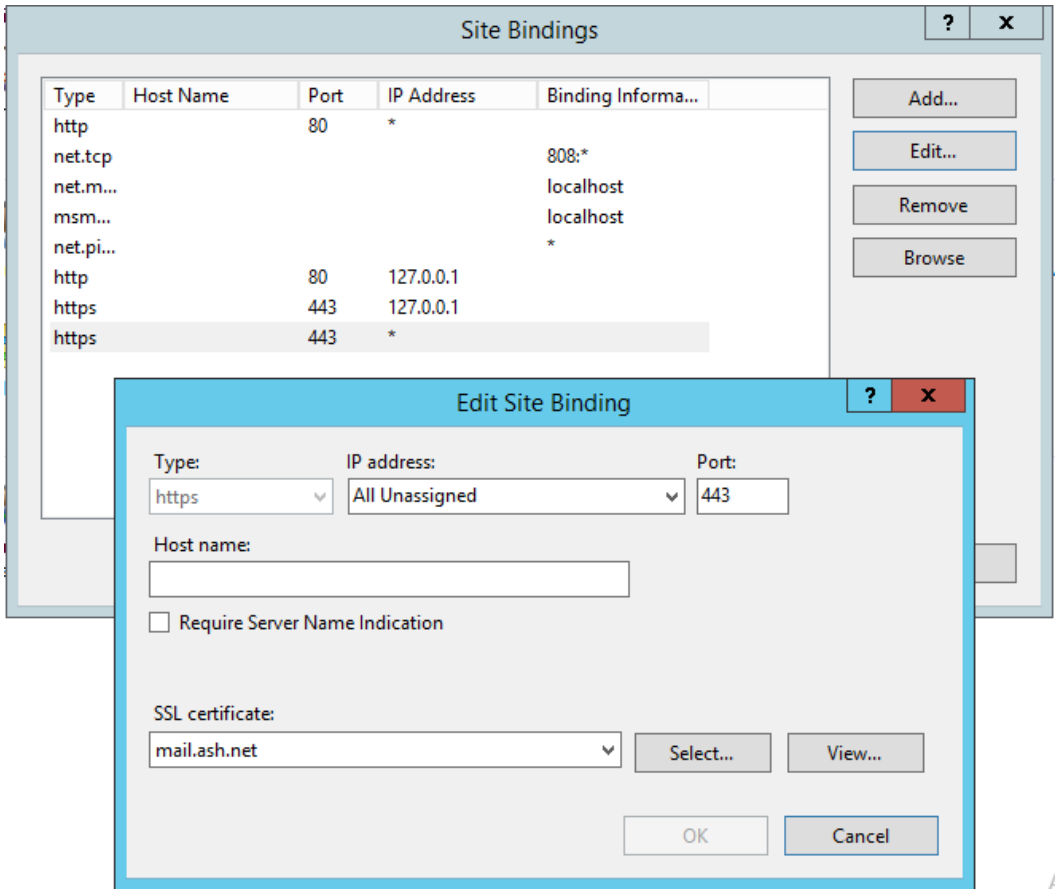


Figure 3-25: Certificate bindings in IIS on the Default Web Site

Figure 3-26 shows the Personal Certificates store of the Local Computer account. This is where manually installed certificates are likely to be stored. In short, when you run *Import-ExchangeCertificate* the certificate ends up here. So similarly you can use this console to Import/Export certificates as well.

Note: Your Personal store will likely look different than mine as my lab server is also a Domain Controller/Certificate Authority.

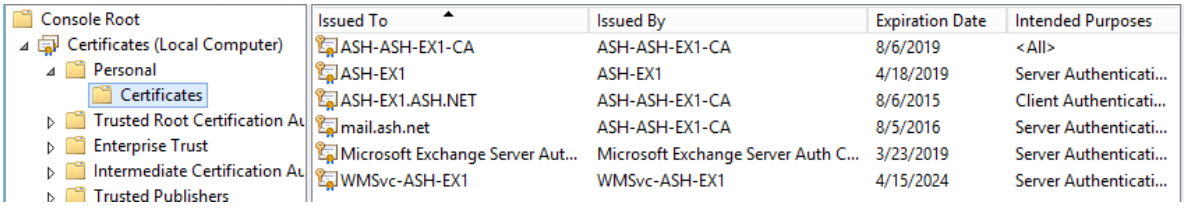


Figure 3-26: Personal Certificates store of an Exchange Server/Domain Controller/Certificate Authority

Certificate issues have historically revolved around generating the request, but the Certificate Request Wizards (Figure 3-27) used in Exchange 2010/2013/2016 have made this task much easier.

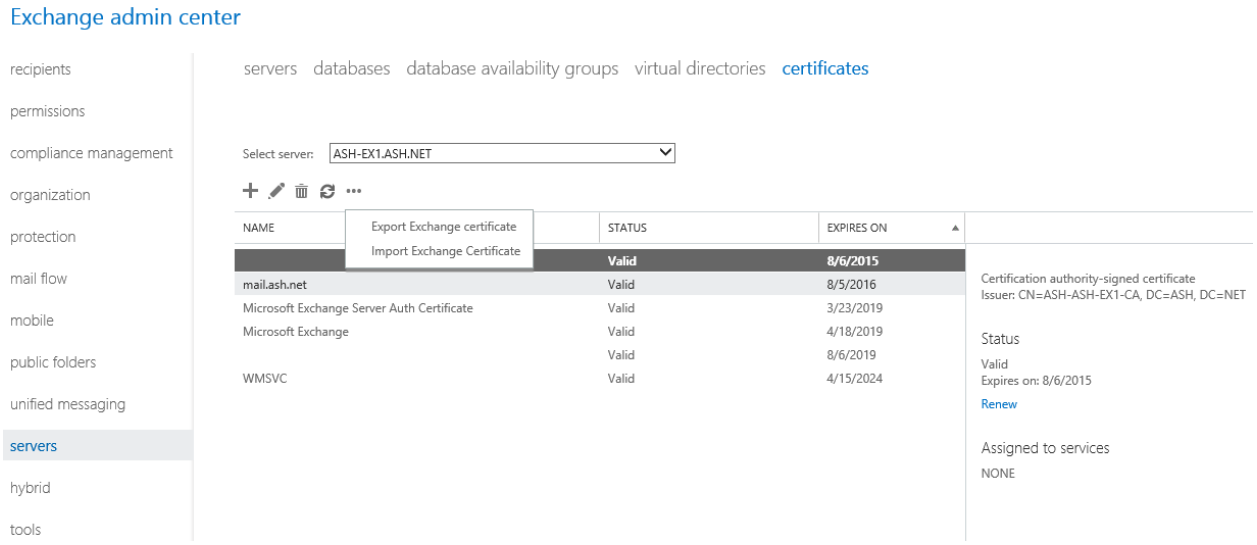


Figure 3-27: Certificate management in EAC and selecting the Certificate Import/Export toolbox options

As previously mentioned in the section covering Certificates, when you generate the certificate request on an Exchange server, you need to leave that request intact until you receive the new certificate file from your issuing Certificate Authority. If you fail to do this, your certificate will be missing the private key and be effectively useless for Exchange usage. I see this frequently when customers request a certificate multiple times or if they try to use a different server to import the certificate. It's possible a customer could connect to a load-balanced name for EAC access and request the certificate on one Exchange Server. Then upon completion of the request, they are taken to a different server. To avoid this, it may be required to connect directly to an Exchange server when requesting a certificate. Once a request has been generated, you'll see the pending request in the EAC Certificates console along with an option to "Complete" the request (Figure 3-28) to execute when you've received the certificate from your CA (this process generates the Private Key).

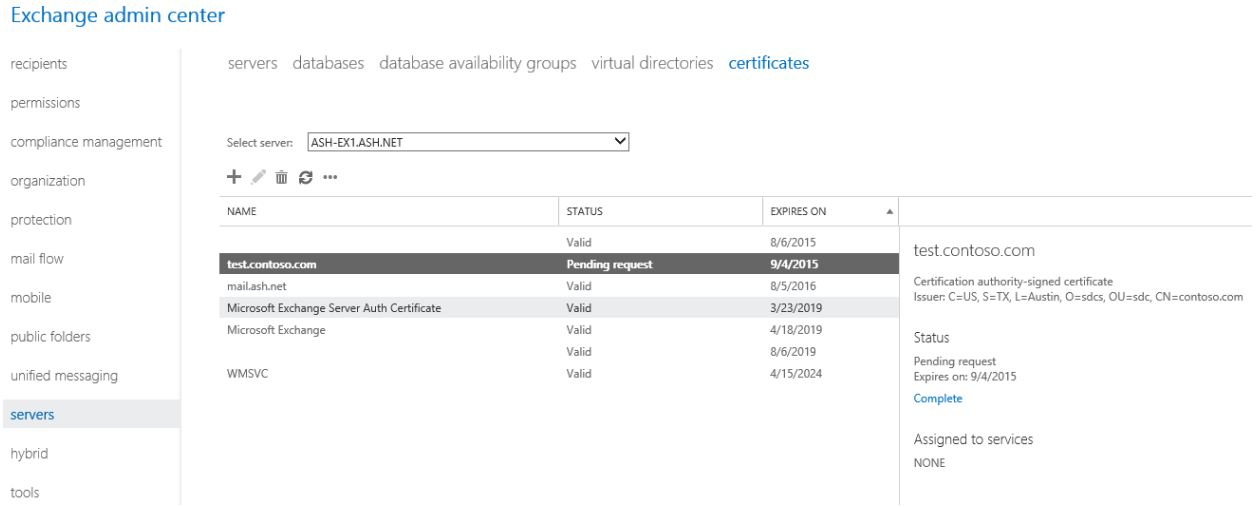


Figure 3-28: A pending certificate request in the EAC Certificates management interface

A common problem first encountered in Exchange 2013 was when the default self-signed certificate on the Exchange Back End website became unbound and caused HTTPS connections to fail. The issue was well documented by Microsoft (Reference1 Reference2). The issue often came to light after using the EAC certificate wizard to import or enable a new certificate. However, knowing how to navigate IIS to manually enable the self-signed certificate or even to initially identify the issue is a useful troubleshooting skill.

Additional Logging

Before leaving this section, I can't omit the plethora of logging that's now present in the install directory (typically *C:\Program Files\Microsoft\Exchange Server\V15\Logging)* of every Exchange Server. In fact, the logging is so vigorous that you'll often find it taking up quite a bit of your disk space. Luckily there are methods to truncate unneeded logs. These logs have come in handy when I've had to troubleshoot odd issues in the past related to CAS proxy behavior. I'd suggest taking time to look through these logs using notepad or even better, Log Parser Studio. It's a tool frequently used by Microsoft Support and great for when you have to parse through many log files trying to find a needle in a haystack. See this great series of blog posts from co-author Paul Cunningham.

Troubleshooting Exchange Load Balancing

Exchange 2013 introduced some architectural changes to remove the requirements for session affinity and layer 7 load balancing. We now have the option to load balance at layer 4, meaning no session affinity is required. While layer 4 load balancing is arguably a simpler configuration, it may not always be the best for your environment. Load balancing is a complicated topic which deserves its own book, but for the purposes of this discussion, let us take a moment to briefly define load balancing terminology.

Layer 4 versus Layer 7

The OSI Model is a conceptual model used for teaching and communication of functionality for networking technologies. Different layers of the OSI Model correspond to different functionality and are commonly identified numerically:

- Physical Layer (Layer 1)
- Data Link Layer (Layer 2)
- Network Layer (Layer 3)
- Transport Layer (Layer 4)
- Session Layer (Layer 5)
- Presentation Layer (Layer 6)
- Application Layer (Layer 7)

For example, a CAT6 network cable is said to operate at the Physical layer (Layer 1) of the OSI model, while a router operates at the Network layer (Layer 3), and Exchange operates at the Application layer (Layer 7). As Exchange generates an email message using SMTP (Layer 7) and creates a connection over port 25 (Layer 4) to another mail server's IP address (Layer 3) using a DSL connection (Layer 1), it goes down the OSI Model through each layer. On the receiving mail server, the connection goes up through each layer in a mirror fashion, starting at Layer 1 and working its way up to Exchange at the Application Layer.

When discussing load balancing, specifically layer 4 load balancing, connections are distributed using only knowledge of the connection at Layers 1-4. This means that the most sophisticated pieces of information we could gather at this layer are TCP/UDP port number and IP address. The load balancer has no knowledge of whether the port 443 traffic is being used for OWA or Outlook Anywhere or Terminal Services Gateway. The load balancer is also unable to inspect the traffic as that information exists at layer 7. As such, it is impossible to determine whether an application/service is in a healthy state. For example, all Exchange services could be stopped on a server, but since the server is still reachable over TCP/IP (PING, etc.) then traffic would still be sent to it.

Alternatively, because layer 7 load balancing inspects traffic at the protocol level, it is able to make decisions based on the actual workload (OWA vs EWS vs Outlook Anywhere etc.). It is also said to have "service

awareness", meaning if a service is unresponsive then the load balancer will not send traffic to that location. This service awareness is typically achieved by using health checks at layer 7, such as loading an HTML page or attempting an SMTP synthetic session.

SSL Offloading

Another benefit of layer 7 load balancing is the ability to perform SSL Offloading, which terminates the SSL session at the load balancer and allows the connection from the load balancer to the server to be unencrypted and removes the SSL processing overhead from the Exchange servers. Although this may not provide significant performance gains given the processing power of modern servers, it may still be a desirable configuration in some situations. I once received a tip from a Microsoft Premier Field Engineer that an alternative to SSL offloading would be to have a 2048-bit (or higher) certificate on the load balancer and a 1024-bit certificate on the Exchange servers themselves. While this would technically be SSL Bridging instead of offloading, you're still able to reduce the load on the Exchange servers by using a less resource-intensive certificate. This also has the advantage of maintaining SSL encryption from client to server instead of passing data unencrypted from the load balancer to the servers.

Session Affinity

Simply put, Session Affinity (aka Sticky Sessions or Persistence) ensures that new client connections for existing client-to-server sessions are always directed to the same server. For example, if a client browser session is established for an e-commerce website, where a user has logged into the server, it's desirable for that server to handle that session for its duration. If that e-commerce website opens a new browser tab requesting the user enter their payment information (creating a new connection for the same session), the user being repeatedly prompted for login is undesirable. This prompt could occur if this new connection were to be directed to a different server by the load balancer because the new server would not have the user's login information or token in its memory. To avoid this, Affinity/Stickiness ensures that for the lifetime of that session, each new connection is directed to the same server. While new sessions can be directed to any server in the load balanced pool, which would then receive any new connections for that session. There are various means to achieve affinity, the most common are:

- Web-Cookie
- HTTP Header
- SSL Session
- Source IP

> **Note:** This information is used by the load balancer to direct incoming traffic to the desired server in the load balanced pool.

Session affinity was required due to the Client Access architecture of Exchange 2010. All rendering and authentication for client sessions (such as OWA) occurred within the Client Access Server role, which then used RPCs to connect to the mailbox database on the user's mailbox server (wherever it may be at the time). This process depended on session affinity for the Client Access Server which initially processed the client request, otherwise the user could be repeatedly prompted for credentials or have connection failures.

Exchange 2013 introduced architectural changes which no longer required this affinity. All rendering occurs on the Exchange 2013 mailbox servers that currently mounts the target mailbox, meaning no matter which Client Access Server the load balancer directs the connection to, it will always be directed to the same mailbox server. Another change was the introduction of a shared hash between Client Access Servers which allows an authenticated session to be shared between each CAS in the load balanced pool. When a user is

authenticated, an authentication hash is created using the installed SSL certificate. The client then uses this hash in future requests to the server, at which point any CAS can now service this request without requiring the user to re-authenticate. Since this hash is generated using the installed SSL certificate on CAS, the only requirement is that the same certificate be enabled for IIS on all CAS in the load balanced pool.

The current recommendation for load balancing Exchange 2016 is to implement layer 7 load balancing without session affinity. This allows layer 7 service awareness for the load balancer, as well as allowing the simplicity of no session affinity. However, if Office Web Apps Server now called Office Online Server is deployed to enable the viewing/editing of Office files in Outlook Web App, cookie-based session affinity is required. Since OOS uses OAuth for authentication, the typical signs of misconfigured affinity (such as repeated authentication prompts) may not be displayed. Instead, you should look for signs such as any slowness while using the OOS features, or changes/edits not being saved while working with Office documents in OWA. These might be the result of misconfigured affinity for the OOS load balanced namespace. When troubleshooting OOS, it's important to know that both the client and the Exchange Server communicates to OOS using the load balanced URL. This means that if you wish to bypass the load balancer, you will require a HOSTS file on both the client machine as well as the Exchange Server (more information in the Load Balancing Troubleshooting Tips module).

Load Balancing Mechanism

Load balancing or Scheduling methods define the way in which new connections are distributed amongst load balanced servers. Common load balancing methods are as follows:

- Round Robin - This method tells the load balancer to direct requests to real servers in a round robin order.
- Weighted Round Robin - This method allows each server to be assigned a weight to adjust the round robin order. E.g. "Server 1" can get 2 times the request that "Server 2" gets.
- Least connection - This method tells the load balancer to look at the connections going to each server and send the next connection to the server with the least amount of connections.
- Weighted least connection - This method allows each server to be assigned a weight to adjust the least connection order. E.g. "Server 1" can get 2 times the connections that "Server 2" gets.
- Agent-Based Adaptive Balancing - This method is resource based load balancing where an agent gets installed on the server and monitors the server's resources (e.g. RAM, CPU...) and then reports back a percentage to the load balancer which is used for load balancing.
- Fixed Weighting - This method is used for redundancy rather than load balancing. All connections will go to the server with the highest weight in the event this server fails then the server with the next highest weight takes over.
- Weighted Response Time - This method looks at the response times of the real servers (based on the response time of the server health check) and which every real server is responding fastest gets the next request.
- Source IP Hash - This method looks at the source IP address that sent the request to the load balancer and will create a HASH value for it and if the HASH value is different it gets sent to a different real server.

Note: List taken from KEMP Technologies Knowledge Base

The Exchange Product Team recommends the Least Connection method with the stipulation that you should use a Slow Start or similar option. This "Slow Start" option (different vendors may use different terminology) will prevent a flood of new connections to a server after it's added to the pool. The Exchange Product Team

discussed this in their Performance Troubleshooting session at Ignite. If the Slow Start option is not used, a newly added server might be overloaded with new connections, causing Managed Availability to restart the server, with the possibility that the server would be immediately overloaded by the load balancer once again after rejoining the pool. This cascading failure scenario can be easily avoided by implementing a Slow Start or similar configuration.

> **Note:** I recommend researching and testing additional features load balancers may provide. In the blog post below, a network congestion prevention feature called Nagle's Algorithm adversely affected the performance of Outlook clients.
> Poor Outlook performance and Nagle's algorithm

Windows Network Load Balancing

Very briefly (because that's all I feel it deserves) we'll discuss Windows Network Load Balancing (WNLB). WNLB is a free solution from Microsoft provided as part of the Windows feature set. It has remained largely unchanged or improved upon for almost 15 years. While WNLB can work fine in a lab environment, I strongly recommend against it in production for the following reasons:

- Can only load balance at Layer 4
- Can only perform Client IP-based affinity
- Has no service awareness
- Port floods by nature and causes excessive network traffic
- Works fine in a vacuum but has many incompatibilities with Anti-Virus filters, hypervisors, and network equipment

Lastly and most importantly, WNLB is incompatible with Windows Failover Clustering, which is required on a Database Availability Group (DAG) member server. In other words, it is impossible to have a DAG node also be a member of a WNLB cluster. In Exchange 2010/2013, this restriction required dedicated Client Access Servers if WNLB was employed. However, because Exchange 2016 now has one consolidated server role, there is no longer an option to install only a Client Access Server. In light of this architectural change, customers must either implement a third-party load balancing solution or use Exchange 2016 servers with no active mailboxes on them (and not members of a DAG) in a WNLB load balanced pool. Since the latter option is a waste of an Exchange Server license, added complexity, and offers no added technical benefit, I strongly recommend against it.

> **Note:** Useful reference for Troubleshooting WNLB

Load Balancing Troubleshooting Tips

As a general rule, the simplest means of diagnosing a suspected load balancing issue is to bypass the load balancer itself. This is usually accomplished by modifying the HOSTS file on the client system which is accessing the Exchange load balanced resource.

For example, if the load balanced name is https://Mail.Contoso.com/owa which resolves to 10.0.0.50 (a VIP on the load balancer), and we would like to bypass the load balancer without configuration changes to Exchange or the load balancer itself, we perform the following steps:

- Navigate to the HOSTS file (C:\Windows\System32\drivers\etc) on the client machine being used to access OWA/Outlook/etc.

- Add an entry for *mail.contoso.com* which resolves to an individual Exchange Server's IP address (such as 10.0.0.10)
- Via an elevated command prompt on the client machine, run *ipconfig /flushdns* to force the client to remove any cached DNS entry for the mail.contoso.com record.
- Connect to the Exchange resource from the client using the usual means (Outlook/OWA/etc.)

Performing these steps will bypass any affinity or networking misconfigurations which may be present on the load balancer. It is possible that the symptoms noted while connecting through the load balancer will be sporadic as in the case of connectivity failures which are resolved by a refresh of the client, an authentication prompt that only occurs once or twice a day, or performance issues at unpredictable times. These symptoms can be caused by a singular load balanced server which has either been misconfigured or has encountered a failure of some kind. In this situation, you can do the following:

- Look for connectivity errors in the Exchange logs on each Exchange server (C:\Program Files\Microsoft\Exchange Server\V15\Logging)
- Use connectivity or health checking logs on the load balancer to identify a server being unresponsive
- Use a HOSTS file to point a client machine to one server and test the behavior. Repeat this process until you encounter the Exchange server exhibiting the issue

> **Note:** Ensure that once your testing is complete, you revert any HOSTS file changes you have implemented. Otherwise, the client system will fail to connect if a single server failure occurs.

Additional References**:**

- Issues With Load Balancing SMTP Traffic
- Troubleshooting long running MAPI connections to Exchange Server 2010 through Network Load Balancers

Validating Exchange Endpoints

A simple method of verifying Exchange services are available and properly responding is to use a web browser to connect to the server. I recommend this method to verify not only that the web site is listening and prompting for authentication, but also to ensure it's providing the proper response. If a particular Exchange client is unable connect, it's often useful to use a browser to validate the endpoint is reachable. In a load balanced environment, you can compare the results when accessing the load balanced name to accessing a server's FQDN or IP address. The screen shots shown below are the expected responses when using a browser to connect to each web service in Exchange:

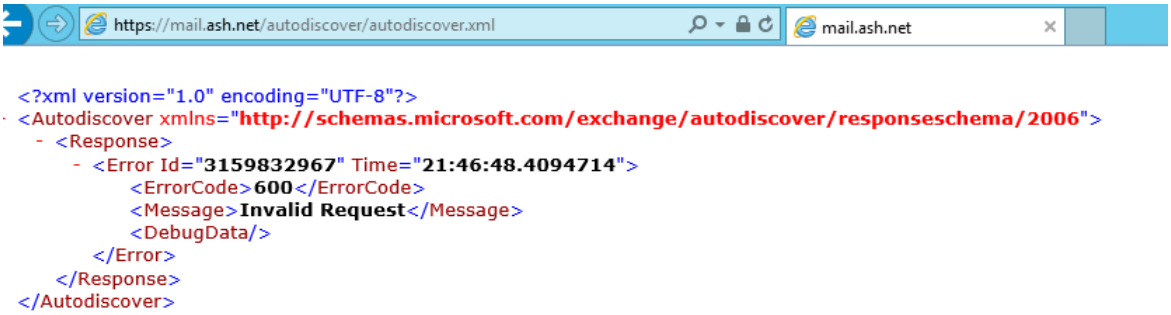


Figure 3-29: Expected response when authenticating against the AutoDiscover URL

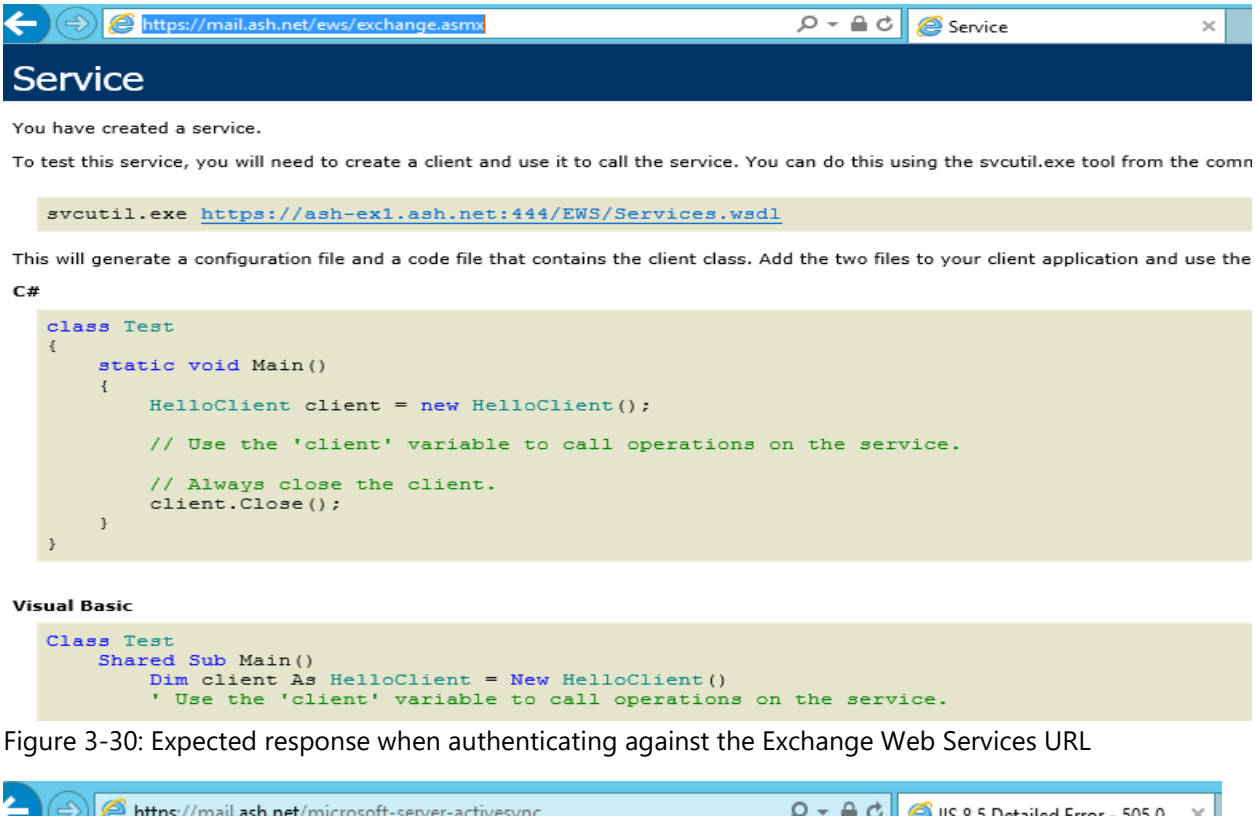


Figure 3-30: Expected response when authenticating against the Exchange Web Services URL

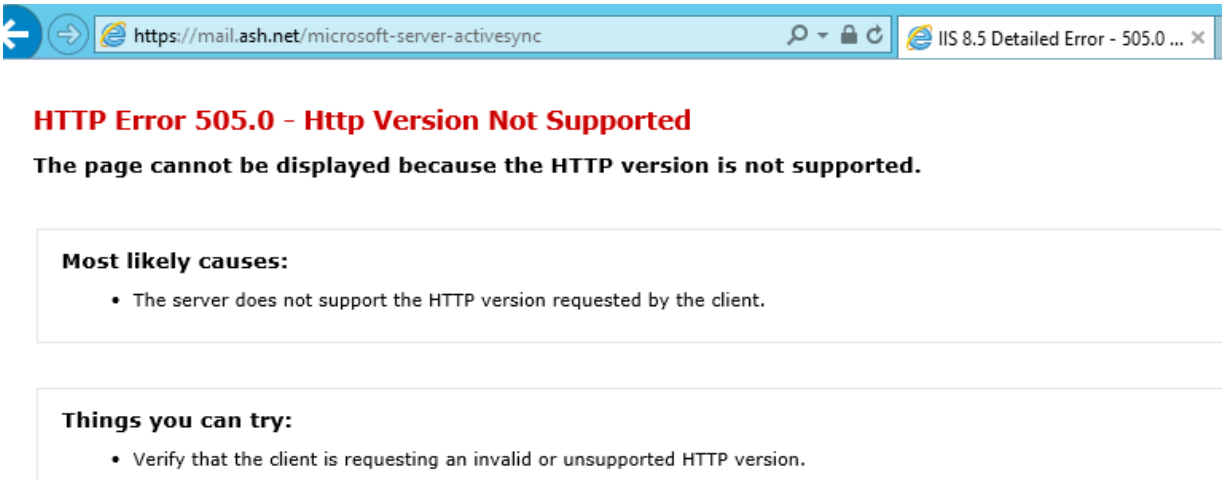


Figure 3-31: Expected response when authenticating against the ActiveSync URL

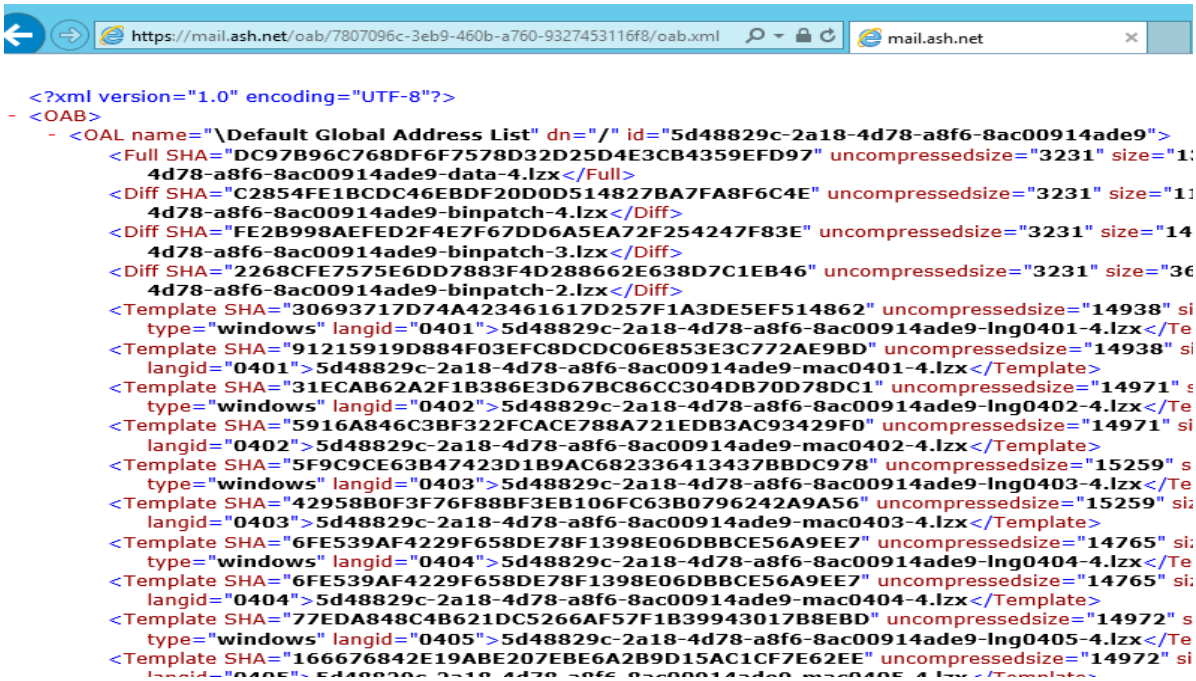


Figure 3-32: Expected response when authenticating against the Offline Address Book URL

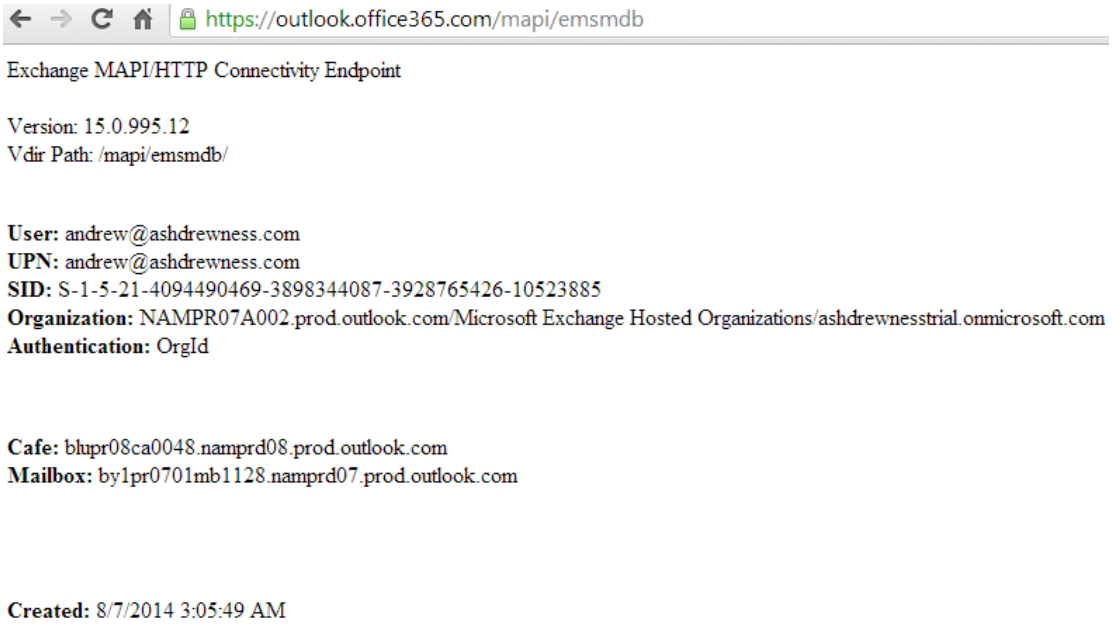


Figure 3-33: Expected response when authenticating against the MAPI URL (MAPI over HTTP)

Figure 3-34: Expected response (blank page) when authenticating against the RPC URL (Outlook Anywhere/RPC over HTTP)

The current URL values can be found on the various Virtual Directories in Exchange itself, or more accurately, based on the configuration AutoDiscover passes to the client. The most common means of obtaining this information and/or testing AutoConfiguration are:

- Microsoft Remote Connectivity Analyzer (TestExchangeConnectivity.com)
- Test E-mail AutoConfiguraton (tool within Outlook client)
- Test-OutlookWebServices
- Support and Recovery Assistant

The first two tools are most useful. Let's see how they work.

Microsoft Remote Connectivity Analyzer

The Microsoft Remote Connectivity Analyzer (ExRCA) is the most useful Exchange troubleshooting tool Microsoft has ever released. Although ExRCA had humble beginnings (running on a web server under the desk of a Microsoft Support Engineer), it has grown to include virtually every Exchange (as well as Lync and Office 365) connectivity test you would need to troubleshoot or validate your solution. The tool can perform the following tests:

- Exchange ActiveSync (with AutoDiscover)
- Exchange Web Services (Availability, Automatic Replies, and Developer Service Account Access)
- Outlook Anywhere (with AutoDiscover)
- Inbound and Outbound SMTP Email
- POP and IMAP
- Lync AutoDiscover and Availability
- Office 365 DNS validation
- Office 365 Free/Busy
- Message Header Analysis
- Office 365 Client Performance/Network Analyzer
- Office 365 Support and Recovery Assistant (SaRA Tool)

Note: For POP3 and IMAP troubleshooting, protocol logging should be enabled by setting the *ProtocolLoggingEnabled* parameter of the Set-PopSettings and Set-ImapSettings cmdlets.

In addition, protocol logging for POP3 and IMAP can be found in the below locations:
%ExchangeInstallPath%Logging\IMAP4\
%ExchangeInstallPath%Logging\POP3\

Figure 3-35 shows the output of the Outlook Anywhere test, with an overview of each phase displayed. The test queries and validates AutoDiscover, as well as the capabilities for both RPC over HTTP and the newer MAPI over HTTP. The individual test phases can be expanded to detail each step for further analysis, or the user can trust the green ticky ticky as the symbol of a successful test. Figure 3-36 shows an expanded availability synthetic transaction performed by the Exchange Web Services test within ExRCA.

Note: For configuring, logging, and troubleshooting MAPI over HTTP specifically, I recommend the below posts:
Configure MAPI over HTTP
Outlook Connectivity with MAPI over HTTP

In addition, logging for MAPI over HTTP can be found in the below locations:
%ExchangeInstallPath%Logging\MAPI Address Book Service\
%ExchangeInstallPath%Logging\MAPI Client Access\
%ExchangeInstallPath%Logging\HttpProxy\Mapi\

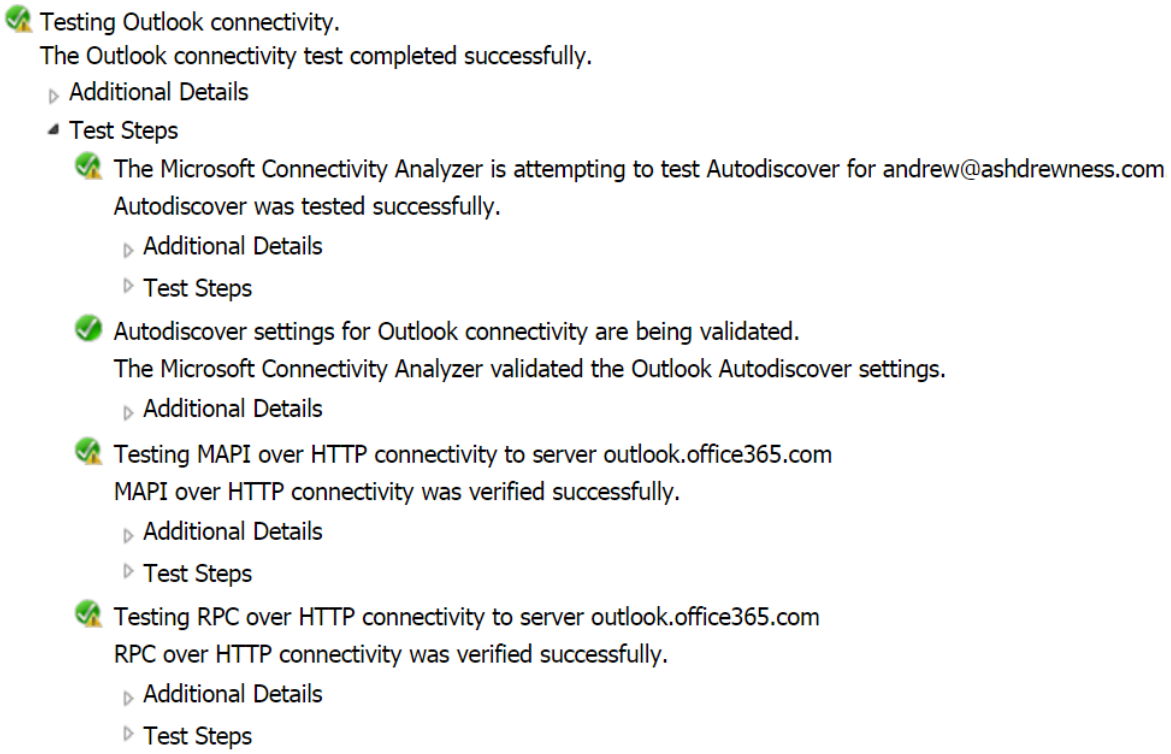


Figure 3-35: Overview of results from a successful Outlook Anywhere test from the Exchange Remote Connectivity Analyzer

The tool allows the results to be exported either to an XML or HTML file for later viewing or sharing with a support representative. It's important to understand though you may receive a failure (red x instead of the

green tick mark) at one phase of testing, it does not mean the entire test failed. For example, there are several phases of AutoDiscover testing that Outlook clients can perform:

- https://domain.com/AutoDiscover/AutoDiscover.xml
- https://AutoDiscover.domain.com/AutoDiscover/AutoDiscover.xml
- http://AutoDiscover.domain.com/AutoDiscover/AutoDiscover.xml
- Locally configured XML file
- SRV AutoDiscover DNS record

Most customers will use but one method for publishing AutoDiscover so it's expected to see all AutoDiscover tests fail but the endpoint published for that Exchange Organization. Figure 3-37 displays the output of the AutoDiscover test against a mailbox in my Office 365 tenant. Since Office 365 uses the HTTP redirect method for AutoDiscover, each previous step failed until the HTTP method was attempted.

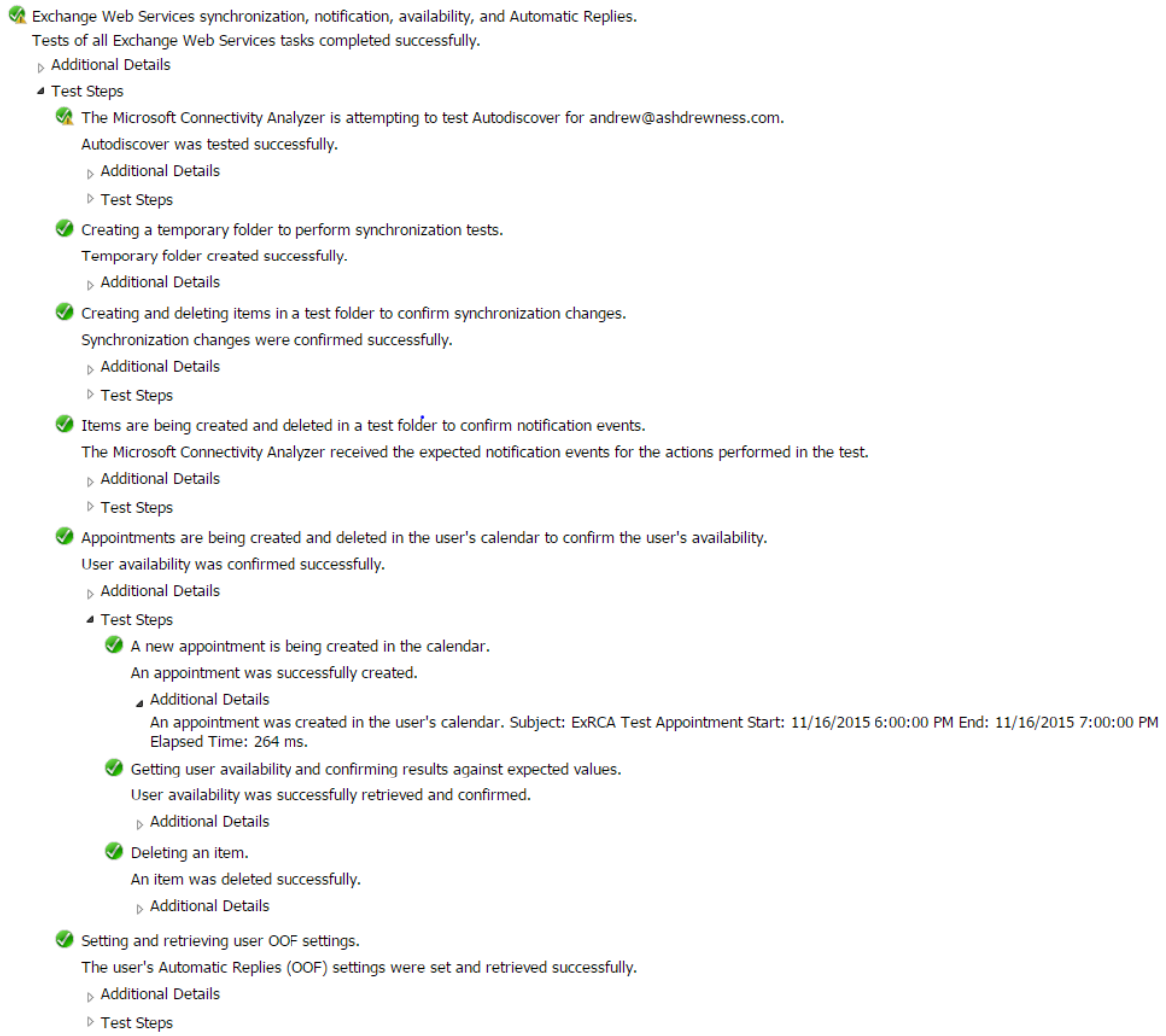


Figure 3-36: Expanded results of Availability Service (EWS) test where an appointment was placed onto user's calendar

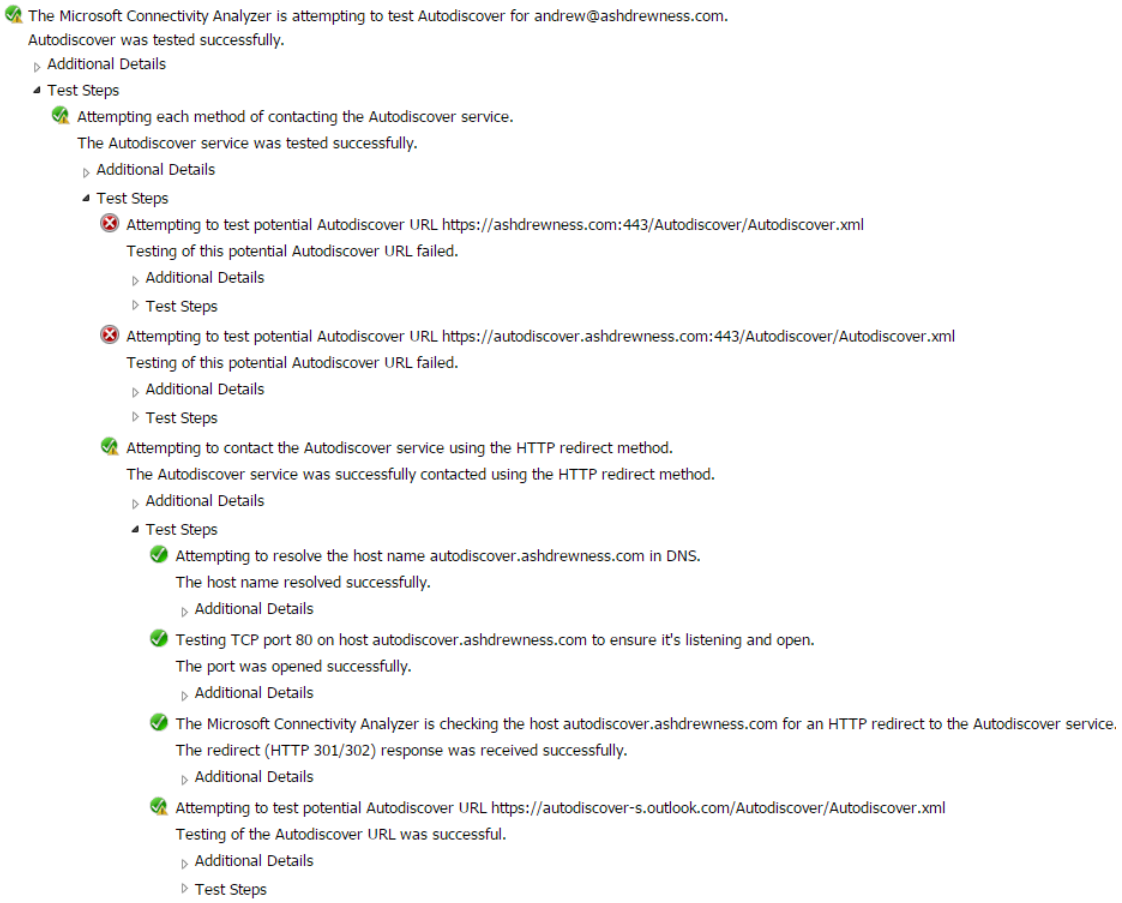


Figure 3-37: ExRCA testing the various AutoDiscover endpoints commonly used by Exchange clients

While the test had several failures within some of its individual phases, the overall test did pass, which is what you should be concerned with when using the ExRCA tool. Personally, I've found this tool most useful in the following scenarios:

- Outlook clients unable to connect
- Authentication or certificate pop-ups in Outlook
- ActiveSync clients are unable to connect
- ActiveSync clients are unable to use AutoConfiguration
- Free/Busy requests are failing either internally or to another organization
- Inbound mail flow failures
- Outbound mail flow failures

Test E-mail AutoConfiguration

Included in every version of the Outlook desktop client since Outlook 2007, the Test E-mail AutoConfiguration tool is an excellent means of viewing the XML configuration file the AutoDiscover service delivers to an Outlook client in a human readable format. Within this XML data, you can find the following information:

- Internal/External Outlook Web App (OWA) URL
- Internal/External Exchange Admin Center (ECP) URL
- Internal/External Exchange Web Services (EWS) URL
- Internal/External Offline Address Book (OAB) URL
- Internal/External Unified Messaging (UM) URL
- Internal/External Outlook Anywhere (RPC/HTTP or MAPI/HTTP) URL
- Authentication Mechanism

- Server FQDN
- Mailbox LegacyDN
- Public Folder SMTP Address/LegacyDN

This information is extremely useful when you must verify that the values you've configured in Exchange are being provided to the Outlook client. For example, it's possible that Active Directory replication latency and IIS Application Pool stale cache result in clients not receiving the proper values. In addition, Outlook can potentially take several hours to refresh its AutoDiscover cache. Knowing what information is being served up to clients is useful in troubleshooting scenarios. To access this tool, perform the following steps:

- With Outlook open and a profile configured, *CTRL+Right Click* the Outlook icon in the taskbar
- Select Test E-mail AutoConfiguration (Figure 3-38)

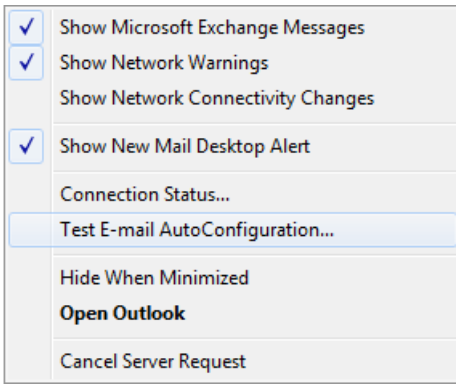


Figure 3-38: Opening the Test E-mail AutoConfiguration tool

Input the primary SMTP address and password of the account you would like to test and clear the two Guessmart checkboxes (Figure 3-39)

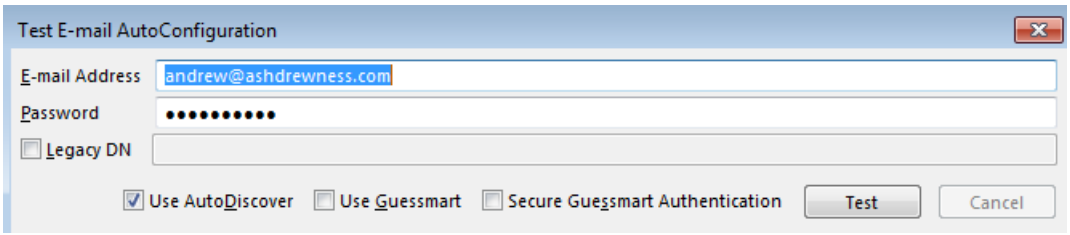


Figure 3-39: Testing a mailbox; must use primary SMTP address

Click *Test* and wait for the tests to complete. You can view the progress on the *Log* tab, which will display the various AutoDiscover endpoints being attempted (Figure 3-40).

```
Autodiscover to https://ashdrewness.com/autodiscover/autodiscover.xml starting
GetLastError=12007; httpStatus=0.
Autodiscover to https://ashdrewness.com/autodiscover/autodiscover.xml Failed (0x800C8203)
Autodiscover to https://autodiscover.ashdrewness.com/autodiscover/autodiscover.xml starting
GetLastError=12029; httpStatus=0.
Autodiscover to https://autodiscover.ashdrewness.com/autodiscover/autodiscover.xml Failed (0x800C8203)
Local autodiscover for ashdrewness.com starting
Local autodiscover for ashdrewness.com Failed (0x8004010F)
Redirect check to http://autodiscover.ashdrewness.com/autodiscover/autodiscover.xml starting
Autodiscover URL redirection to https://autodiscover-s.outlook.com/autodiscover/autodiscover.xml
Autodiscover to https://autodiscover-s.outlook.com/autodiscover/autodiscover.xml starting
GetLastError=0; httpStatus=401.
GetLastError=0; httpStatus=401.
GetLastError=0; httpStatus=200.
Autodiscover to https://autodiscover-s.outlook.com/autodiscover/autodiscover.xml Succeeded (0x00000000)
Redirect check to http://autodiscover.ashdrewness.com/autodiscover/autodiscover.xml Succeeded (0x00000000)
```

Figure 3-40: Each AutoDiscover endpoint method being tested

You can choose to look at the configuration information AutoDiscover proved on the *XML* tab in .XML format, or use the more user friendly *Results* tab (Figure 3-41)

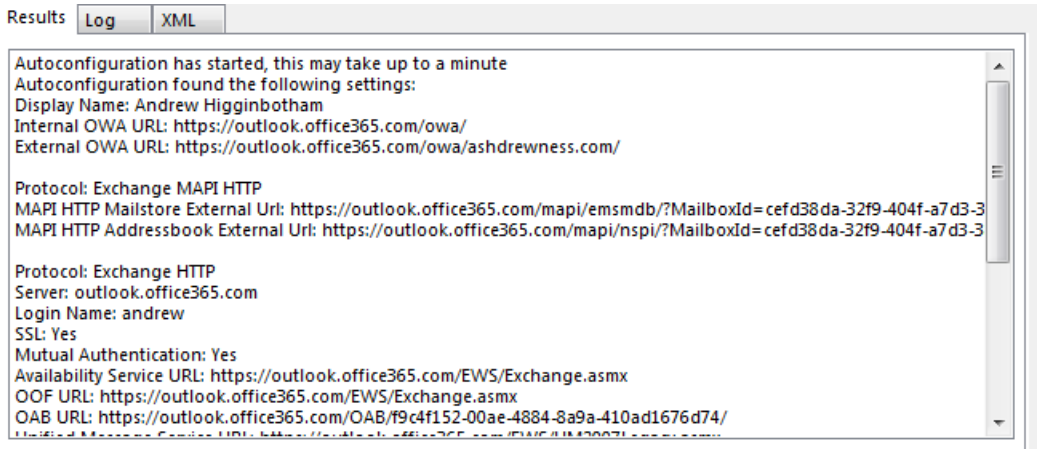


Figure 3-41: Output from a successful AutoDiscover query

Understanding the components and infrastructure which allows clients to connector to their mailbox empowers us to diagnose client access failures. As you would expect, there's little use for an Exchange Server that no client can access. Similarly, a messaging platform that cannot efficiently transport and manage mail flow is hardly a messaging platform at all. We'll now move on to Transport Services where we'll learn how to diagnose and recover from transport issues.

Additional reading

Of front and back ends

- Front-End/Back-End topology
- Load balancing the Front-End servers
- Namespace and URL redirection configuration
- Client Access Server sizing
- Public Key Infrastructure
- Offline Address Book
- Availability Service
- Exchange Web Services
- Outlook for Mac

The Basics of Certificates

- PKI
- SSL
- Exchange 2010 Namespaces
- Exchange 2013 Namespaces
- Exchange 2016 Namespaces
- Multi-named (UCC) certificates
- AutoDiscover SRV record
- Public-Key Cryptography
- Ciphers
- Reusing an Exchange certificate
- The Certificate is Invalid for Exchange Server Usage
- Request an Exchange certificate

- Requesting an Exchange Certificate using Exchange Management Shell
- Manage local client's certificate store
- Wildcard Certificate
- Configure Trusted Roots and Disallowed Certificates
- Support for urgent Trusted Root updates for Windows Root Certificate Program in Windows
- Intermediate Certificate Authorities
- Exchange Server 2013 SSL Certificates
- When, if and how do you modify Outlook Providers?

IIS Basics and Exchange

- Troubleshooting Issues with Client Access Servers
- IISRESET vs Recycling Application Pools
- What does iisreset do in IIS 7.0?
- Using IISReset.exe to restart Internet Information Services (IIS) results in an error message
- Managing Internet Information Services with the IIS PowerShell Snap-In
- Managing Exchange 2013 IIS Virtual Directories and Web Applications
- Exchange MVP Jeff Guillet's series on building a home lab server
- Unable to open OWA, ECP, or EMS after a self-signed certificate is removed from the Exchange Back End Website
- Legacy Method to reset Exchange Virtual Directories
- Recover an Exchange Server
- Exchange admin center in Exchange Online
- Anti-Virus Software in the Operating System on Exchange Servers
- Unable to open OWA, ECP, or EMS after a self-signed certificate is removed from the Exchange Back End Website
- Exchange 2013 Clients not able to connect using Outlook Anywhere
- Removing Old Exchange 2013 Log Files
- DatabaseCopyAutoActivationPolicy Setting Breaks Client Access in Exchange 2013
- Getting Started with Log Parser Studio
- Log Parser Studio Blogs

Troubleshooting Exchange Load Balancing

- OSI model
- The OSI Model's Seven Layers Defined and Functions Explained
- Configuring SSL offloading in Exchange 2013
- SSL Acceleration and Offloading: What Are the Security Implications?
- What Is Persistence?
- Load Balancing Persistence
- Load Balancing in Exchange 2013
- Load Balancing in Exchange 2016
- Office Web Apps Server
- Office Online Server Preview
- Configure OAuth authentication between Exchange and Exchange Online organizations
- KEMP LoadMaster - Scheduling/Balancing Methods
- Tools and Techniques for Exchange Performance Troubleshooting
- Poor Outlook performance and Nagle's algorithm
- Load balancing
- Network Load Balancing Troubleshooting

- Beginner Geek: How To Edit Your Hosts File
- Issues With Load Balancing SMTP Traffic
- Troubleshooting long running MAPI connections to Exchange Server 2010 through Network Load Balancers

Validating Exchange Endpoints

- Avoiding Server Names in SSL Certificates for Exchange Server
- Microsoft Remote Connectivity Analyzer
- Exchange Server 2013: Using Test-OutlookWebServices to Verify Web Services Functionality
- Support and Recovery Assistant
- Office 365 Support and Recovery Assistant -Video
- POP3 and IMAP4
- Set-PopSettings
- Set-ImapSettings
- Protocol logging for POP3 and IMAP4
- Outlook Connectivity with MAPI over HTTP
- Configure MAPI over HTTP
- White Paper: Understanding the Exchange 2010 Autodiscover Service

Chapter 4: Troubleshooting Transport

Paul Cunningham

In Exchange terminology, "transport" (or the transport system) refers to the group of components responsible for mail flow between point A and point B. Transport has evolved significantly over the years as Microsoft released different versions of Exchange, and we've also seen changes in related technologies such as anti-spam protection schemes like the Sender Protection Framework (SPF) that also need to be taken into account along with the Exchange components.

A Brief History of Transport in Exchange Server

Exchange 2000 and Exchange 2003 could be deployed in a front-end/back-end topology so that one or more front-end servers were responsible for inbound and outbound mail flow between the internal organization and the internet. The same Exchange software was used for both types of server; the different roles were achieved through configuration of the server.

Exchange Server 2007 introduced an entirely new server role architecture and had two dedicated transport server roles that could be installed:

- Hub Transport – primarily responsible for internal mail flow, but could also be configured to send and receive email outside of the organization. Hub Transport servers were mandatory in any Active Directory site that hosted mailbox servers.
- Edge Transport – specifically designed to be deployed in a perimeter network to provide secure mail flow between the organization and the internet. The Edge Transport server was optional, and organizations could choose to deploy only Hub Transport servers instead, or use a third party product to perform the same function.

The Edge Transport server role could not co-exist on the same Windows server as other server roles, but the Hub Transport role could be installed with or without other roles, depending on whether the mailbox server role was clustered or not.

From a transport perspective, Exchange 2010 followed the same architecture as Exchange 2007. It wasn't until the release of Exchange 2013 that we saw another significant shift in transport architecture. In Exchange 2013 the majority of the transport functionality performed by the Hub Transport role is collapsed into the Mailbox server role. The Exchange 2013 Client Access server role is responsible for front-end transport services, along with the other Client Access responsibilities that you can read about in Chapter 3. The front-end transport service authenticates and proxies SMTP connections from clients to the transport services running on the Mailbox server role. This front-end/back-end architecture works in the same manner regardless of whether the Exchange 2013 Client Access and Mailbox server roles are installed on separate computers, or when they are installed together on a multi-role server.

Exchange 2016 uses the same underlying front-end/back-end transport architecture, but simplifies the server role architecture by removing the Client Access server role. All of the functionality previously performed by Client Access is now part of the Mailbox server role, a change that also simplifies deployment.

The only other server role that has survived since Exchange 2007 is the Edge Transport server, which is still designed for deployment in perimeter networks for secure inbound and outbound mail flow. Edge Transport remains as an optional server role for Exchange organizations.

Understanding Troubleshooting Scenarios for Transport

Although the evolution of Exchange Server roles over the years is important from a troubleshooting perspective, successful mail flow relies on a lot more than just the Exchange servers themselves. For an email message to travel from the sender to the recipient, many other elements are involved that we'll go through in this chapter, such as:

- The Domain Name System (DNS)
- Internet and network routing
- Firewalls
- Certificates

There is no single approach to troubleshooting transport in an Exchange environment because so many variables are at play. For example, has an email message gone missing because it was sent from Person A to Person B but never arrived? Or is it missing because Person B received it but now can't find it? Each of those scenarios is the same from an end user impact perspective – Person B doesn't have the email message they need – but are different from a troubleshooting perspective. One is a potential email delivery problem, the other is a potential data recovery scenario. And each one has a wide variety of possible root causes.

The first step is to clearly define the scope of the problem. It's rare to receive a support ticket from a help desk team or a report from end users that contains 100% of the information you need. There may be some questions you need to ask, such as:

- Who sent the email?
- Do they work for the same company or are they an external sender?
- When was it sent?
- To whom did they send it?
- Did anyone else receive the email?
- Have you been able to receive emails from that person before?
- Are others around you still receiving emails?
- Are you still receiving emails from other people, or are you receiving no emails at all?
- Is this problem specific to one device, or are you not receiving emails anywhere (e.g. Outlook, OWA, mobile)?
- Did the sender receive any error message or non-delivery report? (Not an easy one to answer if the sender is an external party)
- If you had received the email and it has now disappeared, when did you notice it was gone? When was the last time you remember seeing it?
- Does anyone else have access to your mailbox or account details (e.g. a delegate or assistant)?

If the person who reported the problem is available to speak to, they will usually be able to answer at least some of the questions listed above. If they can't answer them all (e.g. they're unsure about the time the email was sent) then you'll just need to broaden your search to account for different possibilities. I usually find that enough information comes out of a short conversation to add much needed clarity to the situation.

Basic Elimination

Depending on the answers you get for the questions you ask you should go through a short process of elimination to rule out anything that the end user wasn't able to confidently answer. For example, send them a test email from your own computer with a delivery receipt enabled, and make sure it is received in their desktop client as well as any mobile device they might use. With that one simple test you've ruled out multiple possible causes of the problem.

By the way, the delivery receipt when you are testing internal emails is important. For one thing, it means you can do the test without the other person being available to confirm delivery. It also means that you'll know that the email was delivered successfully even if the end user claims it wasn't (e.g. they have an inbox rule or some other issue preventing it from appearing in Outlook or mobile device).

The more possibilities you can rule out quickly, the easier your troubleshooting will be. However, don't assume that anything you've ruled out in the initial part of the investigation should be completely ignored. At this stage you're only trying to identify the best place to start looking. You may need to come back later to things that you ruled out and investigate those as well.

You should also consider what has changed (perhaps by you or your team) recently that may have contributed to the problem. Often we can make changes to the environment which take several days to emerge as a user-impacting problem, so make sure you consider all recent changes, not just those that occurred in the last day or so.

Understanding the Email Topology

To troubleshoot email delivery, you need to have an understanding of the environment you're working in and what the mail flow path should look like for the scenario you're dealing with. Having a transport diagram of your environment printed out on your desk or easily accessible on your computer is a good start. It should include notations for all devices or services that could impact connectivity along the way, such as firewalls, load balancers, security appliances, external smart hosts, and so on. Not only will this guide your troubleshooting but it will also highlight whether any other support teams may need to be involved in the case.

If you're dealing with a new customer and you don't have a diagram like that already then spend a few minutes at the start of the call finding out what's involved in their mail flow and sketch yourself a quick diagram. It might sound a bit basic but it is a worthwhile exercise. My notebooks are full of drawings like that from previous support cases.

Understanding the scope of the issue and the environment in which it is occurring are just the beginning of your troubleshooting. To get into more details about specific factors that can influence email delivery let's begin by looking at the role of DNS in transport.

The Critical Role of DNS

The primary role of the domain name system (DNS) is translating domain names into IP addresses so that network communications can occur. For example, when you type www.microsoft.com into your web browser, DNS is used to look up that name to determine the IP address of the server to connect to. The domain name in that example is microsoft.com.

It should be well understood by any email administrator that an email address of john@domain.com is made up of a unique prefix of "john" and a domain suffix such of "domain.com". Although in some internal systems you may see emails from local, non-resolvable addresses such as *root@localhost*, you're very unlikely to encounter those types of email addresses in email that is travelling around the internet, unless the sender happens to have a very poorly configured server. Let's simply state that the most basic requirement, as defined in RFC2821, is that "*only resolvable, fully-qualified domain names are permitted when domain names are used in SMTP*."

I'll take that one step further and point out that you should only try to use domain names that you own and control for sending email. As obvious as that may seem, I have encountered many customers in the past who try to use a "dummy" domain for a test system or for internal email alerting, and run into trouble because that domain is owned and used by someone else on the internet.

Real World: Microsoft isn't immune to making this kind of mistake. When they first shipped Exchange 2013 they used a domain name of inboundproxy.com as the destination for some of the Managed Availability probe messages. Unfortunately, they didn't own that domain at the time, and customers saw NDRs queueing on their Exchange servers for that destination. You can read the full story here.

MX Records

When an email is sent to an address of john@domain1.com, successful delivery relies on DNS to tell the sending server where to send that email. This is determined by looking up the "mail exchanger" (MX) records for domain1.com in DNS.

You can look up the MX records of a domain name using the *nslookup* utility from a CMD prompt.

```
C:\>nslookup
Default Server:  dns.iinet.net.au
Address:  203.0.178.191

> exchangeserverpro.com
Server:  dns.iinet.net.au
Address:  203.0.178.191

Non-authoritative answer:
exchangeserverpro.com   MX preference = 0, mail exchanger = exchangeserverpro-
com.mail.protection.outlook.com
```

Figure 4-1 illustrates how the lookup process by the sending Exchange server proceeds:

1. The server looks up the authoritative name servers for domain1.com.
2. The server queries the domain1.com name servers for the MX records for the domain.
3. The server looks up the host names (A records) that were returned in the MX records to determine the IP addresses of the email servers to send domain1.com email to.

That's a simplistic view of the process which may be more complicated in the real world. For example, instead of executing the queries itself, the server will query the internal DNS servers for the organization or for their internet service provider (ISP), which will either answer from previously cached responses, or will perform the queries on the server's behalf and reply with the results.

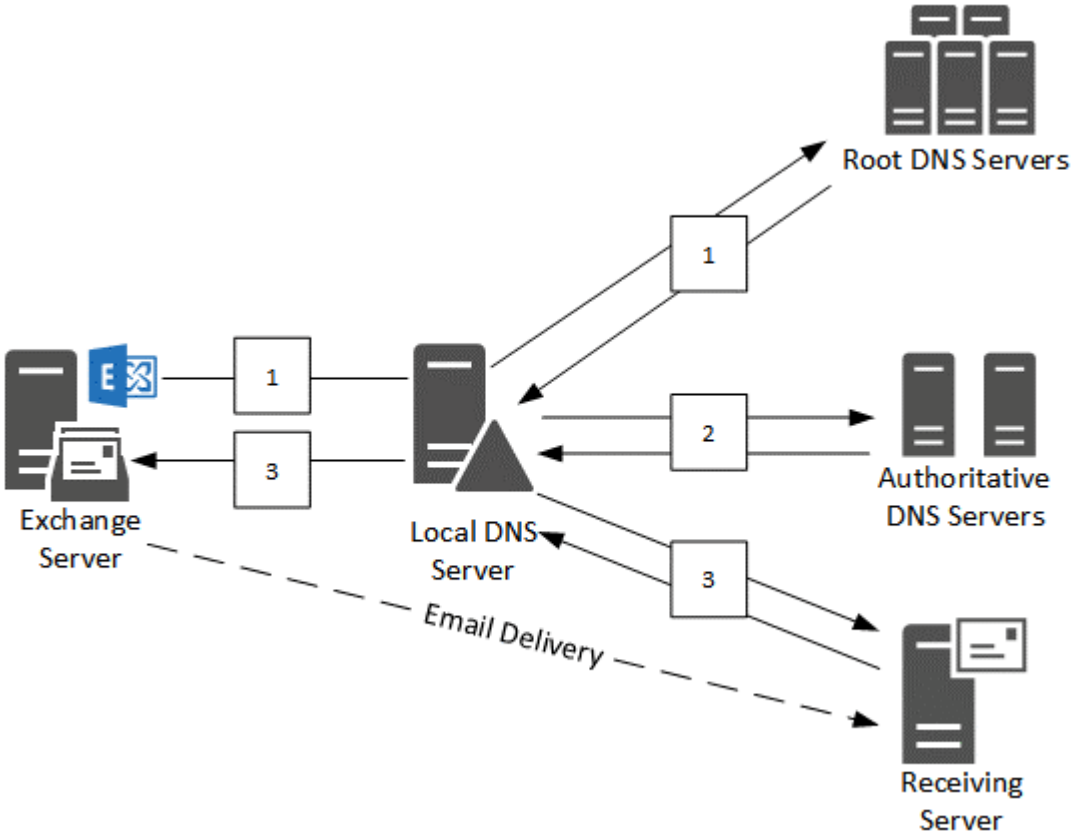


Figure 4-1: How DNS lookups work

The DNS resolution process may be even further complicated when the A records that are listed as the MX records for a domain exist in a completely different domain. For example, while domain1.com's MX records may be A records in the same domain, such as *mail.domain1.com*, the MX records for domain2.com may be something like *inbound1.emailsecurityservice.com*. This is common for domains that are using an externally hosted email security service. For example, the Australian airline Qantas has MX records that point to the Websense service.

```
> qantas.com
Server:  dns.iinet.net.au
Address:  203.0.178.191

Non-authoritative answer:
qantas.com      MX preference = 20, mail exchanger = cust20986-3.in.mailcontrol.com
qantas.com      MX preference = 20, mail exchanger = cust20986-2.in.mailcontrol.com
qantas.com      MX preference = 20, mail exchanger = cust20986-1.in.mailcontrol.com
```

In order to resolve the MX records for the Qantas domain name, the server needs to perform additional DNS queries to locate the authoritative name servers for *mailcontrol.com* as well. This is relevant in troubleshooting scenarios because the delivery of email for one domain name may be dependent on the availability of one or more other domain names. In the example above, a DNS outage for *mailcontrol.com* would have an impact on *qantas.com* and any other Websense customers as well.

By default, Exchange uses the DNS configuration of the operating system for any name resolution needs, the very same DNS servers you see in the output of *ipconfig /all*. However, Exchange can be configured with independent DNS settings at the server-level or the connector-level. These settings are discussed later in this chapter.

> **Note:** MX records in DNS are not used by your Exchange servers for internal mail flow within the organization. The Exchange servers use their own routing table for internal delivery, that takes into account the Active Directory site topology when trying to calculate the best route between two servers.

Reverse DNS

So far we've seen that MX records and A records in DNS play a part in determining where email should be delivered when sending over the internet. There's another DNS record type that is also important to have in place – a PTR (or reverse DNS) record. Resolving a host name to an IP address is the job of the A record. Resolving an IP address to a host name is the job of the PTR record. When a sending server makes an SMTP connection to a receiving server, all the receiving server can immediately see is the source IP address of the connection and the host name that the sending server is using for its opening HELO or EHLO command in the SMTP conversation.

One of the tests that most email servers perform these days is to lookup the source IP address to see if there is a PTR record for that IP address (Figure 4-2). There is no requirement for the PTR record to match the HELO hostname that the sending server is using. After all, a business with a single IP address may be running multiple services behind a reverse proxy, not just their Exchange server, so setting the PTR record to match the Exchange server's EHLO or HELO hostname isn't necessarily the correct way to go.

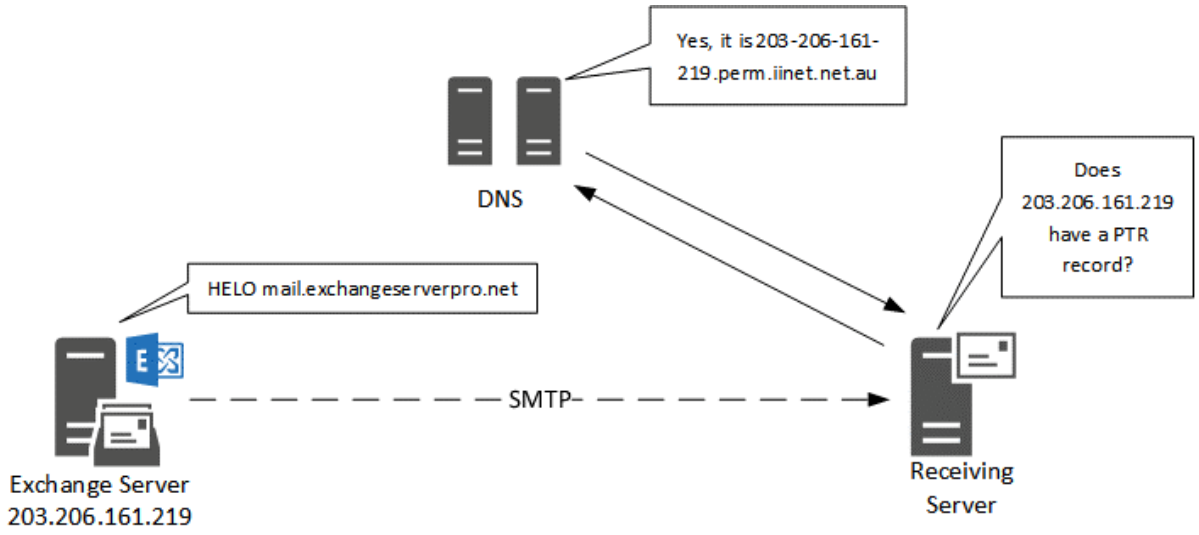


Figure 4-2: Reverse DNS – PTR check

The other test that the receiving server will perform is to check that the host name in the HELO command resolves to the same IP address that the sending server is connecting from (Figure 4-3). If an A record is found that matches the sender's source IP address, then it is likely that the connection is coming from a server that belongs to the owner of that domain.

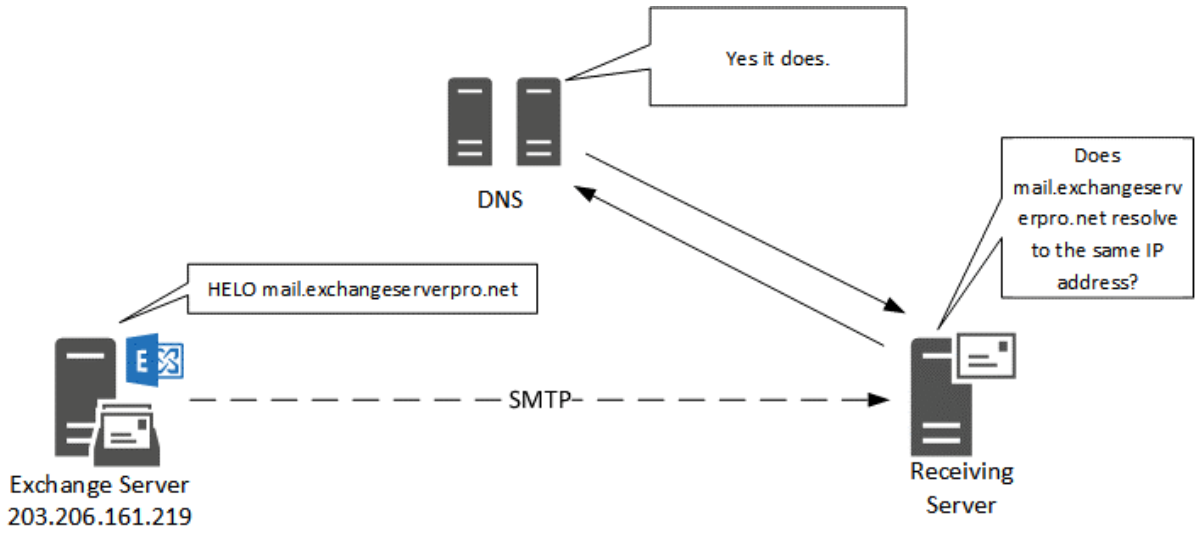


Figure 4-3: Checking the HELO host name

However, if the domain does not contain a DNS record matching the source IP address that is making the SMTP connection, then it is considered more likely that an unauthorized server is attempting to spoof the domain (Figure 4-4). In other words, it's likely to be a spammer or malicious sender.

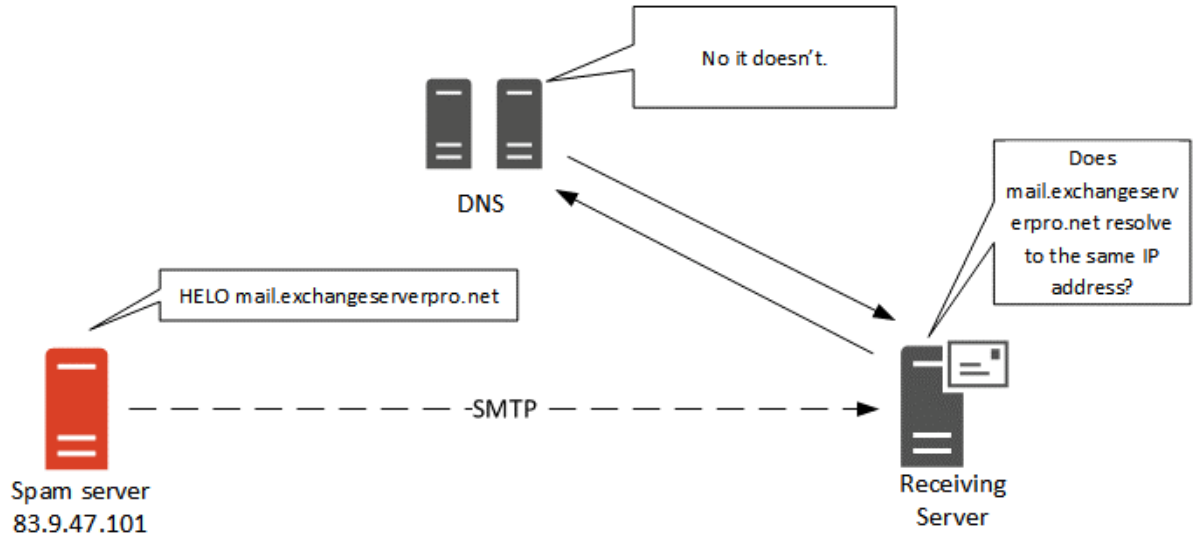


Figure 4-4: Checking for spoofing

The PTR record for an IP address can be changed by the owner of that IP address, which for most companies is the provider that is supplying you with your internet connection. The network provider will either give you access to a portal where you can manage your own PTR records, or will allow you to submit a technical support request to have a valid PTR record put in DNS for your IP address.

Without a working reverse DNS configuration for Exchange, you're likely to see SMTP connections rejected by recipients' servers, or have the emails sent by your server be scored higher on the scale of likely spam.

Similar to MX records, the PTR records are not important for internal mail flow between the Exchange servers in your own environment. Exchange will quite happily send and receive email between the other Exchange servers that it knows and trusts inside your organization whether PTR records exist or not.

SPF Records

Reverse DNS lookups aren't the only method used to detect spoofed email. Many receiving servers, particularly those operated by large email providers such as Microsoft, Google, and AOL, also perform a check of the Sender Policy Framework (SPF) record for the sender's domain when a sending server is attempting to send an email message.

SPF records allow a domain owner to specify which mail servers are permitted to send email for that domain name. When the sending server issues its "MAIL FROM" command in the SMTP conversation, the receiving server will look up the SPF record in the domain name of the "From" address to see if there is a match for the source IP address of the SMTP connection (Figure 4-5).

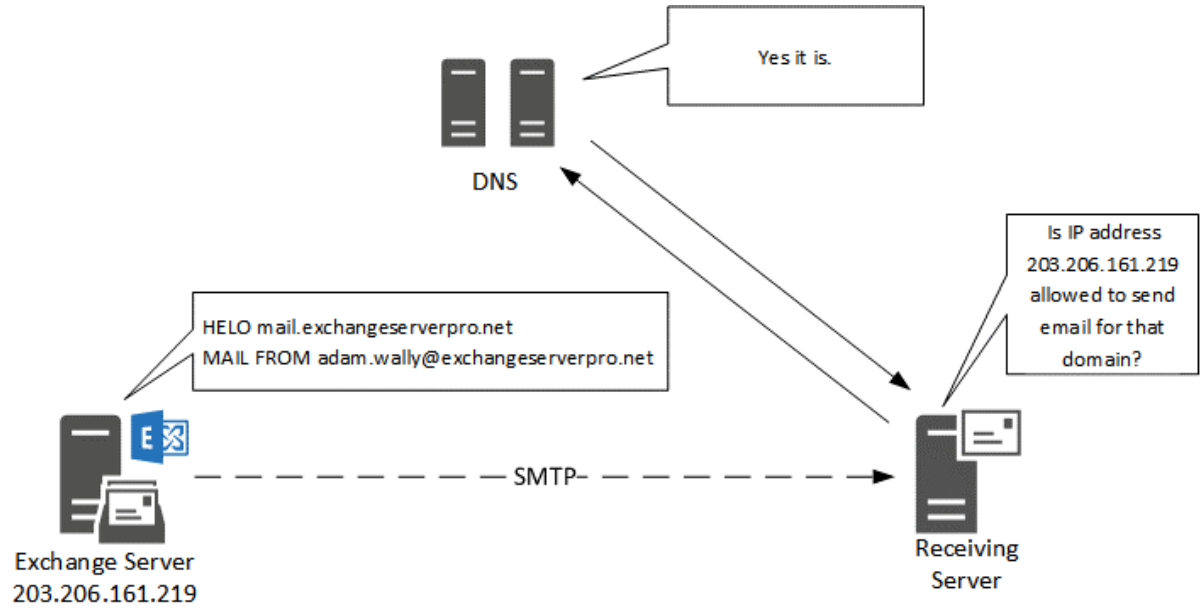


Figure 4-5: Checking a mail server through SPF records

If you read about SPF records on the internet you might find advice from some sites that it is better to have no SPF record than it is to have an incorrect SPF record. There's some truth to that, but also some risk. Some mail hosts will reject mail if there is no SPF record for the domain. Relatively few hosts tend to reject messages on that basis, but because they are very large mail hosts the impact can be quite noticeable. Ultimately it is best to have a correctly configured SPF record in DNS for your domain.

An SPF record is simply a TXT record with a certain syntax. The syntax is made up of two parts; mechanisms, and modifiers. Modifiers are optional and are not commonly used except for special circumstances. During management and troubleshooting of transport you'll most often be dealing with SPF records containing only mechanisms.

The mechanisms for an SPF record define the sets of hosts that can send email from the domain. Mechanisms can be defined by:

- **all** – matches any host, and is placed at the end of the SPF record as a "catch all" for any senders that did not match other mechanisms listed ahead of it.
- **ip4** – matches a single IPv4 address or IPv4 network range.
- **ip6** – matches a single IPv6 address of IPv6 network range.
- **a** – matches a host name or domain name. The IP addresses that the name resolves to in DNS are matched against the sender's IP address. This mechanism is useful for matching against a web server IP address based on the domain name.

- **mx** – matches against the MX records for the domain. This mechanism is useful when the outbound mail is handled by the same servers as the MX records resolve to for inbound mail.
- **ptr** – reverse DNS queries are used to match the sender IP address to the host names that it resolves to. This mechanism is generally not recommended due to the DNS load it causes.
- **exists** – simply checks that the domain exists in DNS.
- **include** – matches the sender IP against the SPF record another domain. This is commonly used when your outbound email is routing via a cloud service such as Exchange Online Protection.

Mechanisms are used in combination with a qualifier that tells the server what to do when a match is found. The qualifiers are:

- "+" for pass (this is the default if no qualifier is explicitly provided)
- "-" for fail (email from unauthorized hosts should be rejected)
- "~" for SoftFail (may result in email being accepted but marked as "likely spam")
- "?" for Neutral (regardless of the result the email should be accepted)

An example of a mechanism paired with a qualifier is "-all" at the end of an SPF record, which means "Fail/reject email from any sender who did not match an earlier mechanism in the SPF record."

If this all seems very complicated to you, don't worry, it starts out that way for everyone who has to deal with SPF records. Fortunately, there are many tools available to help you construct and validate your SPF records. For example, Microsoft provides the Sender ID Framework SPF Record Wizard, which has an awkwardly long name but is nonetheless very useful.

Figure 4-6: Running the SPF Record Wizard

After entering your domain name (Figure 4-6), the wizard will step you through a series of questions to determine the most likely SPF record that you will need. In this example I answered the questions as follows:

- Domain's inbound servers may send mail (in other words, the servers listed as MX records also handle outbound email)
- An additional domain name whose A record is a valid outbound email server (a common example of this is an externally hosted website that uses its own SMTP service to send notifications and other emails)
- This domain sends mail only from the IP addresses identified above (in other words, anything else trying to send email from my domain name should be considered unauthorized)

Figure 4-7 shows the resulting SPF record.

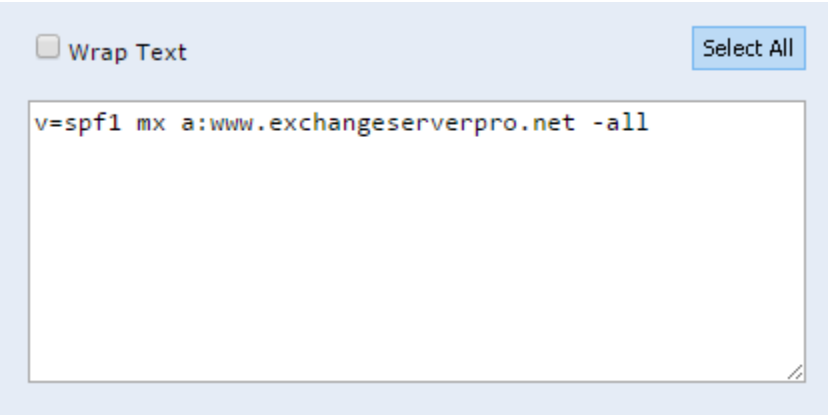


Figure 4-7: Example SPF record

Adding that string as a TXT record in the public DNS zone for the domain name helps to prevent unauthorized email servers from spoofing my domain name. At least, they won't be able to do it when sending to any receiving server that checks and enforces SPF records. Anyone who is not checking SPF records can still receive the spoofed email, but may reject it for other reasons such as spam content or malware. Although SPF is important, you should consider it just one factor among many that different providers will use when filtering email.

Apart from tools to generate your own SPF record, many email services will provide you with the exact strings to add to your SPF record. When you add a domain name to Office 365, Microsoft advises you of the SPF record they suggest, which is appropriate for organizations only sending outbound email using Exchange Online Protection. This likely may not be correct for your environment, especially in a hybrid scenario. Similarly, email marketing services and SMTP hosting services will also have documented solutions to adjust your SPF record so that you can successfully use their services without your email being rejected.

After you have your SPF record in place you should validate it. And in fact, you should repeat this validation test any time you suspect an external organization may be rejecting your email because of your SPF record. MXToolbox has an SPF record validator (Figure 4-8) that takes a domain name and IP address as input and lets you know what the result will be if that IP address sends email for your domain.

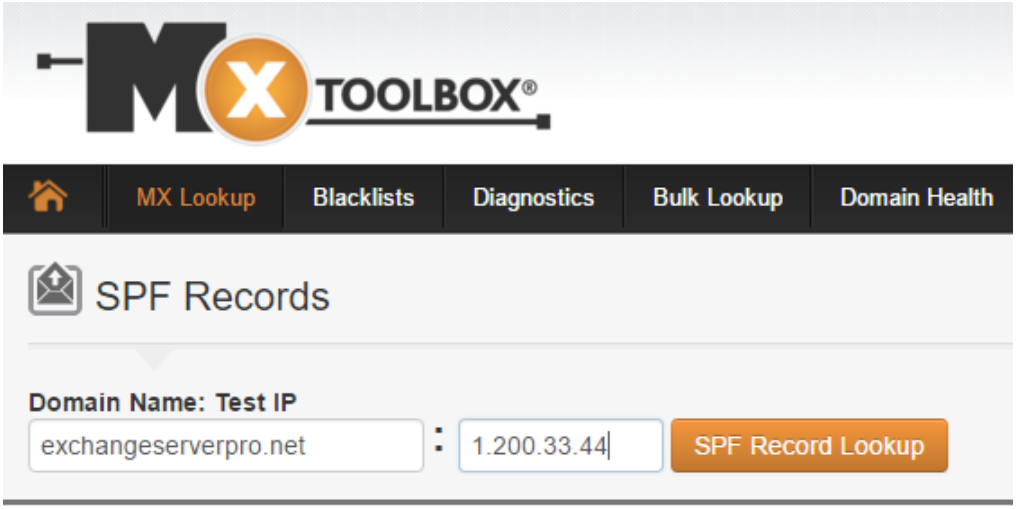


Figure 4-8: MXToolbox

Aside from the result for that specific IP address, the MXToolbox SPF record lookup tool (Figure 4-9) will also validate the general health of your SPF record for problems such as excessive DNS lookups or syntax problems.

	Test	Result	
!	SPF Included Lookups	Too many included lookups	More Info
!	SPF Test Result	SPF Failed for IP	More Info
✓	SPF Record Published	Record found	
✓	SPF Syntax Check	The record is valid	
✓	SPF Multiple Records	Less than two records found	
✓	SPF Record Deprecated	No deprecated records found	

Figure 4-9: SPF Record Lookup

Similar to MX and PTR records, the internal mail flow between Exchange servers in your organization does not depend on SPF records. The Exchange servers in an organization already understand that other Exchange servers in the same organization are authoritative for your domains.

Real World: Take care when modifying SPF records, because it is easy to inadvertently cause all of your domain's outbound email to be rejected. If there is any doubt you can use a SoftFail qualifier on the "all" mechanism (in other words, use "~all" at the end of your SPF record) for a period of time while you test outbound email against major hosts such as Yahoo and Google. Your SPF records should also be considered any time there is a planned change to your email routing.

SMTP Connectivity

So far in this chapter we've looked at how a sender's email server locates a recipient's email server when it needs to send an email message. After the sending server has located the server that it needs to connect to, it will then attempt to connect via SMTP. A variety of things can go wrong at this stage, which we'll go through in this section.

Internet and Network Routing

As with any communication between two computers, a route needs to exist between the source and the destination. For a simple server to server connection this might be as simple as traversing a single network switch. For two email servers in two different countries in the world, a lot of complex networking probably exists between the two hosts.

When you need to test that a route exists between two Exchange servers you can use the good old, reliable *ping* and *tracert* utilities in Windows. Just keep in mind that many servers will block ICMP (ping) requests, so it is not a fool-proof test. For example, here I am performing a *ping* and *tracert* from an on-premises Exchange server to an MX record for microsoft.com that resolves to Exchange Online Protection in Office 365.

```
C:\>ping microsoft-com.mail.protection.outlook.com

Pinging microsoft-com.mail.protection.outlook.com [207.46.163.215] with 32 bytes of data:
Reply from 207.46.163.215: bytes=32 time=228ms TTL=240
Reply from 207.46.163.215: bytes=32 time=228ms TTL=240
Reply from 207.46.163.215: bytes=32 time=257ms TTL=240
Reply from 207.46.163.215: bytes=32 time=227ms TTL=240

Ping statistics for 207.46.163.215:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 227ms, Maximum = 257ms, Average = 235ms
```

```
C:\>tracert microsoft-com.mail.protection.outlook.com

Tracing route to microsoft-com.mail.protection.outlook.com [207.46.163.247]
over a maximum of 30 hops:

  1    <1 ms    <1 ms    <1 ms  192.168.0.1
  2     *        *        *     Request timed out.
  3    18 ms    17 ms    18 ms  lo0.bng1.bne7.on.ii.net [150.101.32.93]
  4    18 ms    17 ms    17 ms  ae5.cr1.bne4.on.ii.net [150.101.33.162]
  5    40 ms    31 ms    34 ms  ae6.br1.syd7.on.ii.net [150.101.33.76]
  6    46 ms    60 ms    66 ms  ae0.br1.syd4.on.ii.net [150.101.33.14]
  7    31 ms    31 ms    32 ms  as12076.nsw.ix.asn.au [218.100.52.4]
  8    32 ms    32 ms    33 ms  ae1-0.syd03-96cbe-1b.ntwk.msn.net [204.152.140.113]
  9   175 ms   175 ms   175 ms  104.44.226.252
 10   204 ms   203 ms   204 ms  104.44.227.144
 11   188 ms   187 ms   188 ms  104.44.8.106
 12   183 ms   183 ms   189 ms  be-4-0.ibr01.by2.network.microsoft.com [104.44.4.3]
 13   209 ms   277 ms   187 ms  ae85-0.by2-96c-1a.ntwk.msn.net [104.44.8.189]
 14     *        *        *     Request timed out.
 15   186 ms   186 ms   186 ms  mail-by24247.inbound.protection.outlook.com [207.46.163.247]

Trace complete.
```

Note: It's useful in real troubleshooting situations to perform the *ping* and *tracert* tests from multiple hosts inside and outside of your network. If you don't have an external host to use for that type of network testing, such as a VM running in Azure, you can use tools such as Pingdom to run basic tests. Testing from multiple locations often gives you a better perspective on the problem.

Running *ping* and *tracert* tests from your server to a destination gives you part of the picture, but it doesn't necessarily mean that a working route exists between the other server and yours. Most of the Exchange servers you encounter in the real world have a private IP address and sit behind a network device that performs a Network Address Translation (NAT) from a public IP address to that private IP address.

When the network device performing the NAT has multiple public IP addresses assigned to it, there is usually one IP address defined for outbound connections, also known as source NATing. On a misconfigured network device, the NAT rule for incoming connections to the Exchange server is missing a matching source NAT rule for the reply packets. This causes the TCP connection to break, because the source of the connection is seeing reply traffic that originates from a different IP address than it is sending to (Figure 4-10).

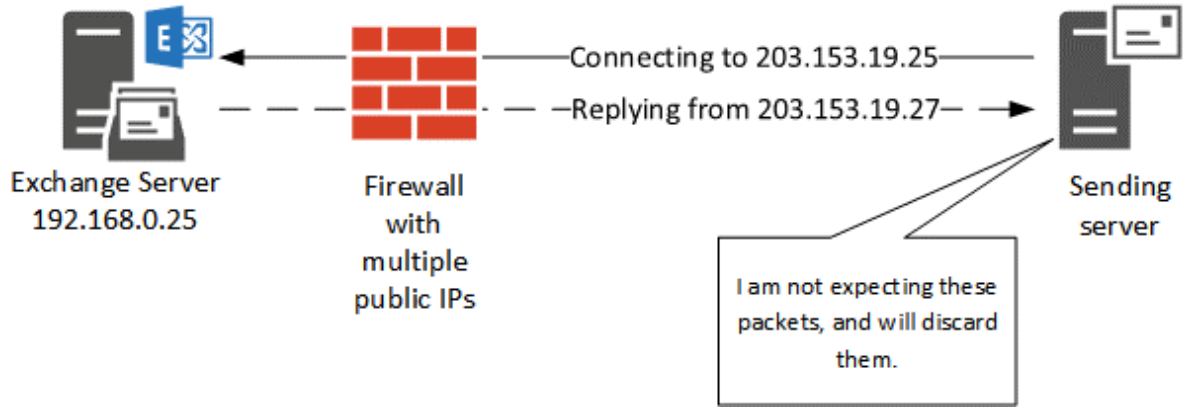


Figure 4-10: NAT and firewalls

A similar problem can occur due to asymmetric routing, which causes replies from the Exchange server to traverse a completely different network route. If that route includes a NAT device, then once again the replies will appear to be coming from a different IP address and will cause the TCP connection to break.

Both of these NAT and routing situations are difficult to diagnose from the perspective of the server itself. You can run a network packet analyser that shows that the inbound connections are being received, and that the outbound replies are being sent, but that will not give you the full picture. A network or firewall administrator will most likely need to become involved, but even then the problem may not be apparent to them, until you mention the possibility of a source NAT or asymmetric routing issue. Neither of those issues will look like a firewall is *rejecting* the traffic, so if that is all the firewall administrator is looking for then they will not see the problem.

You can also help things along by using an external test to show the public IP address to which an outbound connection from your Exchange server is NATed. Using a website such as What Is My IP is an easy way, but perhaps you would prefer not to use a web browser to connect to the internet from your Exchange servers. Instead, you can use PowerShell to retrieve the information, either from a local PowerShell console on the server, or from a remote PowerShell session to the server.

```
PS C:\> $webrequest = Invoke-WebRequest http://checkip.dyndns.com –UseBasicParsing

PS C:\> $webrequest.Content
<html><head><title>Current IP Check</title></head><body>Current IP Address:
203.206.161.219</body></html>
```

Although the output above is not very neat and tidy, the public IP address is obvious. If you'd prefer a cleaner output, try the Get-PublicIPAddress.ps1 script. Either way, providing the results to your network or firewall administrator will be helpful in a troubleshooting situation.

Connecting Through Firewalls

Finding a route between a source and destination server is one thing, but establishing an SMTP connection is another challenge entirely. As it turns out, most of us like to put firewalls in between our servers and the hostile internet. Naturally this means that some firewall rules will need to be put in place.

For SMTP connectivity the standard server-to-server port used is TCP 25. When an email server looks up the MX records for a remote domain, and tries to connect to the IP address contained in the MX record for the domain via SMTP, it will only try on the standard port. There is no way for you to run your email server on another port (which would not be helpful anyway).

However, just because servers are expecting to connect on port 25 doesn't mean different ports can't be used at all. Many client applications and devices that use SMTP can be configured to use any port you like. In fact, there are two other standard ports that are often used by clients for mail submission:

- TCP 587
- TCP 465 (this has been deprecated)

The client submission ports are most commonly used by POP and IMAP clients. POP and IMAP are old client protocols that are not widely used these days, and unless you know that some users need to use these protocols it is usually fine to leave those ports closed on your firewall.

SMTP connectivity from the internet is not all that most organizations need. Internal SMTP usage is very common as well, and firewalls come into play for internal scenarios as well. It's important to note here that placing a firewall between your Exchange servers and the clients and devices that connect to them is fully supported. The ports for client connectivity are well known and are published by Microsoft.

However, even though SMTP ports are well known, it is not supported to place firewalls between servers. Any two Exchange servers on your network should have an "IP any-any" rule on the firewall that permits unrestricted access between them.

> **Real World:** Microsoft's support statement for firewalling Exchange servers does not apply to Windows Firewall. You should leave Windows Firewall enabled on your Exchange servers, and it is supported to do so. Exchange setup creates the necessary Windows Firewall rules to allow Exchange to operate.

You can test inbound connectivity from the internet to your Exchange server on port 25 by running the Inbound Mail Test using the Microsoft Remote Connectivity Analyzer. Alternatively, you can use *Telnet* to perform a basic SMTP connectivity test. If you see the "220" response and banner from the server you're connecting to then you know that the firewall port for SMTP is open.

```
C:\> telnet microsoft-com.mail.protection.outlook.com 25

220 BY2FFO11FD004.mail.protection.outlook.com Microsoft ESMTP MAIL Service ready
```

You don't necessarily have to continue with the SMTP session and issue commands to send a test email. At this stage if all you want to prove is that SMTP connectivity is possible through the firewall, then you've achieved that. However, problems further along the SMTP conversation may still occur. To learn how to test a complete SMTP transaction read on to the next section.

> **Real World:** If you Telnet to a mail server and see a string of asterisks, like ********, instead of the normal 220 response and banner, then the mail server may be located behind a Cisco firewall that is performing SMTP inspection. This will interfere with mail flow, and should be disabled on the Cisco firewall.

SMTP Troubleshooting Tips

So far in this chapter we've seen that many different factors come into play for a successful SMTP connection to be established.

- The MX, PTR, and SPF records should be published in DNS
- A network route must exist between the source and destination
- Firewall ports must be open to allow SMTP connections

You can inspect and test each of those elements separately, as we've seen in the examples provided. You can also very efficiently test multiple factors at the same time by using some tools designed to test connectivity.

The first, and one of the best, is the Microsoft Remote Connectivity Analyzer (Figure 4-11). When you run the Inbound SMTP Test it will send an email via each of the MX records published in DNS for your domain name. This means that you will have tested DNS, SMTP connectivity, and mail flow all the way to the mailbox via every inbound route that you have an MX record published for.

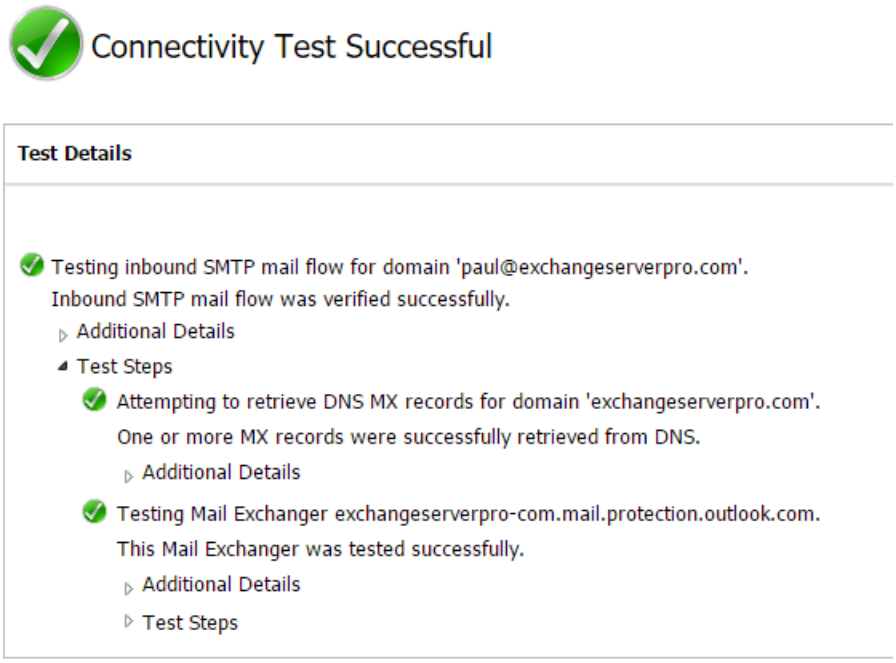


Figure 4-11: Microsoft Remote Connectivity Analyzer

Another useful tool is the SMTP diagnostic at MXToolbox (Figure 4-12). This tool will analyse your server for potential latency or performance issues, as well as validate reverse DNS and check for an open relay. Although it does not go as far as sending an actual email message, the MXToolbox SMTP diagnostic still provides very useful information for troubleshooting.

	Test	Result
!	SMTP Transaction Time	9.110 seconds - Not good! on Transaction Time
✓	SMTP Reverse DNS Mismatch	OK - 65.55.88.138 resolves to mail-me1aus010138.inbound.protection.outlook.com
✓	SMTP Valid Hostname	OK - Reverse DNS is a valid Hostname
✓	SMTP Banner Check	OK - Reverse DNS matches SMTP Banner
✓	SMTP TLS	OK - Supports TLS.
✓	SMTP Connection Time	1.125 seconds - Good on Connection time
✓	SMTP Open Relay	OK - Not an open relay.

Figure 4-12: SMTP Diagnostics

Watch Out for ISPs and Security Products That Block SMTP

One of the frustrating things about SMTP, and email in general, is how heavily it is abused by spammers and malicious attackers. Mailboxes are constantly bombarded by spam, phishing attempts, malware, and attackers looking for misconfigured email servers to exploit.

The risk of attack has led to many ISPs and network providers blocking SMTP ports. For residential customers in particular, outbound SMTP on TCP port 25 is often blocked with the exception of the mail hosts that the ISP operate. Customers can still send email using TCP port 587 but are generally prevented from propagating "mass mailing worms" if their computers become infected by malware.

The same ISPs often also block SMTP for their business customers, until the customer specifically requests that the block is removed. In this situation, you should control outbound SMTP access using your own network firewall, to prevent malware infections from causing spam to originate from your network, which would harm the reputation of your IP addresses. If you're able to establish an SMTP connection to the ISP's mail server, but not to other email servers on the internet, then they are almost certainly blocking SMTP and you'll need to contact their support team to request the block be removed.

Aside from ISPs, it is also common for "enterprise security" products to block SMTP connectivity. Usually, these products are installed on all of the computers and servers on a network, and if care is not taken to configure the software on your Exchange servers correctly you may find that the security product blocks outbound SMTP connections from Exchange, which will of course stop mail flow from working.

The Transport Pipeline

Microsoft describes the transport pipeline as the collection of services, connections, components, and queues that work together to route all messages to the categorizer in the Transport service. That definition remains fairly consistent across all recent versions of Exchange, but the underlying components have changed over time as the server architecture changed. For example, in Exchange 2010 the categorizer is part of the Hub Transport server, whereas in Exchange 2013 and Exchange 2016 the categorizer is part of the Mailbox server.

When you're considering a troubleshooting scenario, a basic understanding of the transport pipeline is helpful for trying to visualize the problem. An email message does not simply pass from mailbox to mailbox; instead it passes through multiple services, potentially across multiple servers, with specific components and connectors being responsible for processing or passing the message along.

The categorizer is at the heart of Exchange transport. This component processes each message in the submission queue to perform recipient resolution, routing, and any content conversion that may be necessary. Each message can be different from the last, for example an email sent to a distribution list that has members spread across multiple sites in the organization will have the distribution list expanded, and then be split into multiple messages to be routed to the different sites where the mailboxes are hosted. Such a message will look different to a simpler email message sent from one mailbox to another when you dig into logs and message tracking.

The operation of the transport pipeline will become clearer as you become more familiar with using different troubleshooting logs and tools to investigate email delivery problems.

Transport Services

There are multiple transport services hosted on Exchange 2013 and Exchange 2016 servers. The key difference is that in Exchange 2013 the Front End Transport service is hosted on the Client Access server role, whereas in Exchange 2016 it is hosted by the Mailbox server role because the Client Access role no longer exists.

- Front End Transport Service – this service simple acts as a stateless proxy for inbound SMTP connections on TCP ports 25 and 587. There's no queueing, logging, or other handling of the actual messages contents by the Front End Transport service.
- Transport Service – this service handles all SMTP mail flow within the organization, acting as a go-between for the Front End Transport service and the Mailbox Transport services on the same or other Exchange servers. The Transport service is the only service where queueing of messages occurs.
- Mailbox Transport service – this service is actually made up of two separate services; the Mailbox Transport Submission service, which is responsible for connecting to mailbox databases to retrieve messages and submit them to the submission queue on the Transport service, and the Mailbox

Transport Delivery service which is responsible for passing messages from the Transport service to the local mailbox databases.

Exchange 2013 and Exchange 2016 Edge Transport servers use a Transport service that is similar to the Transport service on the Mailbox server role.

The most common cause of a Transport service failing to start is a corrupt transport database. If you suspect the transport database is at fault, then you can move the database files to a different location (or simply rename the folder), then start the Transport service again. A new, empty database will be created.

> **Note:** If you move or remove the database then any messages that were still queued for delivery will be lost. The transport database can sometimes be repaired with the ESEUTIL program in the same way as mailbox databases, which is discussed in chapter 9. If you move a repaired database back to its original location and start the Transport service again, then any messages queued in the database will be retried for delivery.

Receive Connectors

Email messages are passed between clients, servers, and transport services using connectors. As the names make quite obvious, receive connectors are used to receive messages, and send connectors are used to send messages.

Every Exchange 2013 or Exchange 2016 server is configured with a set of default receive connectors that follow a standard naming convention that includes the server name.

Three default receive connectors are homed on the Front End Transport service:

- Default Frontend SERVERNAME – this connector listens on TCP port 25 to accept SMTP connections, and acts as the entry point for email into the Exchange organization. It is pre-configured to perform that role, and there are few scenarios in which modifying this connector is appropriate. As a general rule I recommend you do not disable, modify, or remove it. Think of this receive connector as the connector of last resort.
- Client Frontend SERVERNAME – this connector requires client authentication, accepts secure SMTP connections (with TLS applied), and listens on port TCP 587.
- Outbound Proxy Frontend SERVERNAME – this connector listens on TCP port 717 and is only used when a send connector is configured to proxy outbound email through the front end servers or services. This is an optional configuration, and not very common.

Two default receive connectors are homed on the Transport service:

- Default SERVERNAME – this connector accepts connections from other Mailbox and Edge Transport servers using TCP port 2525. Clients do not connect directly to this connector, and there are no reasons for you to ever disable, modify, or remove this connector.
- Client Proxy SERVERNAME – this connector accepts connections from Front End Transport services on the same server, or on other servers, using TCP port 465. Again, clients do not connect directly to this connector, and you should not disable, modify, or remove it.

Receive connector conflicts are a common source of Exchange Server problems. There are two ways that receive connector conflicts can occur.

The first is when two receive connectors homed on different services are configured to listen on the same IP address and port number. This will cause one or both of the services to fail to start, and an event log entry will

be logged to the Application event log reporting the conflict. To avoid this problem, any receive connectors that you create should be homed on the Front End Transport service, where they can all listen on the same IP address and port number without conflicting. Unfortunately, the default role selected during the new receive connector wizard is "Hub", which is the wrong one.

Fortunately, the most recent builds of Exchange 2013, and all builds of Exchange 2016, stop you from creating a receive connector conflict. However, you may still encounter the problem on older installations of Exchange 2013.

Another problem caused by two receive connectors that have an IP address and port conflict is that the installation of cumulative updates may fail, because a check for conflicting receive connectors is performed during the upgrade process. Unfortunately, when this occurs it leaves the server in a non-working state, because setup fails *after* it has already removed the existing Exchange installation.

```
Mailbox role: Transport service FAILED
The following error was generated when "$error.Clear();
$connectors = Get-ReceiveConnector -Server $RoleFqdnOrName;
foreach($connector in $connectors) { if($connector.MaxLocalHopCount -gt 1) { Set-ReceiveConnector -
Identity $connector.Identity -MaxLocalHopCount 5 }};
" was run: "Microsoft.Exchange.Management.SystemConfiguration
Tasks.ReceiveConnectorRoleConflictException: The values that you specified for the Bindings and
RemoteIPRanges parameters conflict with the settings on Receive connector "EX2013SRV2\Test". Receive
connectors assigned to different Transport roles on a single server must listen on unique local IP
address & port bindings.
```

In amongst all of that error information is the clue that you need to find the conflicting connector. In this example the connector is named "Test" and is configured on the server "EX2013SRV2".

```
The values that you specified for the Bindings and RemoteIPRanges parameters conflict with the settings
on Receive connector "EX2013SRV2\Test". Receive connectors assigned to different Transport roles on a
single server must listen on unique local IP address & port bindings.
```

To fix the conflict you can simply run *Set-ReceiveConnector* to modify the transport role that the connector is homed on.

```
[PS] C:\>Set-ReceiveConnector EX2013SRV2\Test -TransportRole FrontendTransport
```

If you only have one Exchange server in the organization, you won't be able to use the Exchange Management Shell (EMS) to run that command. Instead, you will need to use the ADSIEdit utility to adjust the connector's configuration. Follow these steps.

- Launch *ADSIEdit.msc* and connect to the *well-known Naming Context* of **Configuration**.
- The receive connector objects can be found in **Configuration -> Services -> Microsoft Exchange -> Your Organization Name -> Administrative Groups -> Exchange Administrative Group (FYDIBOHF23SPDLT) -> Servers -> SERVERNAME -> Protocols -> SMTP Receive Connectors**.
- The attribute we're interested in is the *msExchSmtpReceiveRole*. Figure 4-13 shows two receive connectors. The one on the left has the role attribute set to *16384*, which is Frontend Transport. The one on the right is set to *32*, which is Hub Transport.
- Modify the value from 32 to 16384 on the offending receive connector and then apply the change. After taking either of the corrective actions above you can then re-run the cumulative update and setup should proceed past the failed step and complete successfully.

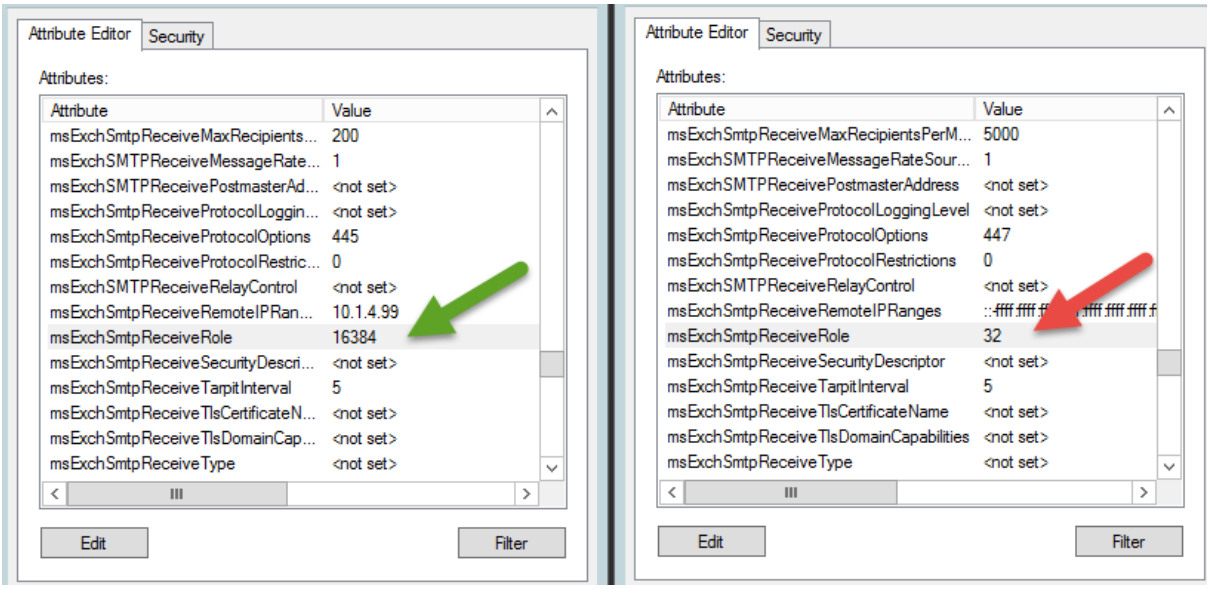


Figure 4-13: Updating *msExchSmtpReceiveRole* with ADSIEdit

The second common problem scenario is when two receive connectors have overlapping remote network IP ranges (Figure 4-14), which can cause an inbound SMTP connection to be handled by a receive connector that you were not expecting. A common example is when a custom receive connector has been configured to allow SMTP relay for specific applications and devices on the network. However, when you attempt to relay messages to external recipients from those applications or devices, the message is rejected.

The key to understanding overlapping remote network IP ranges, which are normal and not necessarily a problem, is that the most specific match will win. The "Default Front End" connector on the server has a remote IP range of "0.0.0.0-255.255.255.255", which effectively means "anything". The "Default Front End" connector will therefore handle any incoming SMTP connection, unless another connector with has a more specific match in its remote IP ranges for the source IP address of the connection.

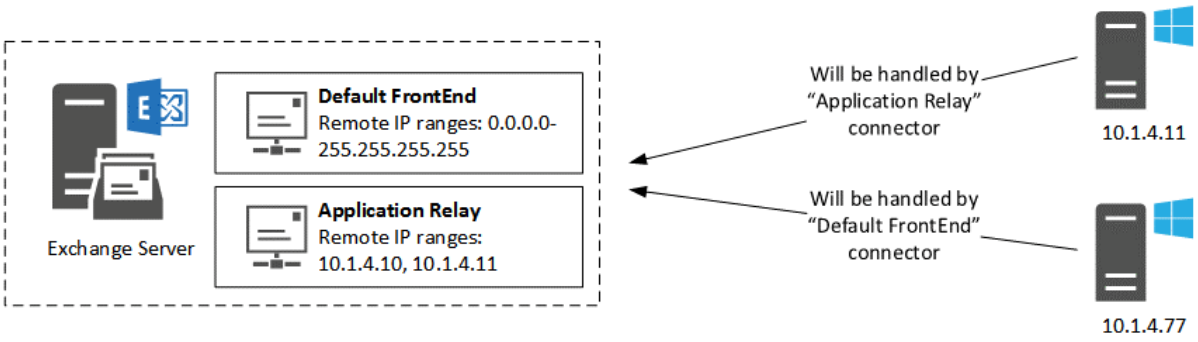


Figure 4-14: Overlapping IP ranges

That source IP might change if the application or device is connecting from behind a firewall or NAT device, if the device is DHCP-enabled, or if the device has multiple IP addresses. Care must be taken to define the correct IP addresses in the remote IP ranges of any custom receive connectors.

Warning: Some administrators add entire subnets or large ranges of IP addresses to their custom connectors. For example, they might add the entire server VLAN, so that any server in their datacenter can use Exchange for SMTP relay. This is problematic if that IP range also includes other Exchange servers, because it may result in Exchange server-to-server communications breaking if the connections

are handled by the wrong connector. If you do add IP address ranges to your custom receive connectors you should take care to exclude the IP addresses of any Exchange servers.

When you suspect that the wrong receive connector is handling SMTP connections from a particular source, there are two techniques you can use to troubleshoot the situation. The first is to modify the SMTP banner of the receive connectors so that each one is unique. This can be easily applied by running two PowerShell commands to set the SMTP banner on each connector so that it displays the receive connector name, demonstrated in Jeff Guillet's blog post.

```
[PS] C:\>$rc = Get-ReceiveConnector -Server EX2013SRV2

[PS] C:\>$rc | % {Set-ReceiveConnector $_.Identity -ProtocolLoggingLevel Verbose  -Banner "220 $_"}
```

The next time that you Telnet to the Exchange server from the device or server that you're troubleshooting SMTP relay for, you'll see the new value in the SMTP banner telling you the connector name that has accepted your Telnet connection.

```
C:\>telnet ex2013srv2 25

220 EX2013SRV2\Application Relay EX2013SRV2
```

If you see an unexpected receive connector name, you can then revisit your configuration to troubleshoot further.

The other technique is to enabled protocol logging on the receive connector, which is discussed in a later section of this chapter.

Send Connectors

In contrast to receive connectors, no send connectors are configured by default for an Exchange Server installation. All send connectors must be manually configured, unless they are automatically configured by an Edge Subscription, and serve to route email messages to servers outside the Exchange organization.

Send connectors rely on several elements that we've already discussed in this chapter:

- Network connectivity
- DNS (if MX records are used for delivery instead of a smart host)
- Firewalls

In other words, a transport server that is trying to send email must be able to resolve the recipient's domain in DNS (if MX records are being used), and then connect to the smart host or MX for that domain across the network and internet through any firewalls.

In addition, send connectors can be enabled for protocol logging to assist you with troubleshooting, which is discussed in a later section of this chapter.

Send connectors have three attributes that control whether outbound email will be sent using that connector. The first two attributes are the cost, and the address space.

The cost of the send connector is included in the aggregate cost calculated by Exchange when determining each possible route to a destination. The cost determines the priority order in which the Exchange server will

consider multiple available send connectors that match the recipient's address space. The lower the aggregate cost of the route, the higher priority, similar to the cost that you configure for MX records in DNS.

If the aggregate cost of two routes to the destination address space exist, then the cost of the send connector itself is used as a tie-breaker. If two send connectors with equal costs exist, then the connector with the alphabetically lower name is chosen.

> **Note:** Aggregate cost refers to the total cost of all site links and connectors on the route to a destination. This is more relevant for multi-site environments where internal Exchange servers may need to route email via other internal servers to reach the internet. This includes AD Site links, which you can view by running *Get-ADSiteLink*. Exchange will use the *ADCost* of an AD Site link, unless a specific *ExchangeCost* has been configured.

The address space defines the domain names that a send connector can be used to send email to. At least one send connector with an address space of "*" is necessary for outbound email to the internet to work. Alternatively, a send connector can have a specific address space configured on it, and will only be used for email addressed to that namespace. This is commonly used to control mail flow to specific partners, or to internal systems that use a domain name that is not publicly resolvable.

Address space matching takes precedence over other factors. In the example shown in Figure 4-15, messages to the *contoso.com* domain will be handled by *Send Connector 2*, even though it has a higher cost, because it matches the address space.

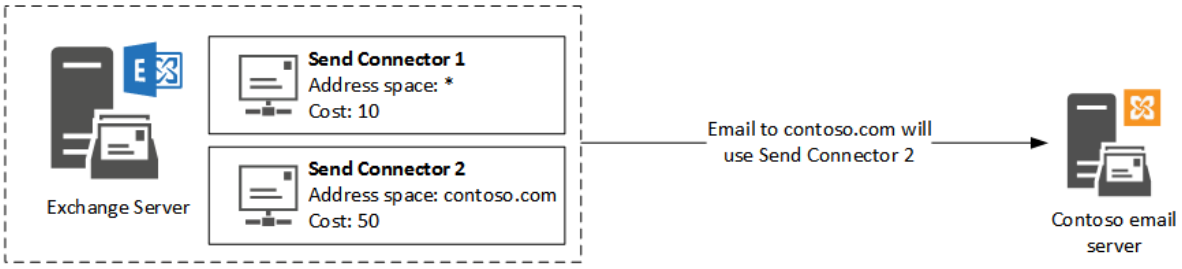


Figure 4-15: Connector costs

Once a send connector has been chosen for delivering an email message that decision is final. Exchange will not attempt to choose another send connector, even if the email message is stuck in a queue, unless you restart the transport service, which will cause all messages in the queue to be re-evaluated for routing. Troubleshooting transport queues is covered later in this chapter.

A common problem scenario is when two send connectors have been configured with matching namespaces (such as "*" or "contoso.com") and matching costs, with the expectation that Exchange will load balance email across both connectors. This is not correct. In the case of two otherwise matching send connectors, the connector with the alphabetically lower name is always used. If for some reason that connector can't deliver the email message, for example because the smart host configured on that send connector is unreachable, then all email messages will be stuck in a queue instead of "failing over" to the other matching send connector.

> **Real World:** If you need outbound email to load balance or fail over between multiple smart hosts, then you should configure one send connector with all of the smart hosts added to that connector. Do not configure one send connector per smart host.

One more attribute to be aware of for send connectors is the concept of scoped send connectors. By default, send connectors are not scoped. If you do choose to enable a send connector to be a scoped send connector, then it will only be considered an available connector for outbound email routing for other Exchange servers in the same Active Directory site as the source transport servers configured on that connector. Scoped send connectors are not commonly used, but if you do encounter issues where servers in one site appear to be not using a send connector in another site, then check whether that send connector is a scoped connector.

Troubleshooting Transport

Several logs and tools that can be used to troubleshoot transport. Keep in mind that as I described at the start of this chapter, each transport or email delivery problem can be quite different from others. This means that the logs or tools that you use will vary in usefulness depending on the scenario.

Protocol Logging

Protocol logs capture the SMTP communications that occur between servers. The information that is written to the protocol log files looks very similar (Figure 4-16) to what you see when you are using Telnet to make an SMTP connection.

```
134.170.52.123:56952,>,"220 mail.exchangeserverpro.net Microsoft ESMTP MAIL Service ready at Wed, 16 Dec 2015 17:29:58 +1000",
134.170.52.123:56952,<,EHLO www.testexchangeconnectivity.com,
134.170.52.123:56952,*,SMTPSubmit SMTPAcceptAnySender SMTPAcceptAuthoritativeDomainSender AcceptRoutingHeaders,Set Session Permissions
134.170.52.123:56952,>,250-mail.exchangeserverpro.net Hello [134.170.52.123],
134.170.52.123:56952,>,250-SIZE 36700160,
134.170.52.123:56952,>,250-PIPELINING,
134.170.52.123:56952,>,250-DSN,
134.170.52.123:56952,>,250-ENHANCEDSTATUSCODES,
,134.170.52.123:56952,>,250-STARTTLS,
,134.170.52.123:56952,>,250-X-ANONYMOUSTLS,
,134.170.52.123:56952,>,250-X-EXPS NTLM,
,134.170.52.123:56952,>,250-8BITMIME,
,134.170.52.123:56952,>,250-BINARYMIME,
,134.170.52.123:56952,>,250-CHUNKING,
,134.170.52.123:56952,>,250-XEXCH50,
,134.170.52.123:56952,>,250 XSHADOW,
,134.170.52.123:56952,<,MAIL FROM:<Admin@TestExchangeConnectivity.com>,
,134.170.52.123:56952,*,SMTPSubmit SMTPAcceptAnySender SMTPAcceptAuthoritativeDomainSender AcceptRoutingHeaders,Set Session Permissions
,134.170.52.123:56952,*,08D3051E24C9453B;2015-12-16T07:29:58.968Z;1,receiving message
,134.170.52.123:56952,>,250 2.1.0 Sender OK,
,134.170.52.123:56952,<,RCPT TO:<adam.wally@exchangeserverpro.net>,
,134.170.52.123:56952,>,250 2.1.5 Recipient OK,
,134.170.52.123:56952,<,DATA,
.134.170.52.123:56952,>,354 Start mail input; end with <CRLF>.<CRLF>.
```

Figure 4-16: SMTP protocol logging

The information in protocol logs is invaluable in troubleshooting scenarios, because it captures events that occur during message delivery that may not appear in other logs on the server.

For example, many administrators are used to looking in message tracking logs when they troubleshoot email delivery. Message tracking logs only record events for messages once they are in the transport pipeline. If a message is never sent/received because the SMTP connection itself is rejected, the message tracking log will show no useful troubleshooting information.

There are two parts to the configuration of protocol logging in Exchange and they are basically the same across all versions from Exchange 2007 onwards. First, there is the per-service protocol log settings on server roles that host transport services. The per-service settings are configured automatically with the following default settings:

- Maximum log age of 30 days. This effectively means a retention period of 30 days, with any log files older than 30 days being automatically deleted from the server's disks.
- Maximum log directory size of 250 MB. If the maximum directory size limit is reached before the maximum log age limit, then the oldest log files will be removed from the server's disks. Which means that it's possible to find fewer log files than you might expect if you were measuring on the log age

alone. The maximum directory size is intended to prevent the server's disks from filling up with protocol log files.
- A log path that is located within the Exchange install directory.

Each transport service has its own protocol log settings. This means that:

- Exchange 2010 Hub Transport servers have one set of "Hub" protocol log files and settings.
- Exchange 2013 Client Access servers have one set of "Frontend" protocol log files and settings.
- Exchange 2013 Mailbox servers have one set of "Hub" protocol log files and settings (even though it is referred to as the "Transport" service).
- Exchange 2013 multi-role servers have both "Frontend" and "Hub" protocol log files and settings.
- Exchange 2016 Mailbox servers also have both "Frontend" and "Hub" protocol log files and settings.
- Any Edge Transport server has one set of "Edge" protocol log files and settings.

To expand even further on that, each transport service has separate "Receive" and "Send" protocol log files and settings. To demonstrate, here's an example of an Exchange 2013 Frontend Transport service's protocol log settings.

```
[PS] C:\>Get-FrontendTransportService -Identity EX2013SRV1 | fl *protocol*


ExternalDNSProtocolOption             : Any
InternalDNSProtocolOption             : Any
IntraOrgConnectorProtocolLoggingLevel : Verbose
ReceiveProtocolLogMaxAge              : 30.00:00:00
ReceiveProtocolLogMaxDirectorySize    : 250 MB (262,144,000 bytes)
ReceiveProtocolLogMaxFileSize         : 10 MB (10,485,760 bytes)
ReceiveProtocolLogPath                : C:\Program Files\Microsoft\Exchange
                                        Server\V15\TransportRoles\Logs\FrontEnd\ProtocolLog\SmtpReceive
SendProtocolLogMaxAge                 : 30.00:00:00
SendProtocolLogMaxDirectorySize       : 250 MB (262,144,000 bytes)
SendProtocolLogMaxFileSize            : 10 MB (10,485,760 bytes)
SendProtocolLogPath                   : C:\Program Files\Microsoft\Exchange
                                        Server\V15\TransportRoles\Logs\FrontEnd\ProtocolLog\SmtpSend
```

Of the available per-service settings that you can modify, the most likely ones that you will adjust are the maximum age and maximum directory size. If you need to retain a very large quantity of protocol log files, you may also consider moving the log path to a different volume.

Here's an example of increasing the maximum log age to 60 days, and the maximum directory size to 1 GB.

```
[PS] C:\>Set-FrontendTransportService -Identity EX2013SRV1 -ReceiveProtocolLogMaxAge 60.00:00:00 -
ReceiveProtocolLogMaxDirectorySize 1GB
```

Ultimately it will depend on your environment how much protocol log data you need to retain, which will be the basis of any changes you make to the configuration.

Real World: The servers where protocol log data is the most valuable are those that sit on a critical path for mail flow in your organization. Examples include the internet-facing servers and any servers that are targets for internal applications that need SMTP access. On those servers you'll often need to use protocol log data to prove or disprove a mail flow issue.

After the per-server settings for protocol logging you also need to look at the per-connector settings. Each send or receive connector in your organization has two possible settings for protocol logging; verbose, or

none. By default, most connectors are set to none. It is recommended to set connectors to verbose instead, so that the log data is already generated and available when you need it for troubleshooting.

Protocol logging is enabled using the *Set-SendConnector* or *Set-ReceiveConnector* cmdlets. For example, to enable verbose logging for all receive connectors on a server you can run the following command.

```
[PS] C:\>Get-ReceiveConnector -Server EX2013SRV1 | Set-ReceiveConnector -ProtocolLoggingLevel Verbose
```

When you need to use protocol log data to troubleshoot a mail flow issue, one of the simplest approaches is to search the log files using PowerShell. Here's an example scenario. In this scenario an email has been sent by exchangeserverpro@gmail.com to alan.reid@exchangeserverpro.net within the last 24 hours, and we want to determine what happened during the SMTP session from Gmail's servers to the Edge Transport server in the Exchange organization.

On the Edge Transport server, we can open PowerShell and change to the directory containing the *receive* protocol log files.

```
[PS] C:\>Set-Location (Get-TransportService).ReceiveProtocolLogPath
```

From that location we can ingest the contents of the last 24 hour's protocol log files into a variable.

```
[PS] ..> $logs = Get-ChildItem | Where {$_.LastWriteTime -gt (Get-Date).AddHours(-24)} | Get-Content
```

Now we can parse the data in *$logs* for the sender's email address.

```
[PS] ..> $logs | Select-String "exchangeserverpro@gmail.com"

2015-12-21T12:26:23.159Z,EX2013EDGE\Default internal receive connector
EX2013EDGE,08D3051E24C97D16,61,192.168.0.25:25,209.85.215.68:34539,<,MAIL
FROM:<exchangeserverpro@gmail.com> SIZE=1645,
```

We get one hit, which provides a key piece of information to locate the rest of the log data for that SMTP session. Each protocol log line uses the following fields:

```
date-time,connector-id,session-id,sequence-number,local-endpoint,remote-endpoint,event,data,context
```

Since every SMTP session has a unique "session-id", we can parse the *$logs* data for all of the lines that include the session ID that we're interested in, which in this case is "08D3051E24C97D16".

```
[PS] ..> $logs | Select-String "08D3051E24C97D16"
```

When you do that for the first time you'll notice that quite a lot of data is output, and sometimes a PowerShell window (Figure 4-17) is not the easiest place to read it due to wrapping of lines.

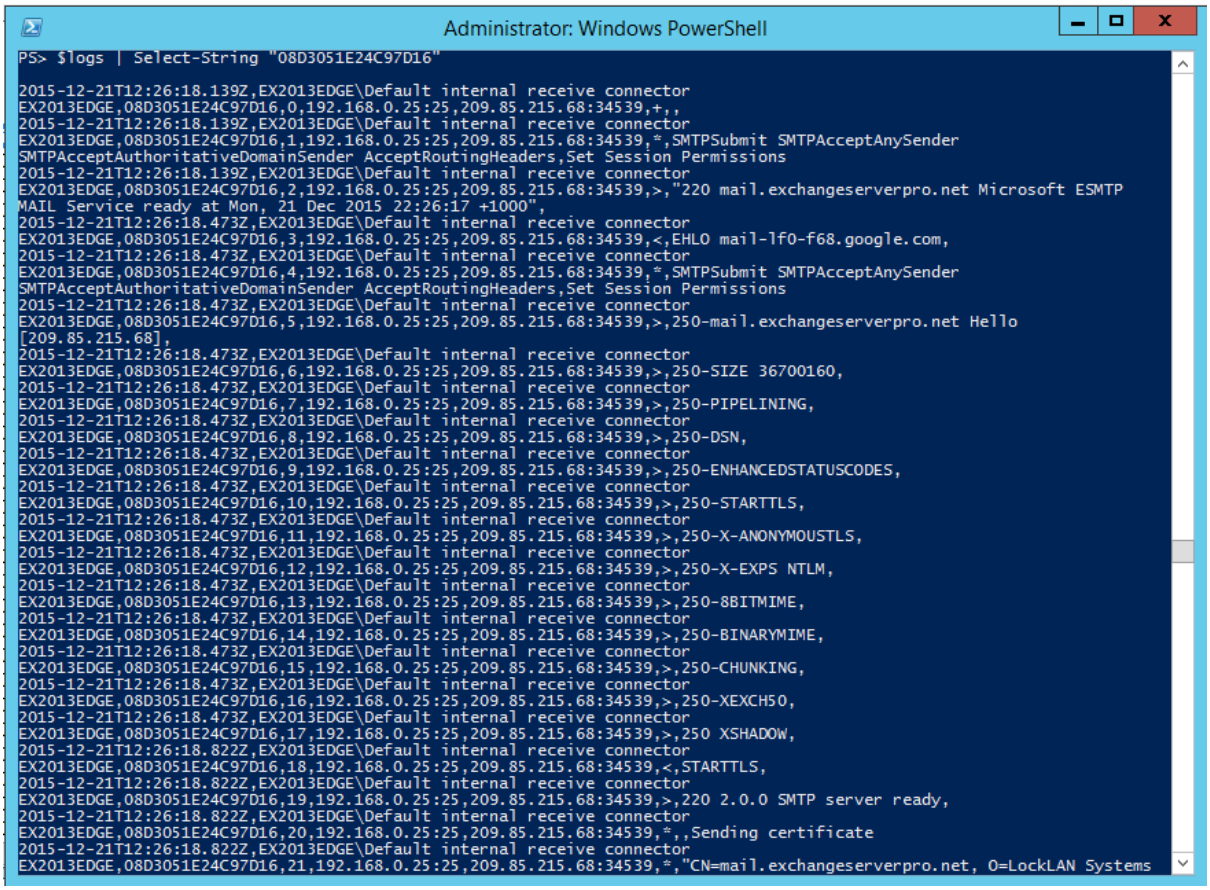


Figure 4-17: Searching through a log file with PowerShell

Sending the output to a text file to open in Notepad is sometimes a better choice.

```
[PS] ..> $logs | Select-String "08D3051E24C97D16" | Set-Content C:\Temp\protolog.txt

[PS] ..> Notepad C:\Temp\protolog.txt
```

If the protocol logs show that an SMTP session was successful and that the email message was accepted by the server, then the next step in troubleshooting would be to look further into the pipeline using message tracking.

Warning: The IMAP and POP services on Exchange servers also have protocol logging available. However, unlike Transport protocol logging, IMAP and POP do not have maximum directory size thresholds. Which means that the IMAP or POP protocol log data will grow with no limits, eventually consuming all available disk space on the volume. As such, you should not leave IMAP or POP protocol logging enabled, unless you have implemented a separate process for removing old log files.

Message Tracking

Message tracking is an Exchange feature that records detailed log files of email traffic as messages are transferred between Exchange servers within the organization, and between different roles, services and components on individual servers. In other words, message tracking logs record what happens to a message as it passes through the transport pipeline.

A default configuration for Exchange 2010 and Exchange 2013 is:

- Message tracking is enabled
- Log retention is 30 days
- The maximum log directory size is 1000MB (1GB)
- The maximum log file size is 10MB
- The tracking log directory is set to *<install path>\TransportRoles\Logs\MessageTracking*
- Subject logging is enabled

Here is an example of how the default configuration looks when queried using PowerShell.

```
[PS] C:\>Get-TransportService -Identity EX2013SRV1 | fl messagetracking*


MessageTrackingLogEnabled               : True
MessageTrackingLogMaxAge                : 30.00:00:00
MessageTrackingLogMaxDirectorySize      : 1000 MB (1,048,576,000 bytes)
MessageTrackingLogMaxFileSize           : 10 MB (10,485,760 bytes)
MessageTrackingLogPath                  : C:\Program Files\Microsoft\Exchange
                                          Server\V15\TransportRoles\Logs\MessageTracking
MessageTrackingLogSubjectLoggingEnabled : True
```

Similar to protocol logs discussed earlier in this chapter, the message tracking logs have both a maximum log age, and a maximum directory size, which work together to prevent the log files from consuming all available free disk space on the volume. However, the maximum directory size applies to each different type of message tracking log file, and there are four types of log files on Exchange 2013 and Exchange 2016. One of the log types is infrequently used, so Microsoft recommends that you apply a 3x multiplier to the maximum directory size value when you are calculating the amount of disk space it will actually use. In other words, if the maximum log directory size is configured for 1 GB, allow for 3 GB of log files to accumulate.

Note: In Exchange 2010 the Hub Transport server is configured with *Set-TransportServer* and the Mailbox server is configured with *Set-MailboxServer*. However, if both roles are installed on the same Exchange 2010 server, as they are in the examples above, there is only one message tracking configuration for the entire server, not one for each separate role. Both sets of cmdlets can be used to get or set the message tracking configuration for the entire Exchange 2010 server. In Exchange 2013 and Exchange 2016 the services previously associated with the Hub Transport server role now reside on the Mailbox server role. This means that *Get-TransportService/Set-TransportService* and *Get-MailboxServer/Set-MailboxServer* can be used in Exchange 2013 and will achieve the same outcomes.

In most environments the default message tracking configuration is likely to be modified to retain more log data, to allow historical searches to go back further in time. For example, you might choose to increase the message tracking log maximum age to 90 days, and increase the maximum directory size to 4 GB. Remember, that would equate to 12 GB of potential log data. If you need to analyse your servers for consistency in their message tracking log configurations, you can use the Get-MessageTrackingConfig.ps1 script.

The message tracking log files themselves are simply text files in comma-separated value (CSV) format (Figure 4-18). The log files can be read in applications such as Notepad or Excel if you have a need to view their contents. The data fields used for the CSV files include the date/time of the event, client details, server details, event IDs, sender and recipient information, and more. You can read a complete list of the message tracking log fields on TechNet.

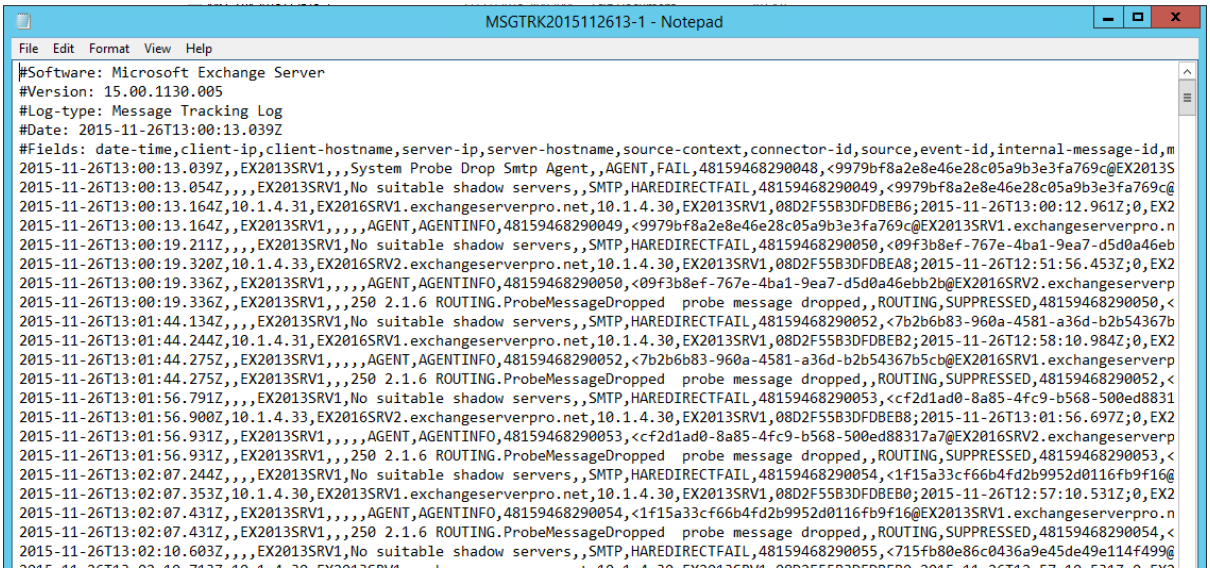


Figure 4-18: Message Tracking Log content

Although the message tracking log files are human-readable, the best way to search message tracking logs is by running message tracking searches using PowerShell. Message tracking log searches are performed in the Exchange Management Shell by running the *Get-MessageTrackingLog* cmdlet. You can run that cmdlet with no parameters on any Transport or Mailbox server that is enabled for message tracking, and it will return the first 1000 results of the log entries on that server.

You can also search a remote server using the **-Server** parameter. This is useful when you are running the search from your own admin workstation or a separate management server, instead of while logged on directly to an Exchange server. In fact, there is no real reason why you need to run the searches directly on an Exchange server. Using the management tools remotely is far more convenient, as well as avoiding problems such as very large searches consuming a lot of memory and impacting server performance.

The *Get-MessageTrackingLog* cmdlet also accepts input from the pipeline. This is a very convenient way to perform searches on multiple servers at once. For example, to search all Mailbox servers at once:

```
[PS] C:\>Get-MailboxServer | Get-MessageTrackingLog
```

The default output for *Get-MessageTrackingLog* presents the information in a table with just a few properties shown. Displaying the output in a list instead, using the Format-List cmdlet, gives you more details to look at.

```
[PS] C:\>Get-MailboxServer | Get-MessageTrackingLog | Format-List
```

When you're performing investigative searches of your message tracking logs, particularly across multiple servers, those queries can take a long time to return the results. If you then found that you need to adjust the query, for example to be more specific, or to format the results in a different way, you have to wait a long time for the query to run a second time as well. For this reason, I recommend that you always collect your query results into a variable, particularly very broad queries that take a long time to run, so that you can pick apart the collected data without having to re-run the query.

For example, if we want to investigate reports of email problems sending to Alan Reid, we can run one broad query across all Hub Transport servers and collect the results in a variable I will call **$msgs**.

```
[PS] C:\>$msgs = Get-MailboxServer | Get-MessageTrackingLog –Recipients
"Alan.Reid@exchangeserverpro.net" –Resultsize Unlimited
```

The **$msgs** variable now contains thousands of message tracking log entries that can be dissected in different ways without re-running that first, time-consuming query. For example, to find the top 10 senders to Alan Reid, we can simply pipe the **$msgs** variable into further PowerShell cmdlets and it will return a result within seconds instead of potentially several minutes.

```
[PS] C:\>$msgs | Group-Object -Property Sender | Select-Object name,count | sort count -desc | select -
first 10

Name                                Count
----                                -----
Andrea.Sharma@exchangeserverpro.net   110
Richard.Bennett@exchangeserverpro.net  108
Judy.Mollo@exchangeserverpro.net      104
Ferzana.King@exchangeserverpro.net    102
Debra.Lowe@exchangeserverpro.net      100
Nicola.Noad@exchangeserverpro.net     100
Diane.Hall@exchangeserverpro.net       96
Caroline.Ball@exchangeserverpro.net    96
Chris.Majumdar@exchangeserverpro.net   96
Hugh.Sharma@exchangeserverpro.net      96
```

The information shown in the example above is interesting, but doesn't help us to actually track an email message through the transport pipeline. To track an email message, you will need to search on message characteristics such as the sender, recipient, or the time that it was sent. Although you can filter the log data that you've captured in a variable by piping it to the *Where-Object* cmdlet (which I'll demonstrate later in this section), it is generally better to run the most precise query you can, based on the information that you have about the message you are tracking, and then filter a smaller set of results as needed.

To filter message tracking log searches by time or date ranges you can use the *–Start* and *–End* parameters for *Get-MessageTrackingLog*.

```
[PS] C:\> Get-MessageTrackingLog –Start 7/21/2015 –End 7/25/2015 –ResultSize Unlimited
```

If you provide a start date for the search but do not provide an end date, the search will run from the start date to the most recent available log entries.

Although these parameters accept standard date/time formatted values as shown in the example above, it is often simpler to pair their usage with the *Get-Date* cmdlet to search on relative times instead.

```
[PS] C:\> Get-MessageTrackingLog –Start (Get-Date).AddHours(-24) –ResultSize Unlimited
```

You can also search message tracking logs based on the sender and recipients of the email message.

```
[PS] C:\> Get-MessageTrackingLog –Sender alan.reid@exchangeserverpro.net

[PS] C:\> Get-MessageTrackingLog –Recipients dawn.evans@exchangeserverpro.net
```

Unfortunately, sender or recipient-based search criteria do not accept wildcards or partial matches. This means that if you want to search for all email messages sent from Gmail addresses to a particular user, then you would need to run a query for all messages send to the user, then filter those results with *Where-Object*.

```
[PS] C:\> $msgs = Get-MessageTrackingLog –Start (Get-Date).AddDays(-3) –Recipients
alan.reid@exchangeserverpro.net –Resultsize Unlimited

[PS] C:\> $msgs | Where {$_.Sender –like *@gmail.com}
```

Notice also in the example above that I've combined a date-based search with a recipient-based search, and used the *–Resultsize* parameter to ensure that I receive all results and not just the first 1000. You can combine multiple search parameters to make your message tracking log searches very precise.

Unlike sender and recipient-based searches, you can search for email messages based on subject lines using the *–MessageSubject* parameter and get partial-matches in the results.

```
[PS] C:\> Get-MessageTrackingLog –MessageSubject "payroll"
```

Despite all the examples so far you might still be wondering how you can search for all message tracking log entries for one specific message. That can be achieved by searching with the *MessageID* parameter. The challenge is in finding the unique message ID first. Let's look at an example. First, let's retrieve all message tracking log entries for a recipient in the last 7 days, from all transport servers in the organization.

```
[PS] C:\>$msgs = Get-TransportService | Get-MessageTrackingLog -Start (Get-Date).AddDays(-7) -
Recipients Alan.Reid@exchangeserverpro.net -ResultSize Unlimited
```

Next, let's look at some information, including the unique message ID, of each entry in those results.

```
[PS] C:\>$msgs | Select timestamp,sender,messagesubject,messageid
```

The output of that command should give you an idea of which message you want to zero in on, and you'll be able to see the unique message ID for that message. If the message ID is getting cut off in the output of that command, pipe it to *Format-List* to see the full value.

After you've determined the unique message ID you can run a new search using that attribute.

```
[PS] C:\>$msgs2 = Get-TransportService | Get-MessageTrackingLog -MessageId
"b9e841ad96924f49b0c3669628a3f836@EX2013EDGE.exchangeserverpro.net" -Start (Get-Date).AddDays(-1) -
ResultSize Unlimited
```

Now we can look at just those events captured in the **$msgs2** variable. You should always sort output by timestamp to ensure that you're looking at the message tracking log events in the order in which they occurred.

```
[PS] C:\>$msgs2 | Sort Timestamp

EventId  Source   Sender                          Recipients                      MessageSubject
-------  ------   ------                          ----------                      --------------
HARED... SMTP     info@relais-promotions.com      {alan.reid@exchangeserverpro.net} Evaluez votre
Quotient Digital
RECEIVE  SMTP     info@relais-promotions.com      {alan.reid@exchangeserverpro.net} Evaluez votre
Quotient Digital
AGENT... AGENT    info@relais-promotions.com      {alan.reid@exchangeserverpro.net} Evaluez votre
Quotient Digital
SEND     SMTP     info@relais-promotions.com      {alan.reid@exchangeserverpro.net} Evaluez votre
Quotient Digital
```

```
DELIVER  STORE... info@relais-promotions.com       {alan.reid@exchangeserverpro.net} Evaluez votre
Quotient Digital
```

If you want to see all of the information for each log entry, pipe it to *Format-List*.

```
[PS] C:\>$msgs2 | Sort Timestamp | Format-List
```

For more information on the event ID and source values for message tracking logs to help you interpret the results of your searches, you can refer to TechNet.

Transport Queues

Exchange servers that host Transport services queue messages for delivery. This is useful for situations when the destination server is unavailable for some reason. Rather than immediately drop or reject a message that can't be delivered, the server will hold the message in its queue and retry at regular intervals.

This means that you will often encounter troubleshooting scenarios where you find messages sitting in transport queues. This may be due to a variety of problems. Consider the factors involved in email delivery that were discussed earlier in this chapter, such as network connectivity, DNS, and firewalls. If any of those elements are experiencing a fault, misconfiguration, or interruption, then you can expect to see queued emails.

The transport queue on an Exchange server can be viewed using the *Get-Queue* cmdlet.

```
[PS] C:\>Get-Queue

Identity                  DeliveryType Status MessageCount
--------                  ------------ ------ ------------
EX2013EDGE\3              SmartHost... Retry  10
EX2013EDGE\333            DnsConnec... Retry  1
EX2013EDGE\334            DnsConnec... Retry  1
EX2013EDGE\335            DnsConnec... Retry  1
EX2013EDGE\336            DnsConnec... Retry  1
EX2013EDGE\337            DnsConnec... Retry  1
EX2013EDGE\338            DnsConnec... Retry  2
EX2013EDGE\Submission     Undefined    Ready  0
```

In the example above you can see that one queue has 10 messages in it. We can take a closer look at that queue by running *Get-Queue* and specifying the queue name.

```
[PS] C:\>Get-Queue EX2013EDGE\3 | Fl

DeliveryType                    : SmartHostConnectorDelivery
NextHopDomain                   : ex2013srv1.exchangeserverpro.net,ex2010srv1.exchangeserverpro.
                                  net,ex2016srv1.exchangeserverpro.net,ex2016srv2.exchangeserver
                                  pro.net
TlsDomain                       :
NextHopConnector                : 623aa60a-0727-4464-ac24-f5a47ac8488d
Status                          : Retry
MessageCount                    : 10
LastError                       : 451 4.4.0 DNS query failed. The error was: DNS query failed with
                                  error ErrorRetry
RetryCount                      : 33
LastRetryTime                   : 12/27/2015 10:10:25 PM
NextRetryTime                   : 12/27/2015 10:41:26 PM
FirstRetryTime                  : 12/27/2015 9:03:04 AM
DeferredMessageCount            : 0
LockedMessageCount              : 0
MessageCountsPerPriority        : {0, 0, 0, 0}
```

```
DeferredMessageCountsPerPriority : {0, 10, 0, 0}
```

The problem is clearly seen; DNS queries for the next hop domains are failing, and the queue has retried 33 times since 12/27/2015 at 9:03:04 am. Could the issue have been caused by a configuration change at around that time that impacted DNS queries from the server? That would be the logical question to answer. We already looked at troubleshooting tips for DNS earlier in this chapter, but using *nslookup* to test the DNS client on the server is just one part of it.

On an Exchange server there are multiple places that DNS can be configured:

- On the network interfaces for the Windows Server operating system
- On the Exchange server properties (Figure 4-19)
- On send connectors

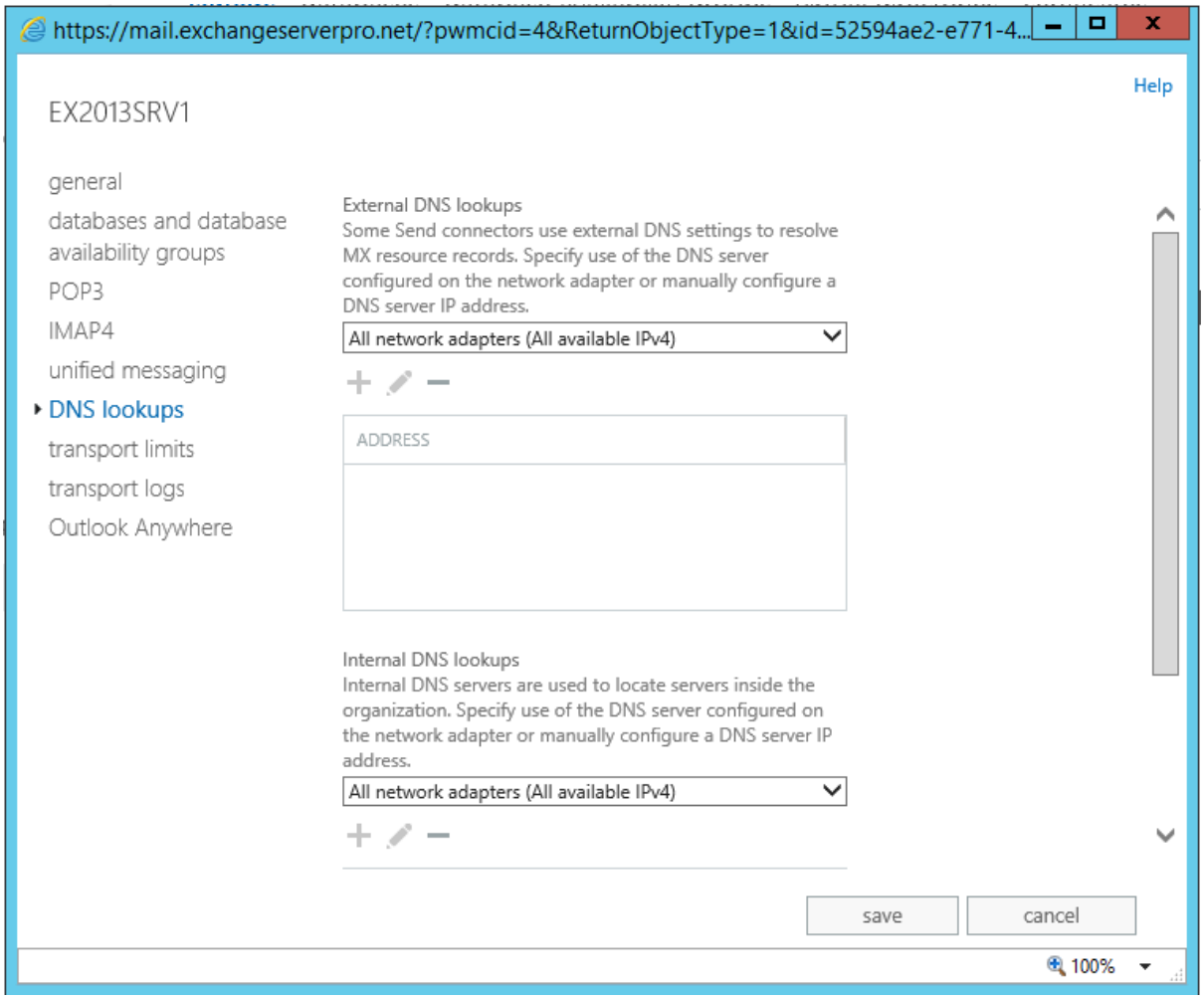


Figure 4-19: Configuring DNS for a server

When you perform an *nslookup* to test DNS without explicitly specifying a DNS server IP to test against, you're only testing the DNS servers configured on the network interfaces for the operating system. In some environments those DNS servers may be different from the DNS servers that the Exchange Server application users. Exchange servers will use all available network adapters for DNS lookups by default, or if necessary you can specify other DNS servers for Exchange to use in the properties of the server.

In addition to the per-server DNS settings, each send connector can also be optionally configured to use specific DNS servers. The setting shown in Figure 4-20 forces a send connector to use the external DNS

lookups configuration on the Exchange server, instead of the Windows Server operating system configuration. In most situations you won't need to modify those settings, however it is important to be aware of their existence. In some cases, you might try adding specific DNS server IP addresses to the Exchange server to see if that resolves a suspected DNS issue.

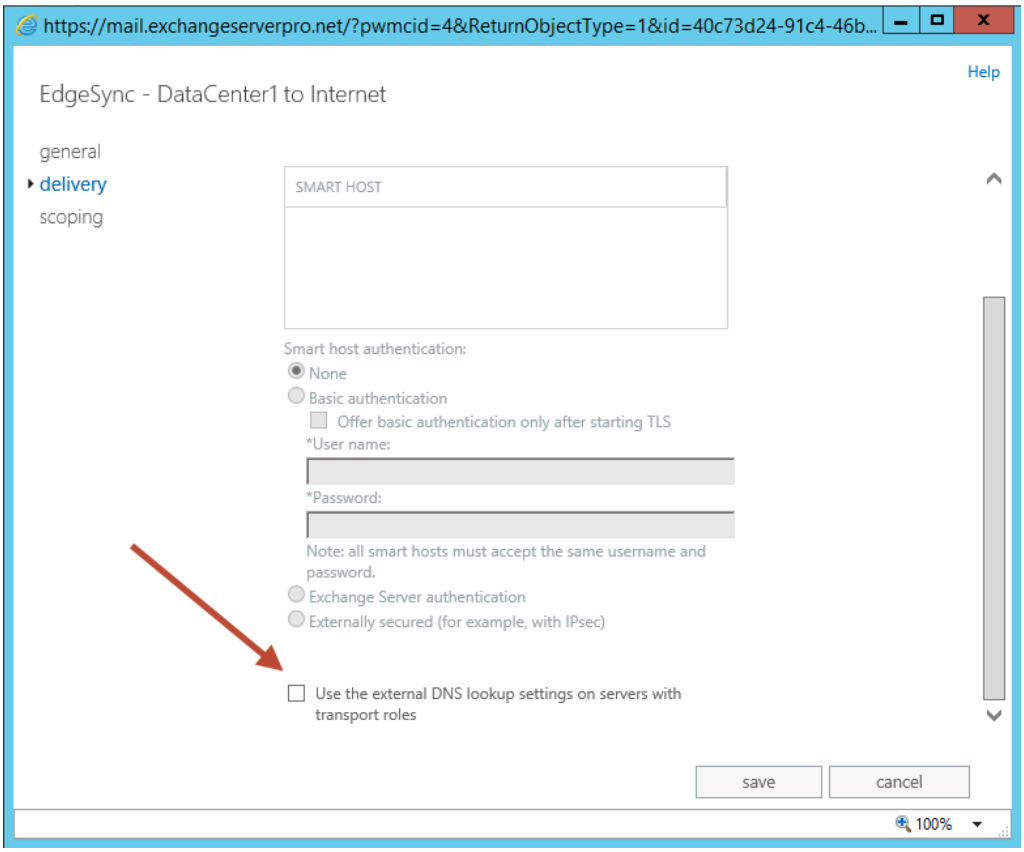


Figure 4-20: Forcing a send connector to use a specific DNS lookup

Sometimes you may notice queued messages to a destination that you are unsure about. If the next hop domain for a queue is one you don't recognize, then you can take a closer look at the messages in that queue by piping *Get-Queue* into *Get-Message*.

```
[PS] C:\>Get-Queue EX2013EDGE\3 | Get-Message

Identity               FromAddress            Status
--------               -----------            ------
EX2013EDGE\3\4199189... drozdov.091@mail.ru    Ready
EX2013EDGE\3\4200048... chefbob1@box1157.blu... Ready
EX2013EDGE\3\4200048... icantbr1@server.ican... Ready
EX2013EDGE\3\4200907... support@ojco.se        Ready
EX2013EDGE\3\4200907... vGPSYcdnavNUsh@hotma... Ready
EX2013EDGE\3\4202196... <>                     Ready
EX2013EDGE\3\4202196... zakaz@domveriko.ru     Ready
EX2013EDGE\3\4202196... zakaz@domveriko.ru     Ready
EX2013EDGE\3\4202625... zakaz@domveriko.ru     Ready
EX2013EDGE\3\4202625... zakaz@domveriko.ru     Ready
```

In the example above the email messages look like spam. It is not unusual to see queues for messages (usually non-delivery reports) to next hop domains that are simply spam domains. If excessive quantities of spam messages in transport queues are causing you a disk space or server performance problem, you can remove messages from the queue.

```
[PS] C:\>Get-Queue EX2013EDGE\3 | Get-Message | Remove-Message

Confirm
Are you sure you want to perform this action?
Removing the message "EX2013EDGE\3\41991895252993".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is "Y"): a
```

However, even if you do not manually remove the messages they will eventually expire from the queue when the retry intervals and maximum age limit for the server have been exceeded. You can see the retry intervals and maximum age for a Transport service by running the *Get-TransportService* cmdlet.

```
[PS] C:\>Get-TransportService | Select MessageRetryInterval,MessageExpirationTimeout

MessageRetryInterval     : 00:15:00
MessageExpirationTimeout : 2.00:00:00
```

If you encounter a problem with an Exchange server that is causing emails to queue, but you notice that another Exchange server in the same organization is able to send and receive messages without any problems, then you can consider redirecting the queued messages from the faulty server to the working server. This is normally performed as part of server maintenance, but can be used in a troubleshooting scenario as well.

There are two steps for the process of redirecting messages. The first step places the server component "HubTransport" into a state of "Draining" so that it is considered to be an unavailable server for new messages. The second is to redirect the messages in the queue. Note that when running *Redirect-Message* you need to use the fully-qualified domain name of the target server.

```
[PS] C:\>Set-ServerComponentState -Identity EX2013SRV1 -Component HubTransport -State Draining -
Requester Maintenance

[PS] C:\>Redirect-Message -Server EX2013SRV1 -Target EX2016SRV1.exchangeserverpro.net

Confirm
Are you sure you want to perform this action?
Redirecting messages to "EX2016SRV1.exchangeserverpro.net".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [?] Help (default is "Y"): y
```

What Else Can Cause Transport Issues?

So far in this chapter I've covered a lot of factors that can cause transport issues in your Exchange environment. When you look at transport issues as being any potential email delivery problem, then there are several more factors to consider.

Remote Domains

Remote domains are used in an Exchange organization to specify settings for mail flow between your Exchange organization and external domains. Remote domains can control which message formats are used (e.g. HTML, plain text), as well as whether automatic replies and non-delivery reports are permitted. This may result in messages being modified, or even rejected, due to remote domains configurations. Keep that in mind when troubleshooting mail flow issues from your organization to external domains.

Recipient Configuration

The configuration of individual recipients can impact mail flow. Recipients are covered in chapter 6 of this book.

Back Pressure

Back pressure describes a resource exhaustion condition for Exchange servers that leads to a server actively refusing some or all connection or email delivery attempts. The overloaded state is based on a series of resource utilization metrics:

- Free disk space on the drive(s) that store the transport queue database and logs
- Uncommitted database transactions in server memory
- Memory utilization by the Transport service process
- Overall memory utilization for the server

Each of those metrics is individually measured, and each can cause the server to go into a back pressure state. The server will be in one of three states:

- Normal – all is well and the server is performing its role as intended.
- Medium – a resource is moderately over-utilized, and the server begins limiting some connection types.
- High – a resource is severely over-utilized. The server ceases to accept any new connections.

The most obvious indication of a back pressure condition is the following response to an SMTP connection attempt: *452 4.3.1 Insufficient system resources*.

The other signs of a back pressure condition are the following entries in the Application event log:

- Event ID 15004 – an increase in the utilization level for any resource (e.g. from Normal to Medium)
- Event ID 15005 – a decrease in the utilization level for any resource (e.g. from Medium to Normal)
- Event ID 15006 – high utilization for disk space (i.e. critically low free disk space)
- Event ID 15007 – high utilization for memory

Back pressure is ultimately a symptom of server performance issues. Server performance troubleshooting is covered in more detail in chapter 8 of this book.

Message Size Restrictions

An Exchange environment can have message size restrictions applied at many different locations:

- The organization level
- On send connectors
- On receive connectors
- On Active Directory site links
- On individual recipients

For ease of management and troubleshooting it is generally recommended to configure all of the message size limits to the same value. However, if you have a need to set different limits, then the largest limit should be applied at the organization level, and the smallest limit on individual recipients.

When email messages are rejected due to size limits, the non-delivery report will state that quite clearly. You can then inspect your message size limits, or consider the limits that an external recipient may have in place, to determine which one caused the email to be rejected.

Transport Rules

Transport rules can apply a variety of actions to email messages in transit, including redirecting email messages or rejecting them entirely. A transport rule can even silently drop an email message, with no notification to the sender or the intended recipient.

Transport rule actions are usually identifiable in message tracking log searches, which was covered earlier in this chapter.

Block lists

Anti-spam block lists can cause your email to external recipients to be rejected. You can monitor for your domain being added to block lists by using services such as MXToolbox (Figure 4-21). Or you can run manual queries if you suspect that your emails may be getting rejected due to being listed on a block list.

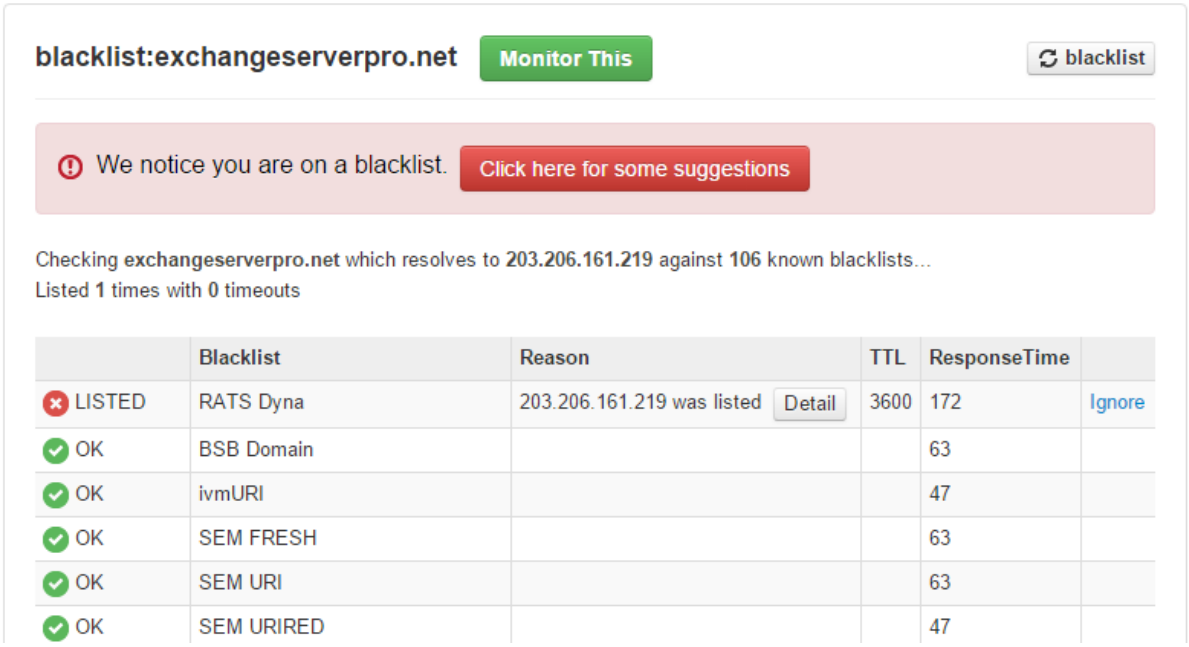


Figure 4-21: Checking block lists

Not all email servers use a block list, and some don't use publicly available block lists that would show up on a search such as that shown above. Instead they use private lists, or aggregated lists that combine several block list sources. In any case, if you are being blocked due to a block list then you will usually receive a non-delivery report that makes that clear. Your only action when you are included on a block list is to address whatever reason caused you to be listed, and then apply for de-listing.

Real World: Sending email via a smart host usually means you are sharing that smart host with other customers, some of which may engage in undesirable email behaviour from time to time. If your smart host is blocklisted due to the actions of another customer, then you will likely notice your own emails getting rejected as well. This is one of the perils of using a third party for your email delivery, and you should always have a backup plan (such as another smart host, or using direct delivery) in case such an incident occurs.

Additional reading

History of Transport in Exchange Server

- Exchange 2016 Architecture
- Exchange 2013 Server Role Architecture
- Overview of Exchange 2010 Server Roles

The Critical Role of DNS

- What is an MX Record, and How Do They Work?
- Reverse DNS Lookup
- Sender Policy Framework Introduction
- RFC2821 - Simple Mail Transfer Protocol
- Sender ID Framework SPF Record Wizard
- MXToolbox SPF Record Validator
- Common Errors in SPF Records

SMTP Connectivity

- Get-PublicIPAddress.ps1 PowerShell Script
- Network ports for clients and mail flow in Exchange 2013
- You Cannot Send or Receive E-Mail Messages Behind a Cisco PIX Firewall
- Configuring Unique Receive Connector SMTP Banners in Exchange Server

The Transport Pipeline

- Mail Flow and the Transport Pipeline
- Receive Connectors in Exchange Server 2013/2016
- Send Connectors in Exchange Server 2013/2016
- How to Configure Exchange Server 2016 for SMTP Application Relay

Troubleshooting Transport

- Protocol Logging in Exchange Server 2013/2016
- Message Tracking in Exchange Server 2013/2016
- Transport Queues in Exchange Server 2013/2016
- Understanding Back Pressure

Chapter 5: Troubleshooting Mailbox Servers

Paul Cunningham

The Mailbox server role occupies the center of Exchange and refers to the services that host mailbox databases and provide high availability for mailbox databases in a Database Availability Group (DAG). The Mailbox server role also performs a number of other Client Access and Transport tasks that are outside of the scope of this chapter (see Chapters 3 and 4). In addition, the internal workings of mailbox databases such as how transaction logging works, and the use of ESEUTIL for recovery scenarios, are covered in detail in chapter 9.

The Information Store

The Information Store is at the heart of the Mailbox server. The Information Store has evolved significantly through the different releases of Exchange, and in Exchange 2013 changes were made to improve the scalability of the Information Store service by changing from a single store.exe process to a controller/worker model in which multiple, isolated instances of different process run in unison.

- The *Microsoft Exchange Replication* service (MSExchangeRepl) runs as the *MSExchangeRepl.exe* process, and is responsible for telling the Information Store process to mount and dismount databases, detect database failures, and initiate recovery action for any database failures.
- The *Microsoft Exchange Information Store* service (MSExchangeIS) runs the controller process Microsoft.Exchange.Store.Service.exe, managing each of the worker process lifecycles by performing the mount and dismount operations as instructed by the Replication service, or by terminating worker processes if there is a database failover event in a database availability group.
- The *Microsoft.Exchange.Store.Service.exe* processes do not exist as Windows services. Each worker process is isolated to performing RPC operations for a single database when the database is mounted, and providing the database cache for that database. If a database is dismounted, the worker process is terminated. If a worker process fails, the Information Store can start another worker process for that database.

That relationship between the Replication service, Store service, and the worker processes, exists on both standalone Mailbox servers and Mailbox servers that are members of a DAG. However, for a standalone Mailbox server there are limits as to what recovery actions can be performed on a failed database, since there are no other database copies to failover to.

Information Store Service Dependencies

Both the Replication service and the Store service are dependent on the *Microsoft Exchange Active Directory Topology* service (MSExchangeADTopology), which provides information about Active Directory to other Exchange services. If MSExchangeADTopology can't start, or fails due to an error, then the Information Store will go offline as well. You can read more about the MSExchangeADTopology service and Exchange's dependencies on Active Directory in Chapter 2.

Information Store Event Logging

The Replication and Store services will log events to the Application event log when they are unable to start, or if they are able to start but unable to mount databases (Figure 5-1). During any troubleshooting scenario in which services fail to start, or databases fail to mount, the Application event log is the first place to look.

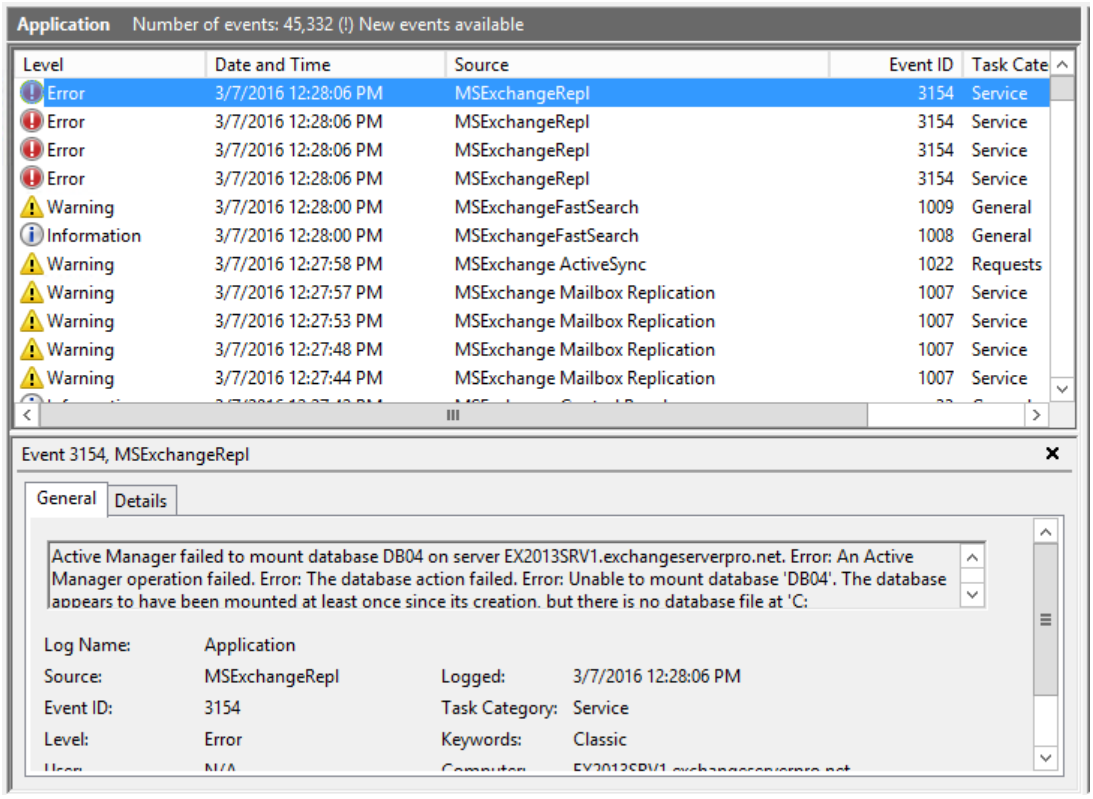


Figure 5-1: Application event log entries logged by the Replication service

Troubleshooting Service Start-up

The *Test-ServiceHealth* cmdlet can be used to identify services that have not started. This cmdlet has existed since Exchange 2007, and still uses the server role references from the Exchange 2007/2010 architecture. Even so, it will tell you when Exchange services are not running.

```
[PS] C:\> Test-ServiceHealth -Server EX2013SRV1

Role                    : Mailbox Server Role
RequiredServicesRunning : False
ServicesRunning         : {IISAdmin, MSExchangeADTopology, MSExchangeDelivery,
                          MSExchangeMailboxAssistants, MSExchangeRepl, MSExchangeRPC,
                          MSExchangeServiceHost, MSExchangeSubmission, MSExchangeThrottling,
                          MSExchangeTransportLogSearch, W3Svc, WinRM}
ServicesNotRunning      : {MSExchangeIS}
```

For cases where a service has simply crashed or been stopped inadvertently, a *Start-Service* command will start it again.

```
[PS] C:\> Invoke-Command -ComputerName EX2013SRV1 {Start-Service MSExchangeIS}
```

The most common causes of an Information Store service stopping, or failing to start when you issue a *Start-Service* command, are:

- Service dependency issues. Another service that the Information Store depends on, such as MSExchangeADTopology, can't or won't start.

- Service corruption issues. A software corruption has occurred, requiring the Exchange server application to be repaired or reinstalled.

In any case, the Application event log is a good place to start looking for clues as to why the services won't start. Even when Information Store services do start, the databases themselves may not automatically mount.

Database Mounting

Each mailbox database has an attribute, *MountAtStartup*, that determines whether the database should automatically mount when the Store services starts.

```
[PS] C:\> Get-MailboxDatabase -Server EX2013SRV1 -Status | Select Name,MountAtStartup,Mounted

Name MountAtStartup Mounted
---- -------------- -------
DB01           True    True
DB02           True    True
DB03           True    True
DB04           False   True
```

The Information Store will also check the last mount state of the database at start up. If the database was in a dismounted state because an administrator had dismounted it, and the Information Store service is restarted, for example during an operating system restart, then it will not attempt to mount the database on the next start up. In the example below, the database DB03 is configured to mount at start up, but was left dismounted.

```
[PS] C:\> Get-MailboxDatabase -Server EX2013SRV1 -Status | Select Name,MountAtStartup,Mounted

Name MountAtStartup Mounted
---- -------------- -------
DB01           True   True
DB02           True   True
DB03           True   False
DB04          False   False
```

The rationale is that the Exchange administrator had deliberately dismounted the database for a good reason, and so it should be left dismounted. This decision is logged in the Application event log as an error (Figure 5-2), so you will be able to accurately determine the root cause of a database not mounted in those conditions.

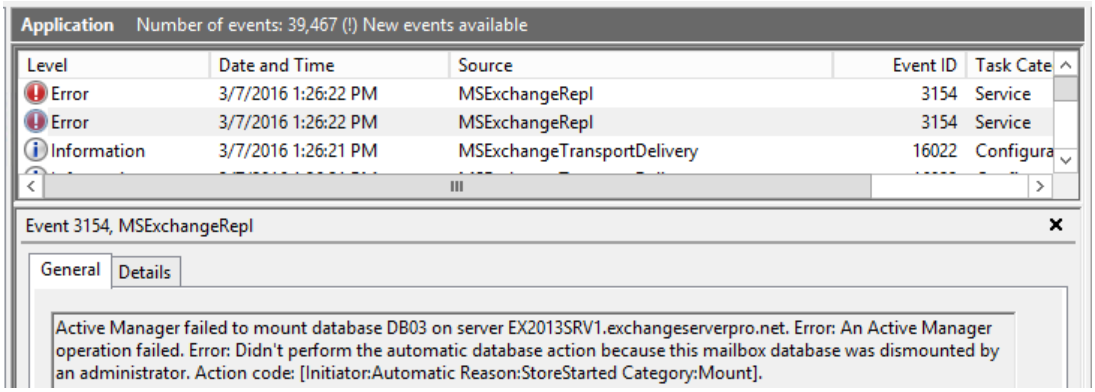


Figure 5-2: The Replication service logs its decision to the Application event log

If a database is configured to not mount at start up, or if the Replication service determines that it should not be mounted, then it can still be manually mounted by running the *Mount-Database* cmdlet.

```
[PS] C:\> Mount-Database DB03
```

For databases that will not mount there are some common root causes:

- Active Directory replication. A newly created database will not mount if the new database object has not replicated to all domain controllers in the Active Directory site. The solution is to simply wait for replication to complete before trying again.
- Storage failures. If the storage path of the database is inaccessible due to permissions problems, or due to a failure of the storage itself, then the database can't be mounted.
- Database corruption. If the database EDB file is damaged or corrupt, and can't be automatically recovered by the Exchange server, then the database will not mount. Recovery from database corruption scenarios is discussed in more detail in chapter 9.

In all cases, the Application event log is where you should look first for a clear indication of why the database won't mount.

Rapid Growth in Database and Transaction Log Size

You might notice an unexpected increase in the size of mailbox databases or the number of the transaction log files associated with a database. Transaction logging is described in more detail in chapter 9, but to summarize here, for every change that occurs within a mailbox database there is transaction log data written as well. Under normal, healthy circumstances, full or incremental backups will truncate the transaction logs, and ample free disk space will exist on the storage volumes that host transaction logs to allow the server to continue working.

However, if the transaction log files unexpectedly grow in size before a backup is complete, the log drive may run out of free disk space, which will cause the database to dismount. Extending the storage volume to allow the database to be mounted, and then running a backup to truncate the logs, is a common resolution to this problem scenario. Enabling circular logging is also often considered, although it carries some risks, because all of the transaction logs will be truncated by the server regardless of whether you've backed up the database. With circular logging enabled, restoring a database from backup only gets you a point in time recovery from the time of the backup, as there are no more log files to "roll forward" to bring the restored database further up to date. As a means of restoring service when you run out of logging disk space, circular logging has its usefulness. But don't leave it enabled without considering the downsides.

The question is then what caused the unexpected growth in transaction logs? Some causes are quite obvious, for example, a large batch of mailbox migrations will generate a significant amount of transaction logging on the destination mailbox database. Other causes are less obvious, such as rogue users or applications. Historically, there have been some famous incidents in which a buggy version of Apple's iOS operating system generated excessive transaction logs due to a calendar sync issue. Such problems will occur from time to time, and are difficult to track down when they do.

Microsoft provides guidance (17 steps in fact) in a useful blog post on the topic. The guidance is written primarily for Exchange 2007 and 2010, but still generally applies to Exchange 2013 and 2016. Just be sure to use the compatible version of tools such as Exmon.

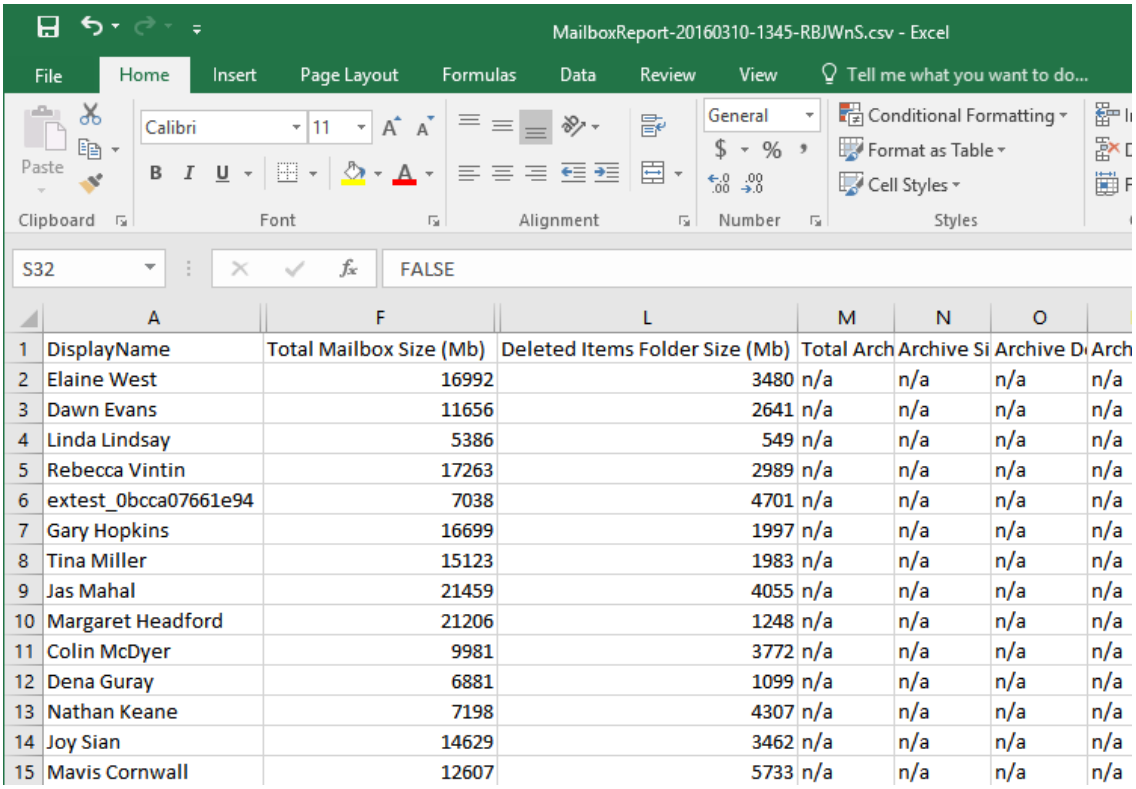
Rapid growth of the mailbox database file itself presents a similar issue to the log files. If the database volume runs out of free disk space, then the database will dismount. Again, extending the volume to allow enough free disk space to mount the database is the usual solution, and then some remedial action can be taken to reduce the growth rate of the database file.

Ultimately, rapid database growth is a problem of "too much data in, not enough data out". If users are receiving new email at a rate that exceeds the deletion of unwanted email, then the database will grow. This is a fact of life that most Exchange administrators simply need to deal with. Users don't like deleting email, and even when they do it's common to find users who never empty their deleted items folder.

Reviewing the statistics for mailboxes on a database allows you to identity problems, such as:

- Unusually large mailbox users, who might be good candidates for archiving, or for moving to a different database.
- Mailboxes with large amounts of deleted items, which you can consider purging. A Group Policy to automatically empty deleted items when Outlook closes is an effective way to do this, though it is often unpopular with users who expect to be able to find things in their deleted items folder.

Running a PowerShell script such as Get-MailboxReport.ps1 will show you the mailbox statistics, including total mailbox size, and deleted item size (Figure 5-3).



	A	F	L	M	N	O	
1	DisplayName	Total Mailbox Size (Mb)	Deleted Items Folder Size (Mb)	Total Arch	Archive Si	Archive D	Arch
2	Elaine West	16992	3480	n/a	n/a	n/a	n/a
3	Dawn Evans	11656	2641	n/a	n/a	n/a	n/a
4	Linda Lindsay	5386	549	n/a	n/a	n/a	n/a
5	Rebecca Vintin	17263	2989	n/a	n/a	n/a	n/a
6	extest_0bcca07661e94	7038	4701	n/a	n/a	n/a	n/a
7	Gary Hopkins	16699	1997	n/a	n/a	n/a	n/a
8	Tina Miller	15123	1983	n/a	n/a	n/a	n/a
9	Jas Mahal	21459	4055	n/a	n/a	n/a	n/a
10	Margaret Headford	21206	1248	n/a	n/a	n/a	n/a
11	Colin McDyer	9981	3772	n/a	n/a	n/a	n/a
12	Dena Guray	6881	1099	n/a	n/a	n/a	n/a
13	Nathan Keane	7198	4307	n/a	n/a	n/a	n/a
14	Joy Sian	14629	3462	n/a	n/a	n/a	n/a
15	Mavis Cornwall	12607	5733	n/a	n/a	n/a	n/a

Figure 5-3: Analyze mailbox statistics to determine the cause of database growth

Of course, a mailbox size report is merely a snapshot in time, and won't tell you which mailboxes are growing the fastest. For that type of analysis, you will need to run multiple reports over a period of time and compare the numbers, or invest in a monitoring system that tracks user mailbox growth trends.

Mailbox databases won't shrink even if you remove data from them, but they will reclaim the unused database pages as available new mailbox space (which we often refer to as "whitespace" even though that term carries a different meaning to the Exchange product team), which will be used for any new data written to the database without immediately increasing the overall EDB file size. However, if you do wish to reduce the EDB file size so that the used disk space can be recovered, there are two options available to you:

- An offline defrag. This is generally not recommended because it requires an outage while the defrag operation is performed, which can take a long time for very large database files.
- Mailbox migrations. By moving all of the mailboxes in the database to a new database, the new database file will grow only to the size needed to host the data in those mailboxes. None of available new mailbox space is migrated, so the new file will be smaller (Figure 5-4). The old database can then be removed to reclaim that disk space.

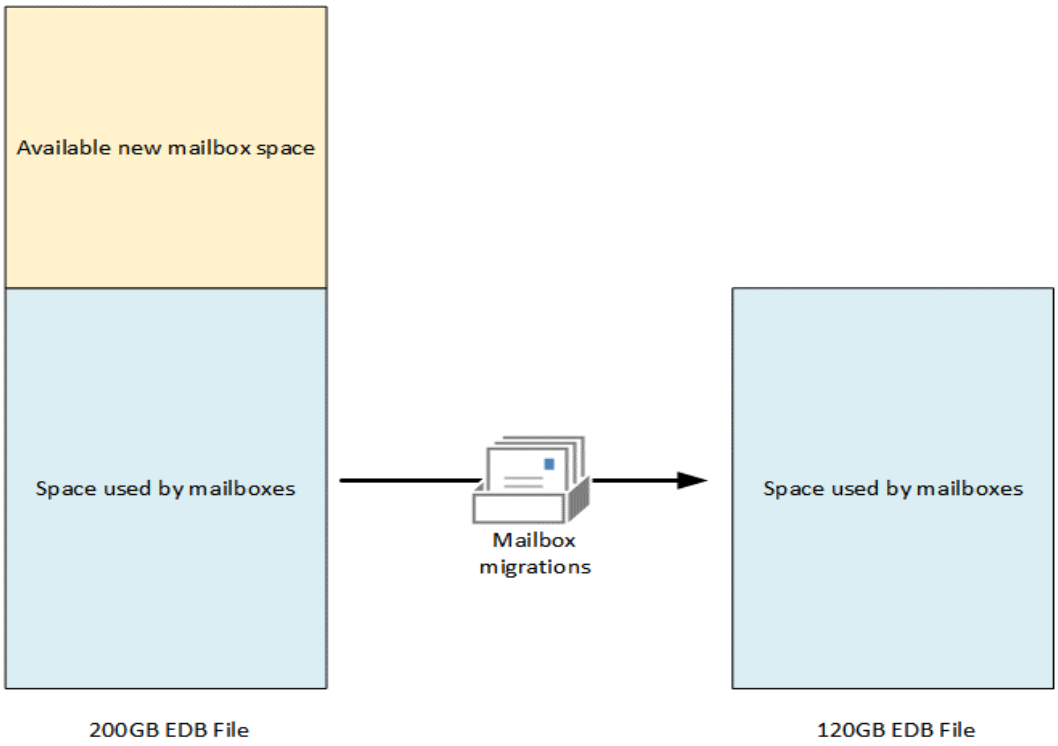


Figure 5-4: Using mailbox migrations to shrink a database file

Content Indexing

Exchange Search provides the capability for end users to quickly search their mailbox contents and locate items. In addition, Exchange Search enables in-place eDiscovery to occur by allowing authorized users (such as auditors) to search for content across multiple mailboxes in the organization. The search capability relies on the content index (also sometimes referred to as the catalog), which is built by the Microsoft Search Foundation content indexing engine. The Search Foundation is able to handle most of the common file formats that are found in email attachments these days, including Microsoft Office files, PDFs, HTML, and plain text. A complete list of file formats can be seen by running the *Get-SearchDocumentFormat* cmdlet.

For performance and efficiency, indexing of email messages and attachments occurs both in the transport pipeline, and in mailbox databases. One content index is maintained per mailbox database. The health of the content index for each mailbox database can be seen by running the *Get-MailboxDatabaseCopyStatus* cmdlet, including for single-copy databases that are not hosted by a DAG member.

```
[PS] C:\> Get-MailboxDatabaseCopyStatus -Server EX2013SRV1 | ft -auto

Name            Status     CopyQueueLength ReplayQueueLength LastInspectedLogTime ContentIndexState
----            ------     --------------- ----------------- -------------------- ----------------
DB01\EX2013SRV1 Mounted    0               0                                      FailedAndSuspended
DB02\EX2013SRV1 Mounted    0               0                                      Healthy
DB03\EX2013SRV1 Mounted    0               0                                      Healthy
DB04\EX2013SRV1 Dismounted 0               0                                      Failed
Test\EX2013SRV1 Mounted    0               0                                      Healthy
```

A content index that is not listed as "Healthy" may not actually be unhealthy, because the state of "Failed" is also used for content indexes of dismounted databases.

```
[PS] C:\> Get-MailboxDatabaseCopyStatus -Server EX2013SRV1 | Where {$_.ContentIndexState -ne "Healthy"}
| Select Name,ContentIndexState,ContentIndexErrorMessage
```

```
Name                      : DB01\EX2013SRV1
ContentIndexState         : FailedAndSuspended
ContentIndexErrorMessage : The content index is corrupted.

Name                      : DB04\EX2013SRV1
ContentIndexState         : Failed
ContentIndexErrorMessage : The database has been dismounted.
```

Similarly, if the database has indexing disabled, which is common for databases that are hosting a journal mailbox, then the content index state for the database will be listed as "Disabled".

Content index failures can be caused by:

- Conflicts with file-system anti-virus protection. If you run an anti-virus product on your Exchange servers, ensure you are complying with Microsoft's recommendations for file-level anti-virus exclusions.
- Very high volume of changes within a database. Journal mailboxes can cause this, as can shared mailboxes that are used by a large number of simultaneous users.

Impact of Unhealthy Content Indexes

Aside from the state of the content index that is returned by *Get-MailboxDatabaseCopyStatus*, the most obvious sign of a content index problem is that end users are unable to search their mailbox to locate messages and other items. However, this will depend on the client they are using to connect to the mailbox.

Outlook running in cached mode is included by default in the indexing that occurs in the Windows operating system, referred to as Windows desktop search. Since this indexing does not rely on the content index built by Search Foundation, a cached mode Outlook user can continue to search their mailbox even when there is a server-side index problem. On the other hand, Outlook on the web (OWA) users are entirely dependent on the server-side content index to be able to perform searches. So are users performing eDiscovery searches. Therefore, a good test to determine whether a content index issue may be occurring is to perform searches using cached mode Outlook and Outlook on the web, and then compare the results.

Real World: The comparison between cached mode Outlook search, and Outlook on the web search, is still useful even when the server-side content index is reporting as "Healthy", but you suspect an indexing problem may be impacting your end users.

In addition to end user searches, unhealthy content indexes can cause mailbox migrations to fail if the target database for the mailbox move has an unhealthy content index. Troubleshooting mailbox migrations is discussed in more detail in chapter 11. Content index health is also a factor that is taken into consideration during database failovers in a DAG, which is discussed later in this chapter.

Testing Exchange Search

Although you can manually test the content index for the database that hosts your own mailbox, a manual test is not necessarily going to tell you the full story. You might, for example, search for a message that has already been indexed. Your search could be successful, while at the same time new email messages are not being indexed. Furthermore, you can't test other databases that your mailbox is not hosted on.

The *Test-ExchangeSearch* cmdlet can be used to initiate a test of a mailbox database by generating a test message to a health mailbox on the database, and then running a search to determine whether the new message can be found, as well as to report how long it took for the new item to be located.

```
[PS] C:\>Test-ExchangeSearch -MailboxDatabase DB02
```

```
Database     Server       Mailbox      ResultFound SearchTime Error
                                                  InSeconds
--------     ------       -------      ----------- ---------- -----
DB02         EX2013SRV1   HealthMai... True        63.829
```

Rebuilding Content Indexes

When the content index for a database has become corrupt, it will need to be rebuilt, or reseeded from another database copy in the DAG. For now, let's look at the process for a non-DAG Mailbox server, and demonstrate the different procedure for DAGs later in this chapter.

This process involves removing the existing content index files, which will trigger Exchange Search to re-index that database. The re-indexing process can cause a high load on the Exchange server, which may impact performance for the server. So you should carefully consider the timing of any content index rebuilds, and how it might impact your end users. The content index files are located in the same path as the database EDB file, in a sub-folder named with a GUID (Figure 5-5).

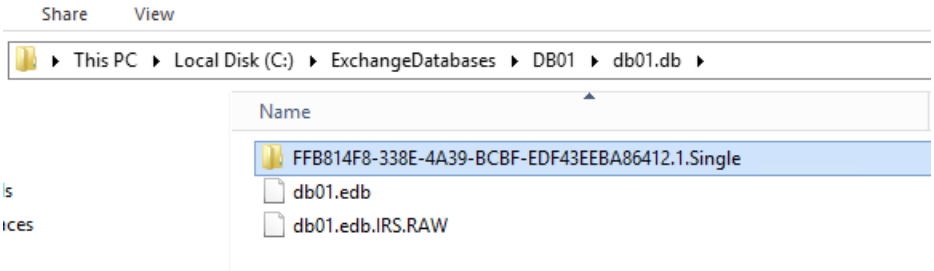


Figure 5-5: The content index folder

Before the corrupt index files can be removed, the Exchange Search services must be stopped. While these services are stopped, searches in OWA will not be able to be performed by end users, and all of the database content indexes on the server will be reported as "Failed" by *Get-MailboxDatabaseCopyStatus*.

```
PS C:\> Invoke-Command -ComputerName EX2013SRV1 {Stop-Service MSExchangeFastSearch; Stop-Service
HostControllerService}
```

Next, delete the GUID-named folder that contains the content index files. If the folder will not delete due to files in use, then it's likely that either:

- You haven't stopped the correct search services
- Another process, such as file-level anti-virus software, has a lock on the folder (and may be the cause of the index corrupting to begin with)

After deleting the files, start the search services again.

```
PS C:\> Invoke-Command -ComputerName EX2013SRV1 {Start-Service MSExchangeFastSearch; Start-Service
HostControllerService}
```

After a delay while Exchange Search evaluates the situation, the database will be re-indexed. The content index will have a state of "Crawling" while this is occurring.

```
[PS] C:\>Get-MailboxDatabaseCopyStatus -Server EX2013SRV1 | ft -auto

Name            Status  CopyQueueLength ReplayQueueLength LastInspectedLogTime ContentIndexState
----            ------  --------------- ----------------- -------------------- ----------------
DB01\EX2013SRV1 Mounted 0               0                                      Crawling
DB02\EX2013SRV1 Mounted 0               0                                      Healthy
DB03\EX2013SRV1 Mounted 0               0                                      Healthy
DB04\EX2013SRV1 Mounted 0               0                                      Healthy
```

You can monitor the progress of the database crawl by watching the **MSExchange Search Indexes\Crawler: Mailboxes Remaining** counter in Performance Monitor for that database instance (Figure 5-6).

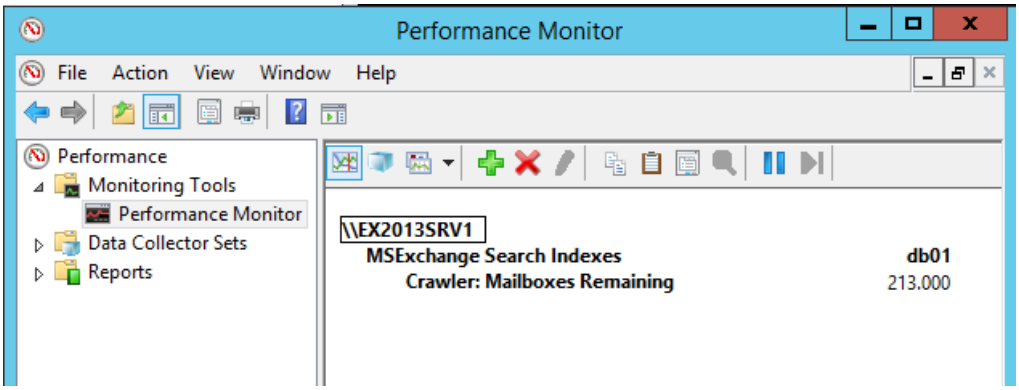


Figure 5-6: Monitoring the number of mailboxes remaining in a database re-index

Database Availability Groups

The high availability building block for Exchange is the database availability group (DAG). DAGs are highly resilient when designed and maintained correctly, and will behave in a predictable way when a failure occurs. On the other hand, a poorly designed and maintained DAG will behave in an unpredictable way, and can be complex to troubleshoot if they are not well understood.

Database Availability Group Concepts

Before we look at troubleshooting DAGs, let's cover some terminology first to clear up some of the often-misunderstood concepts.

- DAG members – the Mailbox servers that are members of the DAG and can host database copies within the DAG. A DAG can have up to 16 members and a File Share Witness.
- Quorum – the process by which the DAG members determine whether a majority of members are online and available. If quorum is lost, the DAG and all of the databases that it hosts may go offline.
- File Share Witness – a non-DAG member that is involved in the quorum voting process to act as a tie-breaker when the DAG has an even number of members.
- Active database copies – the active copy of a database is the copy on one of the DAG members that is mounted and actively servicing clients. There can be only one active copy of a database at any given time.
- Passive database copies – the passive copies of a database are dismounted and receive updates from the active database copy through continuous replication. A database can have no passive copies if it is a single-copy database, or up to 15 passive copies (due to the maximum number of DAG members being 16).
- Continuous replication – the process of replicating changes from the active database copy to the passive database copies.
- Copy queue – the queue of changes on the active database copy that are yet to replicate to a DAG member hosting a passive copy.
- Replay queue – the queue of changes to the active database copy that have replicated to a DAG member hosting a passive copy, but have not been replayed or committed to the passive database copy yet.
- Lagged database copies – lagged copies are passive database copies that have a delay of up to 14 days configured for either the replay or truncation of transaction logs, in continuous replication. By

"lagging" a database copy behind the others, the lagged copy can be used to recover from some logical corruption or data loss scenarios.
- Active Manager – a component of Exchange that is responsible for monitoring the status of the DAG, detecting failures, and making decisions about corrective actions that should be taken.
- Switchover – an administrator-initiated change in which the active database copy is dismounted, and a passive copy is mounted to become the active copy. Switchovers can be targeted (the administrator chooses the database copy that will become the active copy) or targetless (the administrator allows Active Manager to choose the copy that will become the active copy).
- Failover – similar to a switchover, however a failover is not administrator-initiated and occurs in response to an unplanned failure.
- Activation Preference – a value configured by the administrator on each copy of a database to indicate the most preferred database copy to be the active copy.

Managed Availability

Managed Availability is the built-in, intelligent monitoring and recovery system for Exchange 2013 and later. The goal of Managed Availability is to detect and resolve problems without requiring administrator intervention, and before they cause a negative user experience. Simply put, Managed Availability is constantly probing and testing the health and performance of your Exchange servers, and will take corrective action when it detects a problem. The corrective action could be as simple as recycling an IIS application pool, or even failing over a database. For more serious problems, Managed Availability can even "bug check" (force a reboot) of an entire server.

Managed Availability runs as two services on the Exchange server:

- Exchange Health Manager Service (MSExchangeHMHost.exe), which controls worker processes.
- Exchange Health Manager Worker process (MSExchangeHMWorker.exe), which is the worker process that performs each task within Managed Availability.

There are three main parts to Managed Availability:

- Monitors – these define what data to collect about an Exchange service or feature, what is considered "Healthy", and what actions should be taken to restore a feature to a healthy state. The data that is collected by monitors includes direct notifications from components, results from probes, and performance counters.
- Probes – these are how monitors obtain information about the health and performance of Exchange components so that the user experience for end users can be measured. Probes include synthetic transactions such as sending an email to the health mailbox on a database, or testing server-to-server connectivity over different protocols. Some probes are run by services other than the Exchange Health Manager Service, monitoring themselves and reporting results to Managed Availability.
- Responders – these take corrective actions for problems that have been identified by probes and monitors, such as restarting a service.

There are more than a thousand probes in Managed Availability; 1,192 for an Exchange 2013 CU10 multi-role server, and 1,474 for an Exchange 2016 RTM Mailbox server. You can expect those numbers to change over time as new builds of Exchange are released. You can see the full list by running the *Get-ServerHealth* cmdlet. A summary of the health of all of the health sets on the server can be retrieved by running the *Get-HealthReport* cmdlet.

```
[PS] C:\> Get-HealthReport -Server EX2016SRV1

Server             State HealthSet           AlertValue MonitorCount
------             ----- ---------           ---------- ------------
```

```
EX2016SRV1 NotApplicable ActiveSync            Healthy            2
EX2016SRV1 NotApplicable ActiveSync.Protocol   Healthy            7
EX2016SRV1        Online ActiveSync.Proxy      Healthy            1
EX2016SRV1 NotApplicable AD                    Healthy           28
EX2016SRV1 NotApplicable AM_Scheduled          Healthy           10
EX2016SRV1 NotApplicable AMADError             Healthy            2
EX2016SRV1 NotApplicable AMEUS                 Healthy            1
EX2016SRV1 NotApplicable AMFMSService          Healthy            6
EX2016SRV1 NotApplicable AMMessagesDeferred    Healthy            1
EX2016SRV1 NotApplicable AMScanError           Healthy           10
...
```

In a troubleshooting scenario you can use Managed Availability to narrow the focus of your investigation by querying a server health report for any alerts that are not "Healthy".

```
[PS] C:\> Get-HealthReport -Server EX2016SRV1 | Where {$_.AlertValue -ne "Healthy"}

Server     State         HealthSet    AlertValue LastTransitionTime   MonitorCount
------     -----         ---------    ---------- ------------------   ------------
EX2016SRV1 NotApplicable ECP          Unhealthy  3/14/2016 2:29:25 PM 11
EX2016SRV1 NotApplicable MailboxSpace Unhealthy  3/14/2016 1:23:07 PM 10
```

Health sets can be healthy, unhealthy, or one of four additional states:

- Degraded – the monitored item has been unhealthy for less than 60 seconds. After 60 seconds of not returning to a healthy state, it will be marked as unhealthy.
- Disabled – an administrator has disabled a monitor.
- Repairing – an administrator has marked a monitor or server as currently being repaired.
- Unavailable – the monitor is not responding to the Health service.

In the example above, the *MailboxSpace* health set is unhealthy. There are 10 monitors associated with that health set, any one of which could be the cause of the unhealthy state. We can run *Get-ServerHealth* for that health set to determine why the health set is not healthy.

```
[PS] C:\>Get-ServerHealth -Identity EX2016SRV1 -HealthSet MailboxSpace | Where {$_.AlertValue -ne
"Healthy"}

Server     State         Name                              TargetResource HealthSetName AlertValue
------     -----         ----                              -------------- ------------- ----------
EX2016SRV1 NotApplicable StorageLogicalDriveSpaceMonitor DB05           MailboxSpace  Unhealthy
```

Depending on the monitor or health set that is unhealthy, Managed Availability may place a server component into an inactive state. Inactive server components are effectively removed from the production Exchange environment. For example, a Hub Transport server component that is marked inactive will not participate in any mail flow within the organization, because other Exchange servers will not send email messages to it. The component states for a server can be viewed by running *Get-ServerComponentState*.

```
[PS] C:\>Get-ServerComponentState -Identity EX2016SRV1

Server                          Component                 State
------                          ---------                 -----
EX2016SRV1.exchangeserverpro.net ServerWideOffline        Active
EX2016SRV1.exchangeserverpro.net HubTransport             Active
EX2016SRV1.exchangeserverpro.net FrontendTransport        Active
EX2016SRV1.exchangeserverpro.net Monitoring               Active
EX2016SRV1.exchangeserverpro.net RecoveryActionsEnabled   Active
EX2016SRV1.exchangeserverpro.net AutoDiscoverProxy        Active
EX2016SRV1.exchangeserverpro.net ActiveSyncProxy          Active
EX2016SRV1.exchangeserverpro.net EcpProxy                 Active
EX2016SRV1.exchangeserverpro.net EwsProxy                 Active
EX2016SRV1.exchangeserverpro.net ImapProxy                Active
EX2016SRV1.exchangeserverpro.net OabProxy                 Active
EX2016SRV1.exchangeserverpro.net OwaProxy                 Active
EX2016SRV1.exchangeserverpro.net PopProxy                 Active
```

```
EX2016SRV1.exchangeserverpro.net PushNotificationsProxy     Active
EX2016SRV1.exchangeserverpro.net RpsProxy                   Active
EX2016SRV1.exchangeserverpro.net RwsProxy                   Active
EX2016SRV1.exchangeserverpro.net RpcProxy                   Active
EX2016SRV1.exchangeserverpro.net UMCallRouter               Active
EX2016SRV1.exchangeserverpro.net XropProxy                  Active
EX2016SRV1.exchangeserverpro.net HttpProxyAvailabilityGroup Active
EX2016SRV1.exchangeserverpro.net ForwardSyncDaemon          Inactive
EX2016SRV1.exchangeserverpro.net ProvisioningRps            Inactive
EX2016SRV1.exchangeserverpro.net MapiProxy                  Active
EX2016SRV1.exchangeserverpro.net EdgeTransport              Active
EX2016SRV1.exchangeserverpro.net HighAvailability           Active
EX2016SRV1.exchangeserverpro.net SharedCache                Active
EX2016SRV1.exchangeserverpro.net MailboxDeliveryProxy       Active
EX2016SRV1.exchangeserverpro.net RoutingUpdates             Active
EX2016SRV1.exchangeserverpro.net RestApiProxy               Active
```

Server component states are set by requesters. There are five available requesters:

- HealthAPI – this is used by Managed Availability when it marks an unhealthy component inactive. When the associated health sets are healthy again, Managed availability will set the component state back to "Active" again.
- Maintenance – this is used by administrators when they are performing maintenance on the server, such as when preparing a server for monthly Windows updates or a cumulative update installation.
- Functional – this is used by Exchange setup, for example during cumulative update installation.
- Sidelined, and Deployment – these are generally only used by Microsoft within Exchange Online (Office 365).

A server component will remain inactive if any requester has still applied an inactive state to it. For example, if you set the *ServerWideOffline* component to inactive in preparation for a cumulative update using the requester "Maintenance", Exchange setup will also then set the *ServerWideOffline* component inactive with the requester "Functional". At the end of Exchange setup, the *ServerWideOffline* component will be set back to active by the requester "Functional", but the component will remain inactive until the "Maintenance" requester is also used to set the component active again.

This is a good thing, because we wouldn't want Exchange setup to return the component to an active state before we've had a chance to do our own post-upgrade testing. However, it does create the potentially confusing scenario of an administrator trying to set a component back to an active state, and then finding that it remains inactive. In this situation it is useful to be able to check which requester still has the component set to an inactive state.

Server component states are stored in two locations:

- Active Directory – in the msExchComponentStates attribute of the Exchange Server object.
- Registry – in the HKLM\SOFTWARE\Microsoft\ExchangeServer\v15\ServerComponentStates registry key of the server.

We can query those location using the *Get-ServerComponentState* cmdlet.

```
[PS] C:\> $ComponentState = Get-ServerComponentState -Identity EX2016SRV1 -Component ServerWideOffline

[PS] C:\> $ComponentState.LocalStates

Requester      State Timestamp            Component
---------      ----- ---------            ---------
Functional    Active 3/10/2016 6:38:23 AM ServerWideOffline
Maintenance   Active 3/14/2016 5:24:13 AM ServerWideOffline
HealthApi   Inactive 3/14/2016 5:24:05 AM ServerWideOffline
Sidelined     Active 3/10/2016 6:38:09 AM ServerWideOffline

[PS] C:\>$ComponentState.RemoteStates
```

```
Requester      State Timestamp            Component
---------      ----- ---------            ---------
Maintenance   Active 3/14/2016 5:24:13 AM ServerWideOffline
HealthAPI   Inactive 3/14/2016 5:24:04 AM ServerWideOffline
Functional    Active 3/10/2016 6:38:23 AM ServerWideOffline
Sidelined     Active 3/10/2016 6:38:08 AM ServerWideOffline
```

In the example above, the HealthAPI requester (Managed Availability) has marked the component as inactive. This could be due to a server health issue detected after the server maintenance was performed, and should be investigated further.

Real World: Exchange setup uses the requester "Functional" to set the ServerWideOffline component as inactive very early in the setup process. If setup is interrupted, such as an administrator cancelling it before it has completed, the server component state will remain inactive for the requester "Functional" until setup is re-run successfully, or until an administrator manually sets the server component state active with that requester name.

Understanding the basic concepts of Managed Availability is important so that you aren't surprised when it takes corrective action in your environment. In the very early days of Exchange 2013, there were a few Managed Availability bugs, causing it to take corrective action for problems that did not really exist. In one case, Managed Availability restarted Exchange 2013 CU2 DAG members for Exchange environments deployed in multi-domain Active Directory forests, due to a bug with an Active Directory connectivity probe.

However, today Managed Availability has grown into a robust and reliable engine for Exchange high availability deployments. In fact, if you are seeing Managed Availability take action in your environment, the response should not be to try and stop or disable Managed Availability. Rather, you should be using Managed Availability to gain visibility into the health of your servers, and investigating the recurring problems with the Exchange server components themselves that are triggering Managed Availability responders. That is, unless another unfortunate bug rears its head in a future Exchange cumulative update.

Database Availability Group Networks

Each DAG has a minimum of one network that is used for client-facing network traffic, as well as for database replication traffic. Additional networks can be added to a DAG and used as dedicated replication networks, however this is generally not recommended these days due to the availability of high speed datacenter networking (10Gbps and higher) that can easily handle both client traffic and database replication traffic for most environments. The more DAG networks you have, the more complex the DAG will be to configure and maintain. Figure 5-7 is an example of a complex DAG network.

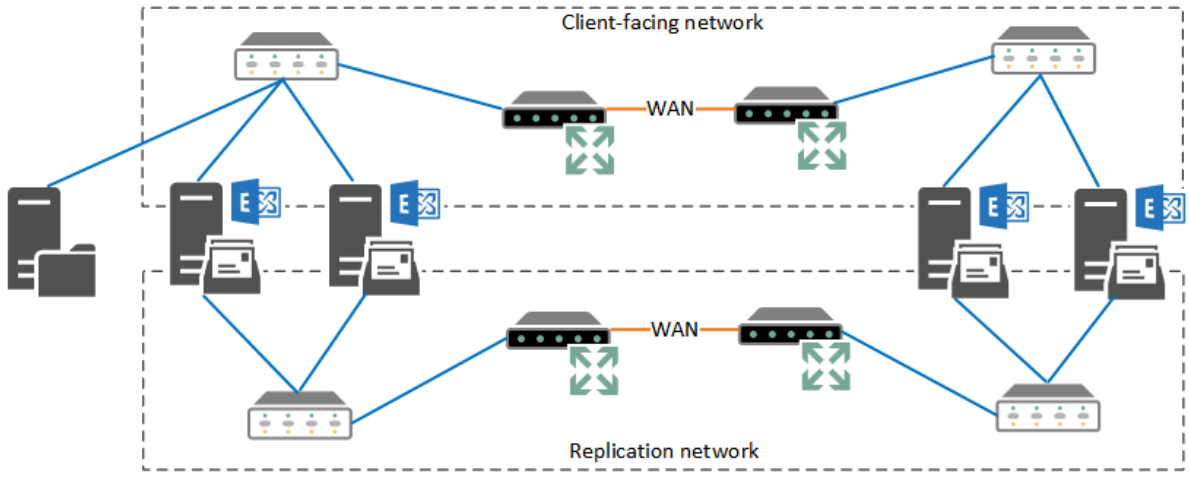


Figure 5-7: A complex, multi-site DAG network is rarely necessary

Complex networks can be easily misconfigured, which can cause database failovers to occur, as well as cause database switchovers to fail. DAG networks are critical to the stability of the DAG, not just to the performance of database replication.

Exchange will automatically configure a DAG network for any network interfaces that it detects in the operating system. DAG network auto-configuration relies on the following configurations on the network adapters.

The client-facing network interface on each DAG member should be configured with:

- A default gateway
- At least one DNS server
- The "Register this connection's addresses in DNS" option enabled

Any dedicated replication network interfaces should be configured with:

- No default gateway
- No DNS servers
- The "Register this connection's addresses in DNS" option disabled
- Static routes, if the network will span multiple IP subnets

If those conditions are not met, then DAG network auto-configuration often fails. The signs of failed network auto-configuration are misconfigured subnets appearing in the output of *Get-DatabaseAvailabilityGroupNetwork*.

```
[PS] C:\> Get-DatabaseAvailabilityGroupNetwork | select Name,Subnets,Interfaces

Name       : MapiDagNetwork
Subnets    : {{192.168.0.0/24,Misconfigured}, {192.168.1.0/24,Misconfigured},
             {10.1.100.0/24,Misconfigured}}
Interfaces : {{MELEX1,Up,192.168.1.101}, {MELEX2,Up,192.168.1.102},
             {SYDEX1,Up,192.168.0.101}, {SYDEX1,Up,10.1.100.2},
             {SYDEX2,Up,192.168.0.102}, {SYDEX2,Up,10.1.100.3}}

Name       : ReplicationDagNetwork01
Subnets    : {{10.1.101.0/24,Up}}
Subnets    : {{10.1.101.0/24,Up}}
Interfaces : {{MELEX1,Up,10.1.101.2}, {MELEX2,Up,10.1.101.3}}
```

Correcting the misconfigurations, and then enabling and disabling manual network configuration for the DAG will allow auto-configuration to make another attempt. Assuming your network interfaces are correctly configured on all DAG members following the guidelines above, auto-configuration should be successful.

```
[PS] C:\> Get-DatabaseAvailabilityGroup DAG1 –ManualDagNetworkConfiguration $true

[PS] C:\> Get-DatabaseAvailabilityGroup DAG1 –ManualDagNetworkConfiguration $false
```

Because the DAG attempts to auto-configure a network for every network interface that is available in the operating system, it can result in DAG networks for network interfaces that you do not want Exchange to use at all. Examples of this include dedicated backup networks, iSCSI storage networks, and out-of-band management ports. Unwanted network interfaces can be entirely excluded from DAG networking by running the *Set-DatabaseAvailabilityGroupNetwork* cmdlet with the *IgnoreNetwork* parameter. At the same time, it's also advisable to rename the DAG network from its auto-assigned name, so that the purposes of the network and the reason for excluding it from the DAG is clear.

```
[PS] C:\> Set-DatabaseAvailabilityGroupNetwork DAG1\ReplicationDagNetwork02 –Name “iSCI”           –
ReplicationEnabled:$false –IgnoreNetwork:$true
```

Database Copy Queues and Replay Queues

Each DAG member that is hosting a copy of a database in a DAG participates in the continuous replication process that keeps the database copies consistent and up to date.

Database replication occurs between DAG members using two methods:

- File mode replication
- Block mode replication

During file mode replication, each transaction log files generated by the active database copy is written until it is closed off when it reaches 1MB in size, and then copied to the DAG members hosting passive database copies. The DAG members hosting passive database copies then replay the transaction log files into their own copy of the database file to update it with the latest changes. File mode replication is used for the initial seeding of a database copy, and also any time a database copy needs to "catch up" with replication.

Block mode replication works in a similar way, except that as each database transaction is written to the log buffer on the DAG member hosting the active database copy, it is also copied to the log buffer of other DAG members hosting passive copies. Each DAG member is then able to build its own transaction log file from the data stored in the log buffer, instead of waiting for an entire 1MB log files to be shipped from the active database copy. Block mode has the advantage that it reduces the amount of potential data loss if there is a failure on the active database copy.

While replication occurs, the copy queue and reply queue for a database copy indicate the amount of transaction log data that not yet been copied or replayed for a passive database copy. You can view the copy and reply queues by running *Get-MailboxDatabaseCopyStatus*.

```
[PS] C:\> Get-MailboxDatabaseCopyStatus *

Name              Status  CopyQueueLength ReplayQueueLength
----              ------  --------------- -----------------
DB05\EX2016SRV1   Healthy 0               0
DB05\EX2016SRV2   Mounted 0               0
DB06\EX2016SRV1   Healthy 0               0
DB06\EX2016SRV2   Mounted 0               0
DB07\EX2016SRV1   Healthy 0               0
DB07\EX2016SRV2   Mounted 0               0
DB08\EX2016SRV1   Healthy 0               0
DB08\EX2016SRV2   Mounted 0               0
```

High copy queue lengths are an indication that transaction log data for changes are occurring on the active database copy that is not able to be copied to other DAG members fast enough, or not copied at all. This could be due to:

- A DAG member that is unreachable due to a server or network failure
- A high volume of change occurring on the database, such as a large batch of mailbox migrations
- Replication being suspended for a database copy by an administrator for maintenance or reseeding purposes
- A database copy that is resynchronizing, for example after a server restart, and has not yet resume replication
- A passive database copy that has failed and been suspended from replication by Exchange

Suspended replication by an administrator would be visible as a status of *Suspended* for the database copy. Similarly, a status of *Seeding* indicates that an administrator has initiated a reseed, and *Resynchronizing* indicates that the system is assessing the state of the database copy after a server restart or maintenance has

been completed. In all of those cases, a high copy queue or replay queue is normal and should resolve itself within a short period of time.

Real World: You might encounter a scenario in which the copy queue length for a database copy is 9223372036854775766 (9 quintillion). This is due to a self-preservation mechanism built in to the DAG for when the cluster registry replication between DAG members is out off by more than 12 minutes. A full explanation has been written up by Microsoft's Tim McMichael.

By design, lagged database copies are likely to have a high copy or replay queue length. Most lagged copies are implemented by configuring a lag period for the replay interval, as this allows the transaction log data to be replicated to the DAG member hosting the lagged copy, and then held there while the lag interval passes, before it is then replayed into the passive database copy. Lagged copies are not immediately obvious when you inspect the output of *Get-MailboxDatabaseCopyStatus*, because the *ReplayLagStatus* is not included in the default output.

```
[PS] C:\> Get-MailboxDatabaseCopyStatus * | Select -Property Name -ExpandProperty ReplayLagStatus |
sort name

Name            ConfiguredLagTime ActualLagTime    Percentage DisabledReason Enabled PlayDownReason
----            ----------------- -------------    ---------- -------------- ------- --------------
DB05\EX2016SRV1 00:00:00          00:00:00                  0                False           None
DB05\EX2016SRV2 00:00:00          00:00:00                  0                False           None
DB06\EX2016SRV1 00:00:00          00:00:00                  0                False           None
DB06\EX2016SRV2 00:00:00          00:00:00                  0                False           None
DB07\EX2016SRV1 00:00:00          00:00:00                  0                False           None
DB07\EX2016SRV2 00:00:00          00:00:00                  0                False           None
DB08\EX2016SRV1 7.00:00:00        06:22:47.6923062          0                 True           None
DB08\EX2016SRV2 00:00:00          00:00:00                  0                False           None
```

Database Failover Behaviour

When a failure occurs within the DAG that causes the active database copy to go offline, Active Manager makes a decision about which copy of a database should become the active copy. This decision making process is called Best Copy and Server Selection (BCSS). BCSS is an important concept to understand, because it will answer the question, "*Why did my database copies fail over to that server instead of that other server?*"

BCSS takes into account multiple variables to determine which database copy should be made active, with the goal of restoring availability of service for end users, while balancing that goal against the risk of data loss if a database copy that is not fully up to date was to be made active. Included in this decision are:

- Database and server configuration settings
- The health of the database copies within the DAG
- The health of the server components on DAG members

Excluded from consideration are any database copies that are:

- Unreachable, for example due to a network failure
- Configured by an administrator to be suspended from activation
- Hosted on a DAG member that is configured by an administrator to be blocked from automatic activation

You can view the configuration for the database copies in a DAG by running the *Get-MailboxDatabaseCopyStatus* cmdlet.

```
[PS] C:\> Get-MailboxDatabaseCopyStatus | Select Name,ActivationSuspended

Name            ActivationSuspended
```

```
----            ------------------
DB01\EX2013SRV1              False
DB02\EX2013SRV1              False
DB03\EX2013SRV1              False
DB04\EX2013SRV1              False
```

Activation is often suspended on database copies that have been configured as a lagged copy.

For server-wide automatic activation settings, there are two options. *DatabaseCopyAutoActivationPolicy* controls the types of automatic activation that mailbox database copies on the server can perform:

- Blocked – prevents the database copies on the server from automatically activating. Administrators can still choose to manually activate database copies on the server.
- IntraSiteOnly – prevents cross-site failovers from occurring by only allowing database copies to activate in the same Active Directory site. If the Active Directory site spans two physical datacenter locations, this setting can't prevent a failover to the other datacenter location, because Exchange is only aware of the Active Directory site boundary and not the physical datacenter boundaries.
- Unrestricted – this is the default setting and allows any database copies on the server to be considered for failover by BCSS, unless the specific database copy is suspended from activation.

You can view the automatic activation policy for Mailbox servers by running the *Get-MailboxServer* cmdlet.

```
[PS] C:\> Get-MailboxServer | Select Name,DatabaseCopy*

Name       DatabaseCopyAutoActivationPolicy DatabaseCopyActivationDisabledAndMoveNow
----       -------------------------------- ----------------------------------------
EX2013SRV1                     Unrestricted                                    False
EX2010SRV1                     Unrestricted                                    False
EX2016SRV1                     Unrestricted                                    False
EX2016SRV2                     Unrestricted                                    False
```

In the output above you can see an additional setting for Mailbox servers, that is named *DatabaseCopyActivationDisabledAndMoveNow*. This setting strikes a comfortable middle ground between the desire to keep active database copies running on specific servers, while not completely blocking a server from automatic activation if no other healthy servers are available. By configuring this setting to *$true*, database copies on the server will only be activated if no other healthy copies exist. Furthermore, if another healthy database copy becomes available on another server, the DAG will perform a switchover to the other healthy copy almost immediately.

After excluding any blocked or suspended database copies, a process called Attempt Copy Last Logs (ACLL) is run to try and copy any missing log files from the server that was hosting the active database copy. ACLL is useful in situations where the failure is not a full server failure and the log files are still accessible, and ensures that database copies are as up to date as possible during the BCSS process.

Exchange 2013 and 2016 also evaluate the health of the server components that are monitored by Managed Availability. Servers with more server components healthy are preferred to those with some unhealthy components. Furthermore, if Managed Availability has initiated the database failover due to a failed server component, BCSS must choose a server on which that same server component is healthy when activating another database copy. These checks prevent databases from failing over to servers that are in a worse state of health than the server previously hosting the active copy. Next, BCSS considers the *AutoDatabaseMountDial* setting on Mailbox servers. This setting configures the threshold for the number of transaction log files that can be missing for a database copy before it is considered un-mountable. The copy queue length is used to determine how many log files are missing for a database copy. Copy queue length is discussed earlier in this chapter.

AutoDatabaseMountDial has three possible settings:

- GoodAvailability – this is the default setting, and allows a database copy to be automatically mounted even if it has a copy queue length of up to 6.
- BestAvailability – allows a database copy to be automatically mounted even if it has a copy queue length of up to 12.
- Lossless – allows a database copy to be automatically mounted only if there are no missing log files.

The AutoDatabaseMountDial setting can be seen by running the *Get-MailboxServer* cmdlet.

```
[PS] C:\> Get-MailboxServer | Select Name,AutoDatabaseMountDial

Name       AutoDatabaseMountDial
----       ---------------------
EX2013SRV1      GoodAvailability
EX2010SRV1      GoodAvailability
EX2016SRV1      GoodAvailability
EX2016SRV2      GoodAvailability
```

The rationale for GoodAvailability and BestAvailability is that it is preferable to restore service, and then mitigate the data loss caused by the missing log files by requesting resubmission of messages from Safety Net, which caches a configurable amount of email messages that have already been delivered to mailboxes (2 days' worth, by default). You can verify that Safety Net is configured for your organization by running *Get-TransportConfig*.

```
[PS] C:\> Get-TransportConfig | Select SafetyNetHoldTime

SafetyNetHoldTime
-----------------
2.00:00:00
```

When *AutoDatabaseMountDial* is set to *Lossless* on any member of a DAG, database copies that are being considered by Active Manager as failover candidates will be sorted in ascending order of activation preference. However, when *AutoDatabaseMountDia*l is set to *GoodAvailability* or *BestAvailability* on all members of a DAG, database copies are sorted by their copy queue length in order of shortest to longest. Only when two database copies have the same copy queue length will activation preference be used as a tie-breaker. Once a decision is made, Active Manager attempts to restore service by issuing a mount command to the most preferred database copy based on all of the BCSS criteria.

The BCSS process is complex and highly intelligent, designed to make the best decision about where databases should fail over to when a problem occurs. This decision if often contrary to the activation preferences that the administrator has configured on database copies, leading to confusion and a sense that the DAG is behaving in an unpredictable way.

Some administrators are then tempted to try and control the DAG failover behaviour by suspending activation on some database copies, or blocking activation on some DAG members. In doing so, you are only reducing the resilience. Instead, when you are troubleshooting database failover events, look at them in the context of the BCSS process and try to understand why the DAG made the decision that it did. It may be that you can improve the chances of the failovers occurring in line with your preferences through better monitoring of the health of the DAG members and database copies.

Loss of Quorum and the File Share Witness

Quorum is the voting process that the cluster uses to determine whether the DAG should remain online or go offline. If the DAG goes offline all of the databases in the DAG are dismounted and are therefore inaccessible to end users, causing an outage.

There are two quorum models:

- Note majority
- Node and file share majority

Node majority quorum mode is used when the DAG has an odd number of members (Figure 5-8). The file share witness is not required for the quorum voting process, because the DAG members can determine a "majority" themselves. For example, if one DAG member fails, 2/3 DAG members are still online (a majority) and the DAG can remain online. If two DAG members fail, 1/3 DAG members are still online, which may result in quorum being lost and the DAG going offline.

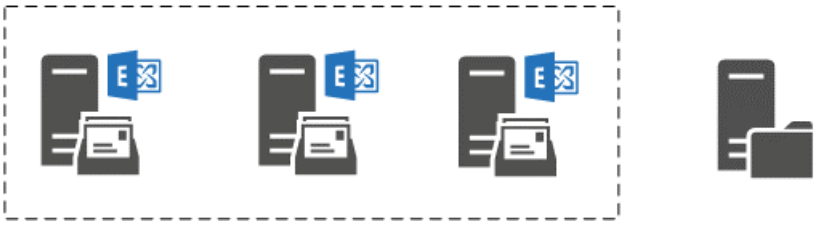


Figure 5-8: A three member DAG, with the file share witness not included in quorum voting

Node and file share majority is used when the DAG has an even number of members (Figure 5-9). The file share witness is included in the quorum voting process to ensure that a "majority" can be determined. For example, in a two-member DAG if one member fails, 1/2 members are still online (not a majority), but you would expect the DAG to be able to withstand a single node failure. The file share witness is used as the tie-breaker, meaning 2/3 "votes" are still available, and the DAG can stay online. Similarly, with a four-member DAG, if two members failed, with the file share witness there are still 3/5 "votes" online, so the DAG can stay online.

Figure 5-9: A four member DAG, with the file share witness included in quorum voting

All database availability groups are configured with a file share witness, whether it is used for voting or not. The quorum model is adjusted automatically by the DAG as you add or remove members.

Because a loss of quorum will cause the DAG to go offline, you should always plan maintenance tasks so that a majority of voting members will remain online during the maintenance. For example, for a three member DAG, perform updates and reboots on the first server, returning it to full operation before beginning your maintenance on the next server, and so on.

Note: A common misconception is that the file share witness must be online 100% of the time. This leads some administrators to try and build a resilient file share witness by using clustered file servers to host the FSW share. The FSW is not required to be online 100% of the time, it can be unavailable for short periods of time (for example, monthly Windows Update installation) just as the other DAG members can be, as long as a majority of quorum voting members remains online. Building a highly resilient FSW only adds complexity to your DAG.

In some circumstances the DAG can sustain a majority of nodes being offline if there have been multiple sequential failures. This is because a feature of Windows Server 2012 clusters called Dynamic Quorum (DQ). DQ makes it possible, but not guaranteed, that a cluster will survive sequential failures all the way down to a "last man standing" situation. DQ is enabled by default on Windows Server 2012 and above clusters, and

should be left enabled, but do not use it as justification to perform your server maintenance in a way that would otherwise cause quorum to be lost.

Some environments experience an unexpected loss of quorum when a single DAG member is taken offline, causing the entire DAG to go offline as well. This can be due to:

- A misconfigured DAG that is pointing to an FSW server name or share name that is inaccessible. This may be due to the FSW server being offline, the share removed, or the permissions on the shared folder being incorrect. The share must be accessible and allow Full Access by the *Exchange Trusted Subsystem* group.
- A cluster that has been manually set to the wrong quorum mode by an administrator.

The FSW path, and the quorum mode can both be viewed by running *Get-ClusterQuorum* on a DAG member.

```
PS C:\> Get-ClusterQuorum | fl

Cluster        : EX2016DAG01
QuorumResource : File Share Witness (\\mgmt.exchangeserverpro.net\EX2016DAG01.exchangeserverpro.net)
QuorumType     : NodeAndFileShareMajority
```

Multi-site DAGs can also be configured with two file share witnesses to facilitate the datacenter switchover process. The file share witness does not automatically fail over to the alternate server though, it is still part of the manual datacenter switchover process. The alternate witness server is not a mandatory property of the DAG though, so it's not unusual to see that no alternate witness server is configured on a DAG. You can use the output of *Get-DatabaseAvailabilityGroup* with the *–Status* switch to confirm the name of the witness server, the directory, and whether the primary or alternate witness share is in use.

```
[PS] C:\> Get-DatabaseAvailabilityGroup -Status | Select Name,*witness*


Name                       : EX2016DAG01
WitnessServer              : mgmt.exchangeserverpro.net
WitnessDirectory           : C:\DAGFileShareWitnesses\EX2016DAG01.exchangeserverpro.net
AlternateWitnessServer     :
AlternateWitnessDirectory :
WitnessShareInUse          : Primary
```

If the cluster management tools have been used to modify the DAG witness settings outside of Exchange, then the *WitnessShareInUse* will display a value of "InvalidConfiguration" instead. To correct this, run *Set-DatabaseAvailabilityGroup* to reset the file share witness settings for the DAG.

Multi-Site Behaviour

DAGs that span multiple datacenter locations are susceptible to failures of the WAN connection between the sites. Depending on the location of the file share witness server, a loss of WAN connectivity could cause either:

- Database failovers to the datacenter where the file share witness is located.
- The entire DAG going offline.

For example, in a two-datacenter deployment such as in Figure 5-10. if the WAN connection is lost then all databases that are active in "Datacenter 2" will failover and become active in "Datacenter 1", because that is where the file share witness server is located and therefore where quorum is able to be achieved.

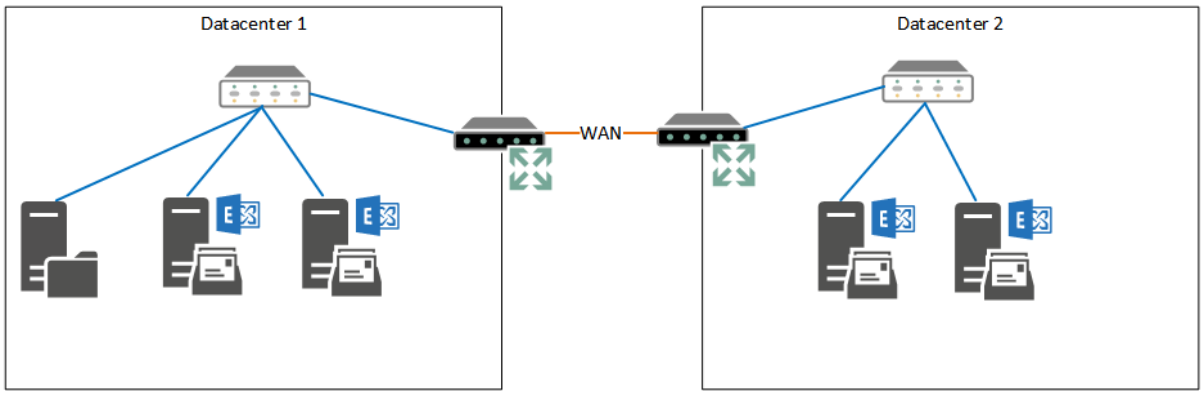


Figure 5-10: A multi-site database availability group

However, if the entire "Datacenter 1" location goes offline due to a power failure, the databases will not be able to become active in "Datacenter 2", because the DAG members in that site can't achieve quorum (only two out of five voting members will be available).

Note: This is normal behavior for multi-site DAGs that do not use a third site for the file share witness. If automatic site failover is required, then a third datacenter that is independently connected to each of the other datacenters should be used to host the file share witness. Azure can be used for this, if the organization doesn't have access to a third datacenter.

In datacenter outage scenarios where the DAG loses quorum, a manual datacenter switchover is required to restore service. It is recommended that Datacenter Activation Coordination mode (DAC mode) be enabled for all multi-site DAGs to prevent split brain scenarios from occurring after a datacenter switchover has been performed.

The DAC mode setting for a DAG can be viewed with the *Get-DatabaseAvailabilityGroup* cmdlet.

```
[PS] C:\> Get-DatabaseAvailabilityGroup | Select Name,DatacenterActivationMode

Name        DatacenterActivationMode
----        ------------------------
EX2016DAG01                  DagOnly
EX2010DAG                        Off
```

If you have not enabled DAC mode for a DAG, the datacenter switchover cmdlets will not be available, requiring the use of failover cluster management tools to bring the secondary site online.

Warning: DAC mode is designed to prevent a split brain condition from occurring if the primary datacenter comes back online before WAN connectivity is re-established. The DAG members in the primary site and the FSW will be able to achieve quorum, and the databases will be mounted in the primary datacenter while they are also already mounted in the secondary datacenter, causing divergence between the database copies that breaks replication. If you perform a datacenter switchover without DAC mode enabled, be careful not to bring the primary datacenter servers online without WAN connectivity being restored first.

Content Indexes for Replicated Databases

Earlier in this chapter we looked at content indexes for mailbox databases, and demonstrated how to repair a failed content index. In a database availability group, it is possible for the content index on one copy of a database to fail. In these cases, it is not necessary to completely rebuild the content index, because you can reseed it from a healthy copy of the content index instead. The content index state can be seen in the output of *Get-MailboxDatabaseCopyStatus*.

```
[PS] C:\> Get-MailboxDatabaseCopyStatus * | Sort Name

Name              Status  CopyQueueLength ReplayQueueLength LastInspectedLogTime ContentIndexState
----              ------  --------------- ----------------- -------------------- ----------------
DB01\EX2013SRV1   Mounted 0               0                                      Healthy
DB02\EX2013SRV1   Mounted 0               0                                      Healthy
DB03\EX2013SRV1   Mounted 0               0                                      Healthy
DB04\EX2013SRV1   Mounted 0               0                                      Healthy
DB05\EX2016SRV1   Healthy 0               0                 3/8/2016 1:20:49 PM  Suspended
DB05\EX2016SRV2   Mounted 0               0                                      Healthy
DB06\EX2016SRV1   Healthy 0               0                 3/8/2016 1:27:41 PM  Suspended
DB06\EX2016SRV2   Mounted 0               0                                      Healthy
DB07\EX2016SRV1   Healthy 0               0                 3/8/2016 1:27:41 PM  Suspended
DB07\EX2016SRV2   Mounted 0               0                                      Healthy
DB08\EX2016SRV1   Healthy 0               0                 3/8/2016 1:18:22 PM  Suspended
DB08\EX2016SRV2   Mounted 0               0                                      Healthy
```

In the example above, several content indexes on otherwise healthy passive database copies are in a suspended state, which usually indicates a failure of some kind. A reseed of the content index can be started by running *Update-MailboxDatabaseCopy* with the *–CatalogOnly* switch.

```
[PS] C:\>Update-MailboxDatabaseCopy DB05\EX2016SRV1 -CatalogOnly

Confirm
Are you sure you want to perform this action?
Seeding database copy "DB05\EX2016SRV1".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [?] Help (default is "Y"): y
```

However, when reseeding the content index on one database copy, or when all copies of the content index are failed, you can rebuild the index using the steps demonstrated earlier in this chapter.

Troubleshooting DAGs and Database Failovers

If you're not monitoring your database availability group, then it's possible that database failovers are happening without your knowledge. This is not entirely a bad thing, Exchange 2013 and 2016 have the intelligence of Managed Availability keeping a constant eye on the health of the servers, and taking corrective action when necessary. However, if you do want to see how many database failovers and switchovers have been occurring, you can use the *CollectOverMetrics.ps1* script that is included with Exchange to produce a report. To create the report, you simply run the script and provide the name of the DAG, and the date ranges that you want to report on.

```
[PS] C:\> cd $exscripts

[PS] C:\Program Files\Microsoft\Exchange Server\V15\scripts>.\CollectOverMetrics.ps1 -
DatabaseAvailabilityGroup EX2016DAG01 -StartTime (Get-Date).AddDays(-90) -EndTime (Get-Date)

Get statistics from EX2016SRV2.exchangeserverpro.net
Get statistics from EX2016SRV1.exchangeserverpro.net
Found total of 133 entries
Searching for ACLL loss reports on EX2016SRV2.exchangeserverpro.net.
Searching for ACLL loss reports on EX2016SRV1.exchangeserverpro.net.

Generated the following per-DAG reports:
C:\Program Files\Microsoft\Exchange
Server\V15\scripts\FailoverReport.EX2016DAG01.2016_03_10_15_48_13.csv
```

The default output is CSV, which can be read in Excel. If you prefer a HTML summary report, you can append the *–GenerateHtmlReport* switch to the command. However, opening the CSV in Excel where you can filter and sort the data will provide you with a lot more visibility than the HTML summary.

Exchange will also log events to several event logs on the server. The Application event log will contain basic information about events logged by Exchange services. Managed availability will log more detailed

information to the Applications and Services Logs, which is also known as the crimson channel logs (Figure 5-11).

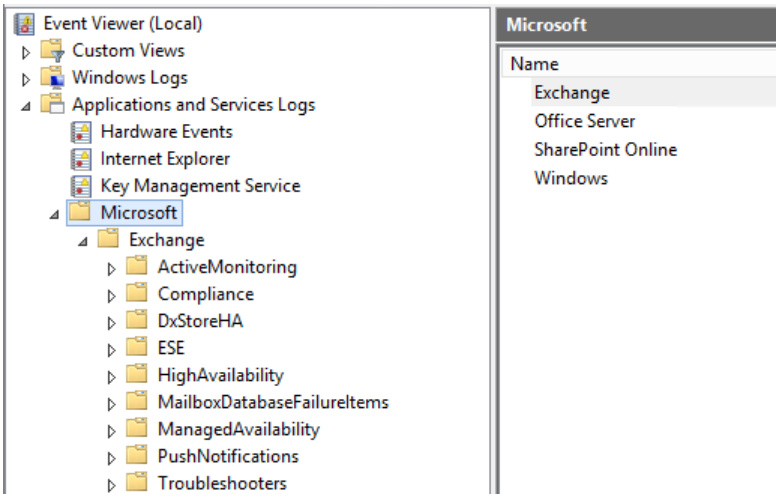


Figure 5-11: The crimson channel in Event Viewer

For database availability groups there are four crimson channels used:

- HighAvailability – for DAG events relating to the Replication service and Active Manager
- MailboxDatabaseFailureItems – for database copy-related events
- ActiveMonitoring – for Managed Availability events relating to probes, monitors and responders
- ManagedAvailability – also for Managed Availability events, relating to recovery actions taken by MA and their results

When you identify an incident has occurred, such as an outage for end users, or a database that was not able to failover to another DAG member, start by reviewing the Application event log to determine the time of the incident. Then, delve into the crimson channel to look for relevant events, and any historical trends that lead up to that event.

Because a DAG uses an underlying failover cluster, the cluster log itself can also be used to troubleshoot some problem scenarios. You may not find the cluster log useful for every application-level issue that happens in a DAG, such as a database failover by Managed Availability due to an unhealthy protocol. But if you experience problems with full server failures or loss of quorum, the cluster log can be useful.

To generate the cluster log run the Get-ClusterLog cmdlet on a DAG member. You can specify a destination for the log files, as well as a timespan (in minutes) that the logs should be collected for.

```
PS C:\> Get-ClusterLog -Destination C:\temp\clusterlogs -TimeSpan 3600

Mode                 LastWriteTime     Length Name
----                 -------------     ------ ----
-a---         3/14/2016  12:08 PM    5048572 EX2016SRV2.exchangeserverpro.net_cluster.log
-a---         3/14/2016  12:08 PM    4365402 EX2016SRV1.exchangeserverpro.net_cluster.log
```

Additional reading

Information Store

- Troubleshooting Rapid Growth in Databases and Transaction Log Files
- Exchange Server User Monitor (ExMon) for Exchange 2013 and 2016
- Get-MailboxReport.ps1 PowerShell Script

Content Indexes

- Running Windows Antivirus Software on Exchange 2016 Servers
- Antivirus Software in the Operating System on Exchange 2013 Servers

Database Availability Groups

- Managed Availability Monitors
- Managed Availability Probes
- Managed Availability Responders
- Server Component States in Exchange 2013
- What Did Managed Availability Just Do to This Service?
- Trauma for Exchange 2013 servers when Managed Availability goes bad
- Exchange 2013 Health Sets
- The Mystery of the 9223372036854775766 copy queue
- Datacenter Switchovers
- Dynamic Quorum

Chapter 6: Troubleshooting Recipients

Paul Cunningham

Exchange Server is installed into organizations to provide email and collaboration services for users. For all the technical features and complexity of an Exchange deployment, it is that simple function of letting people send and receive email, book meetings, and manage their daily tasks that Exchange provides. You'll often be called upon to troubleshoot those functions for the recipients in your organization, whether they are a person (a mailbox user), a resource (such as a meeting room), or a distribution group.

There are actually quote a lot of possible recipient types in an Exchange organization. Here's the full list:

- User mailbox – a mail-enabled user in Active Directory has a mailbox for messages, calendar, contacts, tasks and so on.
- Shared mailbox – a mailbox that is shared by multiple users, or that is not specifically associated with a person.
- Room mailbox – a mailbox used to manage calendar bookings for a meeting room or some other physical location.
- Equipment mailbox – a mailbox used to manage calendar bookings for a piece of equipment such as pool car or loan laptop.
- Site mailbox – a mailbox that is associated with a SharePoint site for document storage. Site mailboxes are not widely used.
- Office 365 mailbox – also known as a Remote Mailbox, refers to a user in a Hybrid deployment that has their mailbox in Exchange Online.
- System mailbox – special purpose mailboxes created and managed by the Exchange server itself for tasks such as eDiscovery and moderated transport.
- Linked mailbox – a mailbox in the organization that is associated with a user in a separate, trusted forest.
- Linked user – a user in the local forest that is associated with a mailbox in a separate, trusted forest.
- Mail user – a mail-enabled Active Directory user that has a mailbox hosted by an external system.
- Mail contact – an email recipient external to the organization who does not have a user account in the local Active Directory forest.
- Mail forest contact – a contact representing a recipient from another forest. These are usually created by Microsoft Identity Integration Server and are not directly managed with Exchange or Active Directory tools.
- Mail-enabled public folder – a public folder that can appear in the Global Address List and receive email messages.
- Distribution group – used to distribute messages to a group of recipients.
- Dynamic distribution group – a distribution group that uses an LDAP query instead of a static membership list to determine who to distribute messages to.
- Mail-enabled security group – an Active Directory security group that can be used to distribute messages as well as to grant or deny permissions to objects.

- Mail-enabled non-universal group – a group that existed in a legacy version of Exchange (2003 or earlier). These can appear in the Global Address List but attempts to send email to them will fail until the Active Directory group is converted to a Universal group.
- Microsoft Exchange recipient – used by Exchange to send system-generated messages to internal recipients, such as non-delivery reports and quota notifications.

That's quite a variety, but can be summarised as mailboxes, users, contacts, and groups. They're collectively referred to as recipients, and during this chapter I may refer to them in the general sense, such as "mailbox", or when necessary will use a specific name, such as "room mailbox".

The term "mail-enabled" is also important to understand. A mail-enabled object is an Active Directory object that has email attributes populated by Exchange. An example of this is contacts. Active Directory can host contact objects whether Exchange is installed in the forest or not. Contact objects in Active Directory can be used to store phone numbers, fax numbers, postal addresses, and so on. But it isn't until the contact object is mail-enabled that a mailbox user can see the contact in the Global Address List and send email messages to them.

Mailbox Access

The most common recipient you'll be administering is the mailbox; or more specifically, the mailbox user. When mailbox problems occur, users notice pretty fast, and you'll quickly hear about it. Let's start with a look at mailbox permissions.

Mailbox Permissions

In addition to the normal permission that a user has to access their own mailbox, there are also permissions required for the Exchange system itself to access mailboxes, as well as access by other users such as delegates or shared mailbox scenarios for teams. The permissions for Exchange access are discussed in chapter 7.

There are three ways for a user to be granted permissions to a mailbox:

- Mailbox permissions – generally used to grant access to an entire mailbox, such as a shared mailbox
- Mailbox folder permissions – used to grant permissions to specific folders, which includes mail item folders as well as other folders such as contacts, calendars, and tasks.
- Active Directory permissions – used to grant specific rights to a mailbox, such as the ability to send as the mailbox.

In this section we'll look at mailbox and mailbox folder permissions. Calendar permissions and send-as permissions are covered later in this chapter.

To view the mailbox permissions for a mailbox, we use the *Get-MailboxPermission* cmdlet.

```
[PS] C:\> Get-MailboxPermission -Identity alex.heyne
```

The output can be quite long for some environments, because it includes permissions set explicitly on that mailbox as well as all of the permissions that are being inherited from the mailbox database and other parent objects. To view only the non-inherited permissions, filter the output of *Get-MailboxPermission*.

```
[PS] C:\> Get-MailboxPermission -Identity alex.heyne | Where {$_.IsInherited -eq $false}

Identity             User                 AccessRights
```

```
--------            ----                 ------------
exchangeserverpro... NT AUTHORITY\SELF    {FullAccess, ReadPermission}
```

Most mailboxes will only have access by "NT AUTHORITY\SELF", which basically means the Active Directory user associated with the mailbox. To grant additional users access to the mailbox, use the *Add-MailboxPermission* cmdlet.

```
[PS] C:\> Add-MailboxPermission -Identity alex.heyne -User alan.reid -AccessRights FullAccess
```

The example above will grant Alan Reid full access to Alex Heyne's mailbox, including all folders and items. Alan can read, modify, or delete anything he likes. In this situation, enabling mailbox audit logging would be advisable, which is covered in chapter 12. In addition to granting Alan access to the mailbox, the default behaviour is also to auto-map the mailbox in Alan's Outlook profile the next time Autodiscover runs (typically within a few hours). Auto-mapping is covered later in this chapter.

Note: Mailbox permissions can also be granted to a group, instead of to individual users. The group must be a universal security group, but does not need to be mail-enabled. When a group is used to grant access to a mailbox, auto-mapping does not work, so the mailbox will need to be manually added to the group members' profiles.

The most common mailbox permissions you'll be granting is *FullAccess*. By default, the permission is applied to all folders and items. However, the default behaviour can be modified by using the *–InheritanceType* switch, which as the following options:

- All – the permission is applied to the top of the mailbox and all child objects (folders). This is the default behavior if the *–InheritanceType* switch is not specified.
- Children – the permission is applied to the immediate children only.
- Descendants – the permission is applied to the immediate children and their descendants.
- None – no inheritance is used, and the permission is applied only to the top of the mailbox.
- SelfAndChildren – the permission is applied to the top of the mailbox and its immediate children (folders), but no descendants.

Real World: There are few if any practical uses for options other than "All". However, if you run *Get-MailboxPermission* and see that a user has *FullAccess* permission to a mailbox, but is unable to access some of the child folders within that mailbox, it is worth trying to remove the mailbox permission and re-add it again, in case another administrator has used a different inheritance type.

It's natural to assume that *Add-MailboxPermission* can be used to grant read-only access to a mailbox, because one of the A*ccessRights* is *ReadPermission*. However, all *ReadPermission* does is grant a user the right to read the mailbox permissions on the mailbox object, not to actually read any of the data within the mailbox. To grant read access to a mailbox we need to use mailbox folder permissions instead, which is discussed later in this chapter.

When troubleshooting mailbox permissions issues, a common issue is that changes to the permissions do not take effect immediately. Some administrators even go as far as restarting the Information Store service, or the entire server, when trying to resolve it. A restart tends to result in the permissions changes taking effect, leading the administrator to assume that there is a fault somewhere in their server that requires a restart for

any permission changes to work. In fact, the real cause of the delay is caching of mailbox information by the Exchange server, which by default may be cached for up to 2 hours. The solution for mailbox permission changes that don't immediately take effect is therefore to simply wait longer.

Mailbox Folder Permissions

For scenarios other than granting full access to the entire contents of a mailbox, it is necessary to grant mailbox folder permissions instead. The mailbox owner, or anyone with full access to the mailbox, can make these changes in Outlook, or the administrator can do it using PowerShell.

Mailbox folder permissions are much more granular and customizable than mailbox permissions. You can configure specific permissions, such as *ReadItems*, or you can configure roles such as *Author* that include a bundle of permissions. The full list of permissions that can be applied as mailbox folder permissions is available on TechNet.

Using the example of read-only access, the first step is to grant *Reviewer* permissions to the top of the mailbox.

```
[PS] C:\> Add-MailboxFolderPermission -Identity alex.heyne:\ -User Alan.Reid -AccessRights Reviewer

RunspaceId   : 2cc2f5f2-77a3-42b6-9221-83cf24c494c6
FolderName   : Top of Information Store
User         : Alan Reid
AccessRights : {Reviewer}
Identity     : Alan Reid
IsValid      : True
```

When you're troubleshooting mailbox folder permissions it's important to understand how inheritance works. Unlike mailbox permissions, there's no inheritance applied by default when you use *Add-MailboxFolderPermission*, and there is also no parameter that you can use with the cmdlet to control inheritance. Instead, the inheritance behaviour is:

- New mailbox folder permissions do not inherit to existing sub-folders
- Newly created sub-folders will inherit the mailbox folder permissions from the parent folder

If you do need to apply mailbox folder permissions to existing sub-folders of the mailbox, you will need to apply them to each folder. For example, to grant *Reviewer* permissions to the Inbox folder, run the following command.

```
[PS] C:\> Add-MailboxFolderPermission -Identity alex.heyne:\Inbox -User Alan.Reid -AccessRights Reviewer

RunspaceId   : 2cc2f5f2-77a3-42b6-9221-83cf24c494c6
FolderName   : Inbox
User         : Alan Reid
AccessRights : {Reviewer}
Identity     : Alan Reid
IsValid      : True
```

You can be as granular with mailbox folder permissions as you like, granting Reviewer permissions to the top of the mailbox, and different permissions for sub-folders.

Note: Adding the same mailbox folder permissions for every existing folder in a mailbox is a tedious task, but you can speed it up by using the script demonstrated in this blog post.

A common error when trying to grant mailbox folder permissions is that an existing permission entry was found.

```
[PS] C:\> Add-MailboxFolderPermission -Identity alex.heyne:\Inbox -User Alan.Reid -AccessRights Editor
An existing permission entry was found for user: Alan Reid.
    + CategoryInfo          : NotSpecified: (:) [Add-MailboxFolderPermission],
UserAlreadyExis...nEntryException
```

Mailbox folder permissions can only have one entry per user or group. If there is an existing Reviewer permission on the folder for a user, and you want to grant Editor permissions instead, use the *Set-MailboxFolderPermission* cmdlet to update the entry.

```
[PS] C:\> Set-MailboxFolderPermission -Identity alex.heyne:\Inbox -User Alan.Reid -AccessRights Editor
```

Also, keep in mind that mailbox folder permissions do not use auto-mapping for the mailbox, so any mailbox access that has been configured using folder permissions will require the mailbox to be manually added to the user's Outlook profile.

Auto-Mapping

In Exchange 2010 Service Pack 1, Microsoft introduced the capability for Outlook clients to automatically add mailboxes to the Outlook profile of users who have full access to the mailbox. This greatly simplifies the process of granting access to shared mailboxes, because administrators do not need to assist the end user with adding the mailbox to their profile.

Auto-mapping is enabled by default when you grant FullAccess mailbox permissions, which was discussed earlier in this chapter. Let's look an example of Alan Reid being granted access to the Payroll shared mailbox.

```
[PS] C:\> Add-MailboxPermission -Identity Payroll -User Alan.Reid -AccessRights FullAccess
```

After running the command shown above, the **msExchDelegateLinkList** attribute of the Payroll mailbox is updated to include Alan Reid.

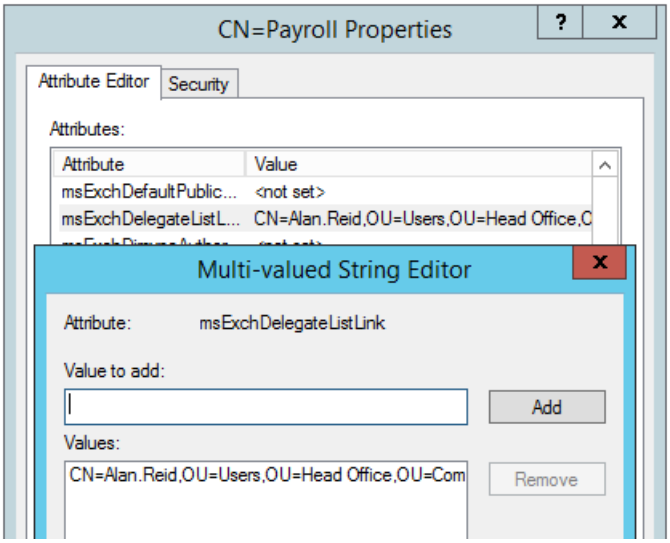


Figure 6-1: The msExchDelegateListLink attribute viewed using ADSEdit

In subsequent Autodiscover queries, the XML data returned by the Exchange server's Autodiscover service includes the Payroll mailbox as an **AlternativeMailbox**.

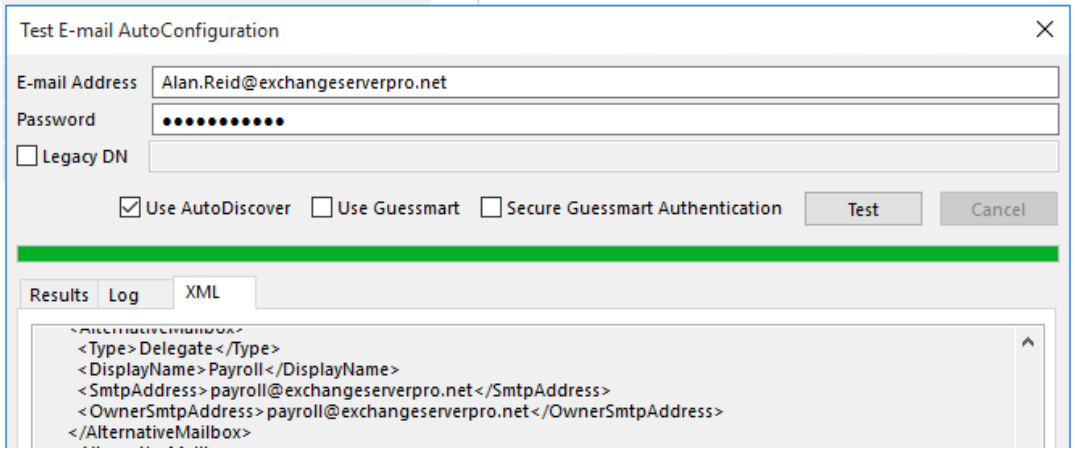


Figure 6-2: AlternativeMailbox details provided by Autodiscover

Outlook will then automatically map the alternative mailboxes for the user. This Autodiscover involvement makes three important points when troubleshooting mailbox auto-mapping issues:

- The Autodiscover service caches information, so it may not immediately return the alternative mailbox details. The MSexchangeAutodiscoverAppPool in IIS can be recycled to clear cached information.
- Outlook clients poll for Autodiscover information at regular intervals, and also at start up, but they also cache Autodiscover information locally, which may create a delay between applying mailbox permissions and auto-mapping occurring.
- If Autodiscover is not working in the environment, then auto-mapping will not work at all.

Outlook and Autodiscover are covered in more detail in chapter 7.

Although auto-mapping is convenient, it can create performance problems for the end user when too many mailboxes are auto-mapped in an Outlook profile. It's rare that one individual needs a large number of mailboxes permanently mapped, so you can reduce the number by disabling auto-mapping. It's possible to disable auto-mapping for existing mailbox permissions by re-running the *Add-MailboxPermission* cmdlet with the *–AutoMapping* parameter set to *$false*.

```
[PS] C:\> Add-MailboxPermission -Identity Payroll -user Alan.Reid -AccessRights FullAccess
-AutoMapping:$false
```

If you have multiple users with permissions to the mailbox, and you want to disable auto-mapping for all of them, you can achieve that with a few steps in PowerShell.

```
[PS] C:\> $Users = Get-MailboxPermission Payroll | Where {$_.AccessRights -eq "FullAccess" -and
$_.IsInherited -eq $false}

[PS] C:\> $Users | ForEach {Add-MailboxPermission -Identity $_.Identity -User $_.User -AccessRights
FullAccess -AutoMapping:$false}
```

Sometime after running the commands above, any users that have the mailbox auto-mapped in their Outlook profile will see that mapping remove itself. They will then need to manually map the mailbox if they want to have it permanently mapped in their Outlook profile.

Note: Auto-mapping does not work at all when mailbox permissions are granted to a group. It is sometimes more efficient to use universal security groups to grant an entire team access to their shared mailbox, because access is then managed solely through the group membership. However, it does mean you'll have the additional support burden of assisting users to manually map mailboxes in Outlook.

Corrupt Mailboxes

Under the hood, mailboxes are complicated data constructs living within a database. And like most data, from time to time they can suffer from corruption. Mailbox corruption often appears during mailbox migrations, as the migration process finds issues with old data in the mailbox that the end user never looks at any more. Migration issues caused by mailbox corruption are covered in chapter 11.

However, end users will notice more obvious signs of mailbox corruption, such as:

- Folders showing item counts that are incorrect (e.g. a folder display 1 unread item when there are in fact no unread items)
- Search folders not displaying the correct results
- Mailbox folders missing expected items from some views

If you suspect mailbox corruption, you can run a mailbox repair request in detect mode. Mailbox repair requests can be run for one or more of the following corruption types:

- SearchFolder
- AggregateCounts
- FolderView
- ProvisionedFolder

Mailbox repair requests can be run against a single mailbox, or an entire database. For example, to run a repair request for Alan Reid in detect only mode, we can use the *New-MailboxRepairRequest* cmdlet with the *–Mailbox* and *–DetectOnly* parameters.

```
[PS] C:\> New-MailboxRepairRequest -Mailbox alan.reid -CorruptionType
SearchFolder,AggregateCounts,FolderView,ProvisionedFolder -DetectOnly
```

To run a repair request against an entire mailbox, use the *–Database* parameter instead.

```
[PS] C:\> New-MailboxRepairRequest -Database DB01 -DetectOnly -CorruptionType SearchFolder
```

Mailbox repair requests are background tasks, which will start and stop depending on the performance of the Exchange server and its load at the time.

Note: For performance reasons, only one database-level repair request or 100 mailbox-level repair requests can be active at the same time. Once a repair request has started, it can't be stopped unless you dismount the mailbox database.

You can monitor the progress of your mailbox repair request by running the *Get-MailboxRepairRequest* cmdlet.

```
PS] C:\> Get-MailboxRepairRequest -Mailbox alan.reid

Identity               Task                  Detect Only       Job State         Progress
--------               ----                  -----------       ---------         --------
ac97939-c2e0-4bd3-9... {SearchFolder}        True              Succeeded         100
ac97939-c2e0-4bd3-9... {AggregateCounts}     True              Succeeded         100
ac97939-c2e0-4bd3-9... {FolderView}          True              Succeeded         100
ac97939-c2e0-4bd3-9... {ProvisionedFolder}   True              Succeeded         100
```

To view the results, look at the CorruptionsDetected and CorruptionsFixed properties of the repair requests.

```
[PS] C:\> Get-MailboxRepairRequest -Mailbox alan.reid | fl Tasks,Corruption*

Tasks               : {SearchFolder}
CorruptionsDetected : 0
CorruptionsFixed    : 0
Corruptions         :

Tasks               : {AggregateCounts}
CorruptionsDetected : 0
CorruptionsFixed    : 0
Corruptions         :

Tasks               : {FolderView}
CorruptionsDetected : 0
CorruptionsFixed    : 0
Corruptions         :

Tasks               : {ProvisionedFolder}
CorruptionsDetected : 0
CorruptionsFixed    : 0
Corruptions         :
```

If you need to repair mailboxes, re-run the mailbox repair request command without the *–DetectOnly* switch.

Corrupt Delegates

When a user adds another user to their mailbox as a delegate, and chooses to have meeting requests sent to their delegates, a hidden inbox rule is created in their mailbox to perform the forwarding of those meeting requests. This can also apply to delegates room and equipment mailboxes.

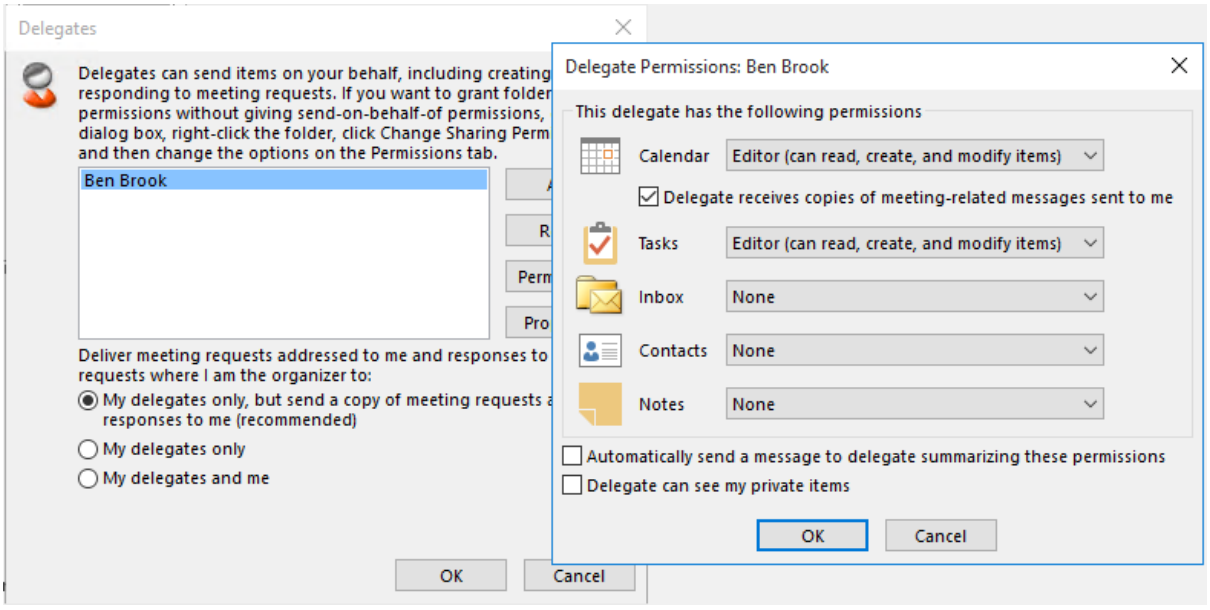


Figure 6-3: Mailbox delegate configured to receive copies of meeting requests

When the delegate is later removed from the list, they might continue to receive the meeting requests. If the former delegate's mailbox has been removed from the organization, then people who send meeting requests to the mailbox for which they were previously a delegate might receive a non-delivery report.

The root cause of both issues is a stale, or corrupt, delegate entry still being included in the hidden inbox rule. There are two solutions you can try to fix this:

- Remove all delegates from the mailbox, and then re-add them.
- Use MFCMAPI to remove the hidden inbox rule (which also removes all delegates, requiring you to re-add them).

In either case, it is wise to make a note of the delegates that you want to keep on the mailbox, and their delegate permissions, before you attempt to fix the problem.

Calendars

Users rely on their calendars to know when and where they need to be, so any calendar issues will quickly cause them pain. But most calendar issues come down to a handful of root causes, which we'll explore in this section.

Calendar Permissions

The permissions for calendars can be configured in two ways:

- By an administrator configuring mailbox folder permissions. The calendar is just another folder in that sense.
- By the end user configuring their own delegates or folder permissions.

Mailbox folder permissions have already been covered earlier in this chapter.

Calendar Time Zones and Work Hours

Users can, in theory, work from anywhere in the world. So it makes sense that their calendars would be configured to know which time zone they work in. When a user logs into Outlook on the web (OWA) for the first time they are prompted to set their time zone. If they have not used OWA, then the mailbox time zone defaults to the time zone of the Exchange server. If the user does not work in the same time zone as the server then this can cause some issues with time zone offset for their calendar items.

The mailbox calendar time zone can be viewed by running *Get-MailboxCalendarConfiguration*.

```
[PS] C:\> Get-MailboxCalendarConfiguration –Identity Dawn.Evans | Fl

RunspaceId            : 9646d4ad-70ba-48d9-98ae-db322193f06b
WorkDays              : Weekdays
WorkingHoursStartTime : 08:00:00
WorkingHoursEndTime   : 17:00:00
WorkingHoursTimeZone  : W. Australia Standard Time
WeekStartDay          : Monday
ShowWeekNumbers       : False
FirstWeekOfYear       : FirstDay
TimeIncrement         : ThirtyMinutes
RemindersEnabled      : True
ReminderSoundEnabled  : True
DefaultReminderTime   : 00:15:00
WeatherEnabled        : True
WeatherUnit           : Default
WeatherLocations      : {}
```

```
Identity              : exchangeserverpro.net/Company/Head Office/Users/Dawn.Evans
IsValid               : True
ObjectState           : New
```

Similarly, the work hours setting for a mailbox will impact the classification of free/busy time for the calendar, and in the case of resource mailboxes, will also impact the suggested times and enforcement of bookings only within work hours.

You can modify the time zone or work hours for a mailbox by running the *Set-MailboxCalendarConfiguration* cmdlet.

```
[PS] C:\> Set-MailboxCalendarConfiguration -Identity dawn.evans -WorkingHoursTimeZone "W. Australia
Standard Time"
```

A full list of valid time zones can be displayed by running the following command in PowerShell.

```
[PS] C:\> $TimeZone = Get-ChildItem "HKLM:\Software\Microsoft\Windows NT\CurrentVersion\Time zones" |
foreach {Get-ItemProperty $_.PSPath}; $TimeZone | sort Display | Format-Table -Auto PSChildname,Display
```

Free/Busy Information

When a user is trying to book a resource mailbox or another person for a meeting, they rely on free/busy information displayed in Outlook (Figure 6-4) to let them know when the meeting invitees are available.

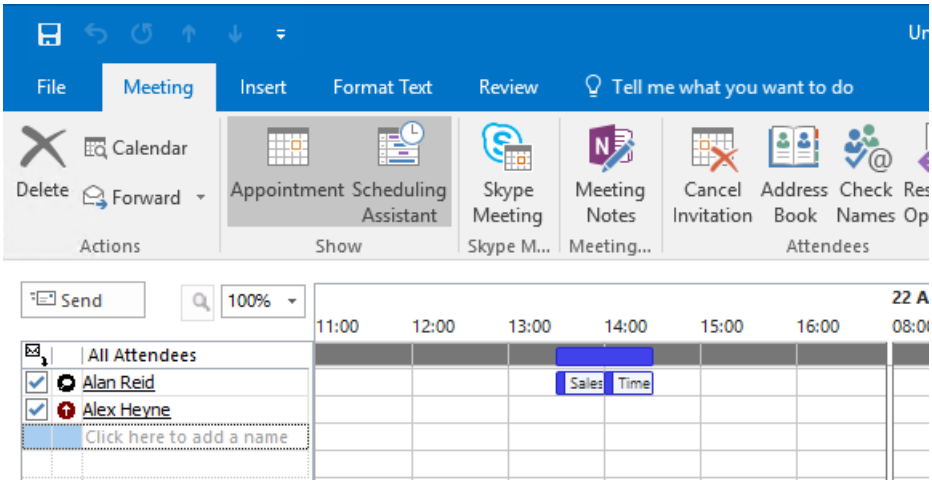


Figure 6-4: Free/busy information displayed in Outlook.

Free/busy information relies on Autodiscover for the client to be able to discover where the Exchange Web Services (EWS) endpoint is, and also relies on EWS being reachable. This means that the EWS namespace (URL) must be correctly configured, resolvable in DNS, and reachable across the network. And, because the EWS connection from Outlook is over HTTPS, the SSL certificate on the Exchange server must also be correctly configured. Namespaces and certificates have been covered in chapter 3, and Outlook connectivity to EWS is covered in chapter 7.

Real World: One of the most obvious signs of EWS not working is when free/busy lookups are failing. The other is when an Outlook user can't change their Out of Office settings. Both of those features rely on EWS.

Calendar Repair

We've already looked at mailbox repairs in this chapter, which focuses on four distinct mailbox corruption scenarios. In addition to mailbox repairs, Exchange also performs calendar repairs on mailboxes, using the Calendar Repair Assistant (CRA).

The CRA is set to run in *RepairAndValidate* mode by default for Exchange 2013 and later, which you can view by running *Get-MailboxServer*.

```
[PS] C:\> Get-MailboxServer | Select Name,CalendarRepairMode

Name       CalendarRepairMode
----       ------------------
EX2013SRV1 RepairAndValidate
EX2010SRV1 ValidateOnly
EX2016SRV1 RepairAndValidate
EX2016SRV2 RepairAndValidate
```

Individual mailboxes can also be enabled or disabled for calendar repair. By default, mailboxes are enabled.

```
[PS] C:\> Get-Mailbox alan.reid | select name,calendarrepair*

Name                   : Alan.Reid
CalendarRepairDisabled : False
```

The goal of the CRA is to detect and fix inconsistencies in calendar items so that users have reliable calendar information to work from. Inconsistencies can be introduced to calendar items by things such as the meeting organizer dragging the meeting to a different time in their calendar, or cancelling the meeting without sending an update to invitees. The CRA attempts to detect these inconsistencies and resolve them.

A full list of CRA actions for different inconsistencies is available on TechNet. An actions taken by the CRA are logged to the calendar repair log path on the Mailbox server, so that in a troubleshooting situation you can review the log and determine whether the CRA has taken action on a calendar item.

```
[PS] C:\> Get-MailboxServer | Select Name,CalendarRepairLogPath | ft -auto

Name       CalendarRepairLogPath
----       ---------------------
EX2013SRV1 C:\Program Files\Microsoft\Exchange Server\V15\Logging\Calendar Repair Assistant
EX2010SRV1 C:\Program Files\Microsoft\Exchange Server\V14\Logging\Calendar Repair Assistant
EX2016SRV1 C:\Program Files\Microsoft\Exchange Server\V15\Logging\Calendar Repair Assistant
EX2016SRV2 C:\Program Files\Microsoft\Exchange Server\V15\Logging\Calendar Repair Assistant
```

In Exchange 2013 and later the CRA is a workload-based process, meaning it will run automatically behind the scenes, stopping and starting depending on the server load at the time.

Real World: It's easy to suspect the Calendar Repair Assistant when calendar items go missing unexpectedly. But a far more common cause of missing calendar items is mistaken deletion by the mailbox owner or their delegate. Use mailbox audit logging (chapter 12) to investigate further.

Resources

Room and equipment mailboxes are, as the names suggest, mailboxes used to manage the calendar bookings for meeting rooms and other resources such as loan laptops and pool cars. Room and equipment mailboxes are similar to user mailboxes, but surface a different set of properties in the Exchange admin center, such as booking options and room capacity.

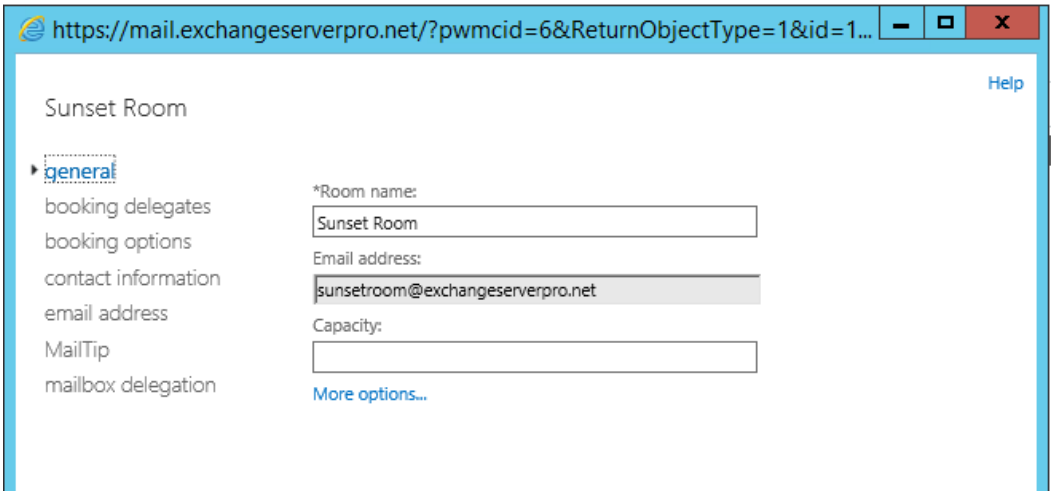


Figure 6-4: Booking options visible in the properties of a room mailbox

Converting Mailboxes to Resource Mailboxes

If you don't see the booking options for a mailbox that you intend to be used as a room or equipment mailbox, it's possible that the mailbox has been created as the wrong type, such as a user or shared mailbox.

```
[PS] C:\> Get-Mailbox "Lakeview Room" | Select Name,RecipientType*

Name                 : Lakeview Room
RecipientType        : UserMailbox
RecipientTypeDetails : UserMailbox
```

In the example above, the Lakeview Room mailbox has been created as the wrong recipient type. To convert it to a room mailbox, use the *Set-Mailbox* cmdlet.

```
[PS] C:\> Set-Mailbox "Lakeview Room" -Type Room
```

Similarly, you can run the same command to convert a mailbox to Regular, Shared, or Equipment.

Note: When you convert a mailbox to a room, equipment, or shared mailbox, the associated user account object in Active Directory is disabled. This prevents anyone directly logging on to the account.

Automating Bookings with Calendar Processing

One of the main benefits of room and equipment mailboxes is the capability to automatically process meeting requests, effectively providing a self-service model for users in your environment to book resources that they need for meetings and other events.

However, there are a number of ways that the processing of meeting requests will behave, depending on how you've configured the booking options for the mailbox. By default, a resource mailbox will not automatically process meeting requests.

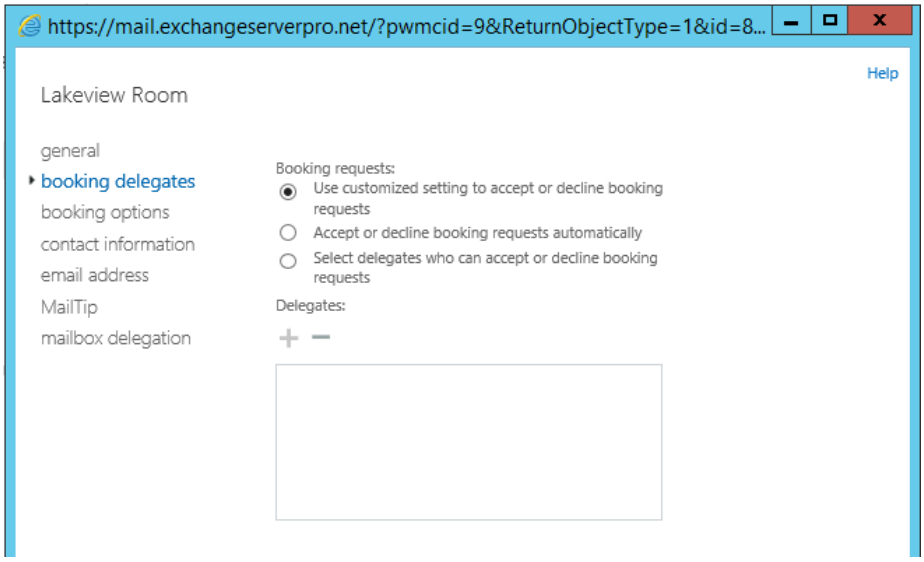


Figure 6-5: Default meeting request processing for a resource mailbox

Although the basic calendar processing configuration for a resource mailbox can be managed in the Exchange admin center, PowerShell offers a greater level of control. To view the calendar processing settings for a mailbox, use the *Get-CalendarProcessing* cmdlet.

```
[PS] C:\> Get-CalendarProcessing -Identity Lakeview.Room

Identity                                                  AutomateProcessing
--------                                                  ------------------
exchangeserverpro.net/Company/Head Office/Users/Lakeview ... AutoUpdate
```

To turn on automated processing, use *Set-CalendarProcessing* to set the calendar to *AutoAccept*.

```
[PS] C:\> Set-CalendarProcessing -Identity Lakeview.Room -AutomateProcessing AutoAccept
```

Users will now be able to book the resource, within the constraints of the booking policies which are discussed next.

Controlling Automatic Calendar Processing

Although automation provides a nice, low-effort way for resource mailboxes to be booked, there are scenarios in which you want to control who can book resources, and when they can be booked.

By default, a resource mailbox has several restrictions in place that are enforced by the Resource Booking Attendant:

- Conflicting bookings are not allowed
- Bookings can be made up to 180 days in the future
- Meeting instances can last no longer than 24 hours
- Recurring meetings are allowed
- Bookings are allowed at any time of day
- External senders can't book resources

You can see these settings, some of which are the result of a combination of multiple settings, by running *Get-CalendarProcessing*.

```
[PS] C:\> Get-CalendarProcessing -Identity Lakeview.Room | fl

RunspaceId                          : 78f05d8a-31d0-4efc-9a99-ccb73993e4dc
AutomateProcessing                  : AutoAccept
AllowConflicts                      : False
BookingWindowInDays                 : 180
MaximumDurationInMinutes            : 1440
AllowRecurringMeetings              : True
EnforceSchedulingHorizon            : True
ScheduleOnlyDuringWorkHours         : False
ConflictPercentageAllowed           : 0
MaximumConflictInstances            : 0
ForwardRequestsToDelegates          : True
DeleteAttachments                   : True
DeleteComments                      : True
RemovePrivateProperty               : True
DeleteSubject                       : True
AddOrganizerToSubject               : True
DeleteNonCalendarItems              : True
TentativePendingApproval            : True
EnableResponseDetails               : True
OrganizerInfo                       : True
ResourceDelegates                   : {}
RequestOutOfPolicy                  : {}
AllRequestOutOfPolicy               : False
BookInPolicy                        : {}
AllBookInPolicy                     : True
RequestInPolicy                     : {}
AllRequestInPolicy                  : False
AddAdditionalResponse               : False
AdditionalResponse                  :
RemoveOldMeetingMessages            : True
AddNewRequestsTentatively           : True
ProcessExternalMeetingMessages      : False
RemoveForwardedMeetingNotifications : False
MailboxOwnerId                      : exchangeserverpro.net/Company/Head Office/Users/Lakeview Room
Identity                            : exchangeserverpro.net/Company/Head Office/Users/Lakeview Room
IsValid                             : True
ObjectState                         : Changed
```

Some restrictions are easy to apply, such as limiting the duration of meetings in a high demand meeting room to 1 hour.

```
[PS] C:\> Set-CalendarProcessing -Identity Lakeview.Room -MaximumDurationInMinutes 60
```

Other restrictions might require a combination of settings. For example, limiting bookings to work hours only. Part of this configuration is performed with *Set-CalendarProcessing*.

```
[PS] C:\> Set-CalendarProcessing -Identity Lakeview.Room -ScheduleOnlyDuringWorkHours:$true
```

However, the work hours are configured in the mailbox calendar configuration, which can be viewed using *Get-MailboxCalendarConfiguration*.

```
[PS] C:\> Get-MailboxCalendarConfiguration -Identity Lakeview.Room | fl

RunspaceId            : 78f05d8a-31d0-4efc-9a99-ccb73993e4dc
WorkDays              : Weekdays
WorkingHoursStartTime : 08:00:00
WorkingHoursEndTime   : 17:00:00
WorkingHoursTimeZone  : Pacific Standard Time
WeekStartDay          : Sunday
ShowWeekNumbers       : False
FirstWeekOfYear       : FirstDay
TimeIncrement         : ThirtyMinutes
RemindersEnabled      : True
ReminderSoundEnabled  : True
DefaultReminderTime   : 00:15:00
WeatherEnabled        : True
WeatherUnit           : Default
WeatherLocations      : {}
Identity              : exchangeserverpro.net/Company/Head Office/Users/Lakeview Room
IsValid               : True
ObjectState           : New
```

In some cases, human management of bookings is necessary. Resource mailboxes can have delegates assigned to process meeting requests.

```
[PS] C:\> Get-CalendarProcessing -Identity Lakeview.Room | fl AutomateProcessing,*delegates

AutomateProcessing         : AutoAccept
ForwardRequestsToDelegates : True
ResourceDelegates          : {}

[PS] C:\>Set-CalendarProcessing -Identity Lakeview.Room -ResourceDelegates dawn.evans

[PS] C:\>Get-CalendarProcessing -Identity Lakeview.Room | fl AutomateProcessing,*delegates

AutomateProcessing         : AutoAccept
ForwardRequestsToDelegates : True
ResourceDelegates          : {exchangeserverpro.net/Company/Head Office/Users/Dawn.Evans}
```

However, you should be aware that adding a delegate is not enough to require manual processing of meeting requests. You'll also need to set the **AllBookInPolicy** setting to *False*, and **AllRequestInPolicy** to *True*.

```
[PS] C:\> Set-CalendarProcessing -Identity Lakeview.Room -AllBookInPolicy:$false
–AllRequestInPolicy:$true
```

Even with a delegate in place to manage resource bookings, it's sometimes desirable to allow specific individuals to make their own bookings. This is where the booking policy and request policy settings can be used to provide the desired outcome. The policy requirements that are assessed by the Resource Booking Attendant are those mentioned earlier in this section (e.g. meeting duration, whether recurring meetings are allowed, and so on), and you can read a full list of the available settings and their meaning on the TechNet page for Set-CalendarProcessing.

Note: The policy settings will have no effect at all if the **AutomateProcessing** setting is not set to *AutoAccept* to enable the Resource Booking Attendant.

As you can see there are multiple settings that need to be considered as part of a whole configuration in order to determine what will actually happen when a meeting request is sent that includes a resource booking. This also tends to make troubleshooting complex. Here's a few examples of the behaviour you can expect to see.

Example 1: Fully automated resource booking. All resources are available for anyone to book.

- AutomateProcessing: AutoAccept
- AllBookInPolicy: True

Example 2: Resource bookings fully managed by a delegate. The delegate ensures that reasonable resource bookings are made, and handles any conflicts that arise.

- AutomateProcessing: AutoAccept
- AllBookInPolicy: False
- AllRequestInPolicy: True
- ResourceDelegates: List of delegates

Example 3: Policy settings apply for most user requests, with some users allowed to make out of policy bookings.

- AutomateProcessing: AutoAccept
- AllBookInPolicy: True
- AllRequestInPolicy: n/a
- AllRequestOutOfPolicy: False
- RequestOutOfPolicy: List of users allowed to make out of policy requests

Real World: The more complex your requirements, the more thought needs to be put into how to control resource bookings. Keeping things simple will make your administrative life easier, and create fewer surprises for your end users who are trying to make resource bookings. You should also consider whether your configuration is likely to make things worse when there is high demand for a limited number of resources.

Troubleshooting Resource Bookings

No matter how well you fine tune your resource mailbox settings, there will eventually be some problem that occurs with end users trying to book the resources. The most common issues are:

- Requests declined if they are too far in the future. If a user tries to book a meeting room too far into the future, the Resource Booking Attendant will decline it. This can be controlled by configuring the **BookingWindowInDays** setting.
- Recurring meeting requests are declined if the occurrences extend too far in the future. This will occur if a recurring meeting starts before the **BookingWindowInDays**, but extends beyond that limit. This is controlled by configuring the **EnforceSchedulingHorizon** setting. If set to *False*, the meeting is not rejected, but any occurrences beyond the horizon will be removed.
- Requests declined due to conflicts. By default, no conflicting appointments are accepted. However, this can become troublesome when a recurring meeting is booked, and only a few of the meeting

instances have a conflict. This can be controlled by configuring the **ConflictPercentageAllowed** setting.
- Requests are not received by resource delegates. This can occur when the **AutomateProcessing** setting has not been configured to *AutoAccept*.
- Conflicts are not permitted in the policy, but conflicting appointments still occur. This can happen if a user has calendar permissions that allow them to make direct bookings in the calendar without sending a meeting request. Direct bookings are not controlled by the Resource Booking Attendant so there is nothing to stop someone manually causing a conflict like this. The solution is to limit direct access to resource calendars, and require users to send meeting requests instead.

Groups

Exchange distribution groups are used to send email to multiple recipients. The more accurate name is mail-enabled distribution groups, because a distribution group can exist in Active Directory even when it is not mail-enabled. But for the purposes of this section, assume that any reference to distribution group implies that it is mail-enabled.

Correct Usage of Group Types in Exchange

There are a few basics to be aware of when it comes to using groups with Exchange server. Often a mistake with these basics results in a group not functioning as intended, so a lot of group problems can be fixed by revisiting these points:

- Groups must be mail-enabled to be visible in Exchange address lists, and to be able to be used to distribute email.
- Hiding a distribution group from the global address list does not prevent email from being sent to the distribution group if the SMTP address is known by the sender.
- Exchange can only use universal groups. Global or domain local groups will not work, but may be present from a legacy version of Exchange.
- Mail-enabled security groups can be used to apply permissions to objects, as well as to send email. Mail-enabled distribution groups can only be used to send email, and can't be used to apply permissions to objects.
- An empty group will not cause any non-delivery reports. But obviously any email sent to an empty group will not be received by anybody.
- By default, distribution groups do not accept email from unauthenticated senders.
- If you create an In-Place Hold for a group, the group membership is expanded at the time the hold is created, and the group members are included in the In-Place Hold. If the group membership later changes, the mailboxes included in the In-Place Hold do not automatically update.

Sensitive Groups and Exchange Permissions

In some organizations there are sensitive groups that are restricted from access by all but a few administrators. I'm not referring to privileged groups built in to Active Directory, such as Domain Admins. Rather I am referring to groups that are often created to hold VIP users in an organization, such as an "Executives" group. To prevent modification of these sensitive groups by low level administrators, the groups are often placed into different organizational units (OUs) in Active Directory, and then the OU permissions adjusted to prevent access by low level admins.

This tends to break the group from an Exchange perspective. Exchange needs permission to read the group membership so that it can determine who to send an email to when the message is addressed to that group. If

the OU has been locked down in a way that causes the Exchange ACLs to be removed, then Exchange can't read the membership and any emails send to the group will not be received.

To resolve this issue, enable permission inheritance for the OU so that the Exchange ACLs re-apply. Then, disable inheritance, but choose to convert the inherited permissions into explicit permissions on the object.

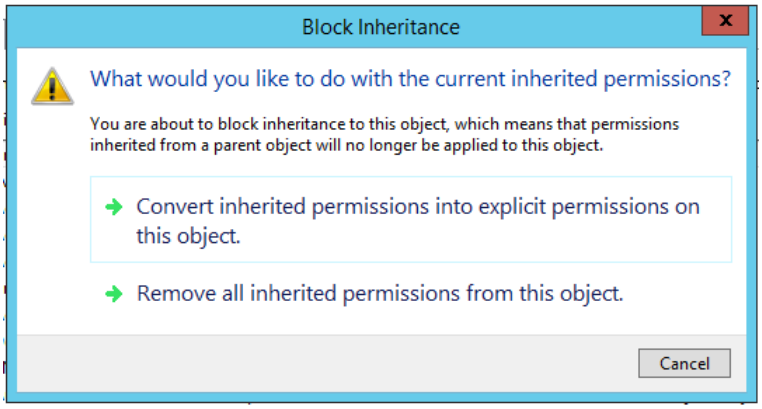


Figure 6-7: Disabling inheritance for an OU.

This preserves the Exchange ACLs, and you can then selectively remove the other permissions that you don't want on the OU.

Note: This applies to nested groups as well. Exchange needs to be able to read all of the parent and child groups when nesting is being used. If any of the groups are inaccessible to Exchange, it will "fail closed" by not sending email to the group members, even those it was able to read from the groups it could access.

Dynamic Distribution Groups

Dynamic distribution groups serve the same function as other distribution groups – sending email to multiple recipients. However, instead of a static membership list like a regular distribution group, a dynamic distribution group uses an LDAP query (or recipient filter) to determine who should receive the email message, for example all users whose department is set to "Human Resources". The query is assessed every time an email is sent to the dynamic distribution group, ensuring that the group membership is always up to date (assuming your recipient filter criteria and Active Directory are accurate and up to date as well).

There are two common issues with dynamic distribution groups:

- An incorrect recipient filter causes the wrong recipients to be included or excluded in the group membership.
- The recipient container for the group is incorrect, so that even when the recipient filter is correct the intended recipients can't be located when calculating membership.

To view the recipient filter for a dynamic distribution group, run the *Get-DynamicDistributionGroup* cmdlet.

```
[PS] C:\> Get-DynamicDistributionGroup "DDG - Head Office" | select RecipientFilter

RecipientFilter : ((((RecipientTypeDetails -eq 'UserMailbox') -and (Office -eq 'Head Office')))
                  -and (-not(Name -like 'SystemMailbox{*')) -and (-not(Name -like 'CAS_{*')) -and (-
                  not(RecipientTypeDetailsValue -eq 'MailboxPlan')) -and (-
                  not(RecipientTypeDetailsValue -eq 'DiscoveryMailbox')) -and
                  (-not(RecipientTypeDetailsValue -eq 'PublicFolderMailbox')) -and (-
                  not(RecipientTypeDetailsValue -eq 'ArbitrationMailbox')) -and (-
```

```
                not(RecipientTypeDetailsValue -eq 'AuditLogMailbox')))
```

To test a recipient filter, capture the dynamic DG as a variable, and then run *Get-Recipient*.

```
[PS] C:\> $ddg = Get-DynamicDistributionGroup "DDG - Head Office"

[PS] C:\> Get-Recipient -RecipientPreviewFilter $ddg.RecipientFilter
```

The command above will test the recipient filter, but won't validate that the recipient container for the dynamic DG is correctly specified.

```
[PS] C:\> Get-DynamicDistributionGroup "DDG - Head Office" | select RecipientContainer

RecipientContainer
------------------
exchangeserverpro.net/Users
```

In the example above, the dynamic DG will only calculate the users who match the recipient filter and are located in the exchangeserverpro.net/Users OU or any child OU. If the recipients are located in a different part of the Active Directory OU structure that doesn't fall under that container, then they will not receive emails sent to the dynamic DG.

Real World: Setting the recipient container to the domain root is the simplest approach, and should work fine in smaller environments. However, in very large, complex environments, setting the recipient container to the domain root may cause a high load on servers as the dynamic DG membership queries run. In such cases, narrow the scope of the query by setting the recipient container to an OU that more closely represents where the dynamic DG members are located.

Email Delivery

Across most recipient types the same email delivery issues can occur. There are in fact quite a few different factors that can prevent an email from reaching a recipient. Troubleshooting transport and mail flow has already been covered in chapter 4, but in this chapter let's take a look at things from a recipient perspective.

Non-Delivery Reports

A non-delivery report (NDR) is not a cause of email delivery issues, but does usually provide diagnostic information that helps with troubleshoot. Always read the NDR when one is available. Many issues reported in NDRs will be server-related, and you should refer to chapter 4 for more guidance on those.

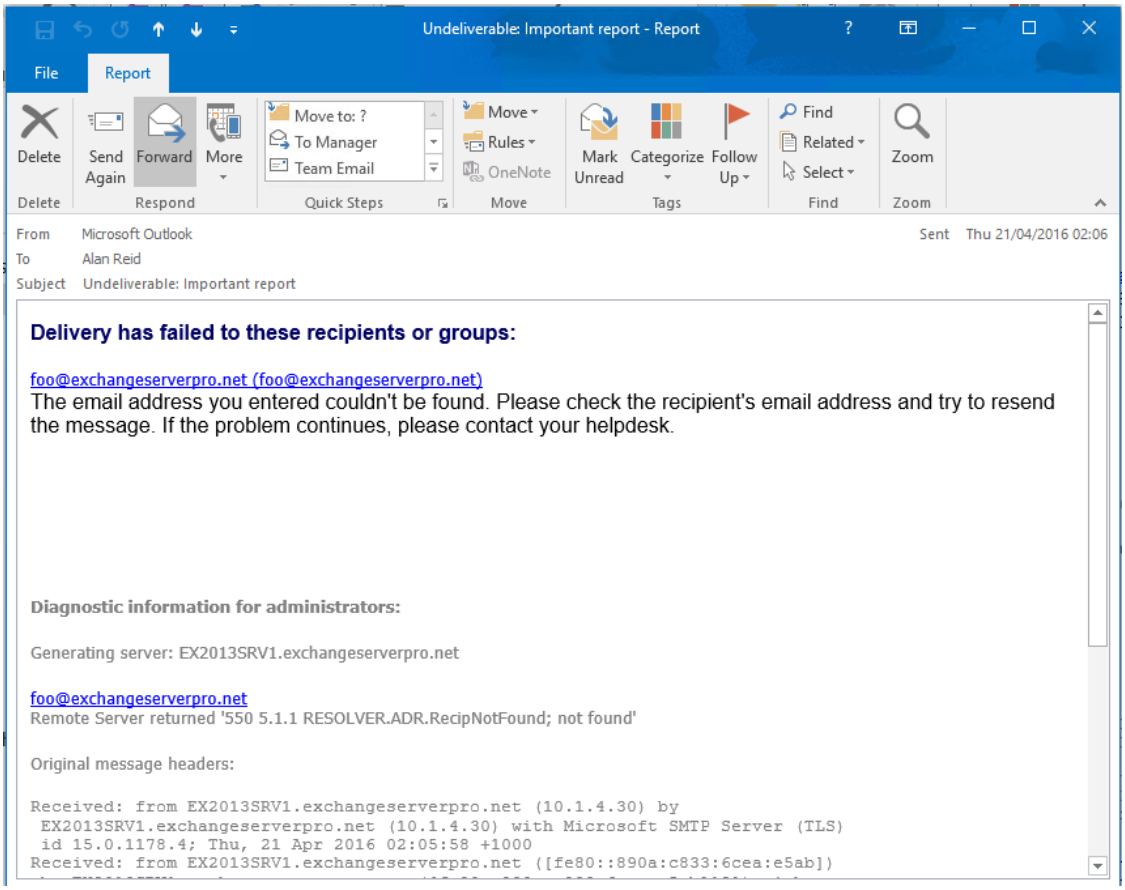


Figure 6-8: A non-delivery report for a non-existent recipient.

However, the NDR may reveal an issue that is more likely to be recipient-related.

The Infamous IMCEAEX Non-Delivery Report

Throughout this section many of the email delivery issues discussed will cause an NDR, but one in particular deserves special mention; the IMCEAEX non-delivery report. This NDR will occur when the **LegacyExchangeDN** attribute has changed for a recipient. A common reason for the attribute changing is when a user has left the organization, and their mailbox has been removed, and then they have later re-joined the organization and had a new account and mailbox created with the same name and SMTP address.

When other users in the organization sent email to that recipient, an entry was added to their Outlook auto-complete cache. Auto-complete cache entries utilize the LegacyExchangeDN attribute, which is always unique. When the new account and mailbox is created, a new LegacyExchangeDN value is used, so the auto-complete cache entries no longer match.

There are two solutions to this problem:

- Clear the auto-complete cache entries for users in the organization. This solution does not scale well, and is inconvenient for end users, so it is not recommended.
- Add an X500 address to the new mailbox for the old LegacyExchangeDN value.

The X500 address can be calculated from the NDR by following the steps provided in this Microsoft Support article.

Email Forwarding

It's quite normal to forward the emails from one mailbox to another one, especially in cases where a person has left the organization and another person needs to continue receiving their emails for a period of time afterwards, but they don't want the hassle of managing them in a separate mailbox.

Email forwarding can be configured in the **mailbox features** section of the mailbox properties (Figure 6-9).

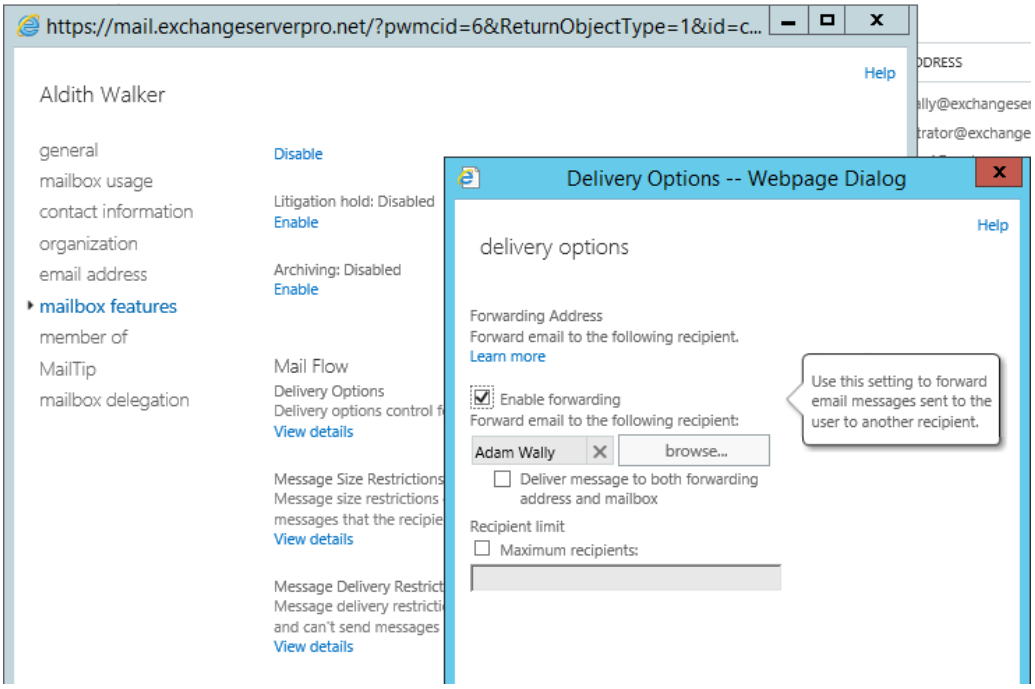


Figure 6-9: Forwarding email in the mailbox properties.

When mail forwarding is configured using that option, the **altRecipient** property of the Active Directory object is updated, which can be seen using ADSIEdit (Figure 6-10).

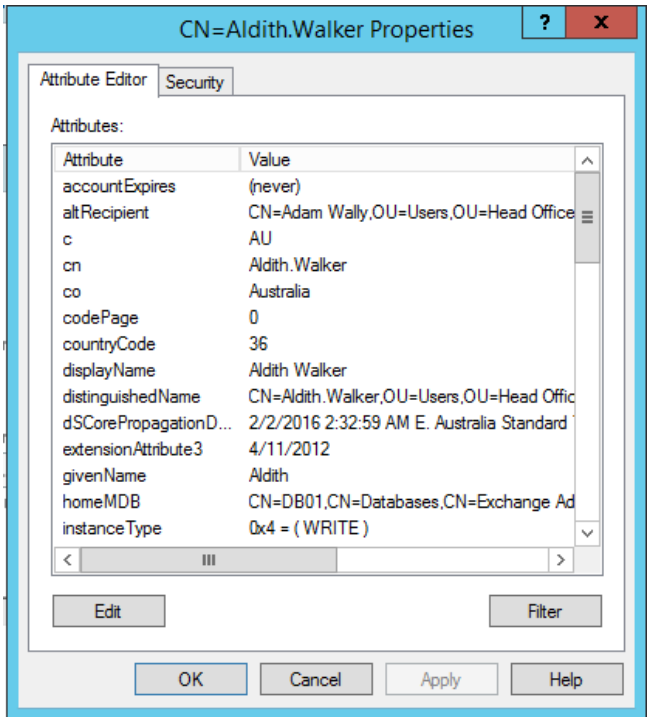


Figure 6-10: Mail forwarding values as seen in ADSIEdit.

However, this is not the only way that email can be forwarded. Many migration and co-existence tools, including Microsoft's native process for performing staged and hybrid migrations to Office 365, use a different attribute called **targetAddress** (Figure 6-11). Aside from Office 365 scenarios, you'll most often see that attribute used in cross-forest migrations.

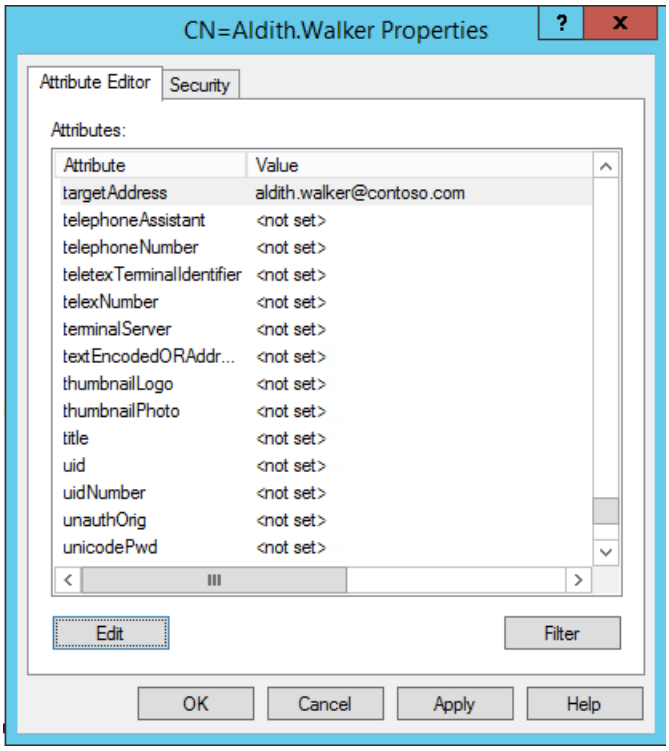


Figure 6-11: The targetAddress field as seen in ADSIEdit.

When the **targetAddress** property has been mistakenly set, or not removed after a migration, you can clear it using ADSIEdit to stop the mail forwarding.

Inbox Rules

Even when an email message has been successfully delivered, the end user may think that it was not received due to an inbox rule deleting or moving the item. When you are troubleshooting an email delivery case, a message tracking log search is a good way to determine whether the message was delivered to the mailbox or not. Message tracking is explained in chapter 4.

After confirming that an email message was successfully delivered, there are three remaining possibilities as to why it is not visible to the recipient:

- Client connectivity issues are preventing them from seeing new email messages. Clients are covered in more detail in chapter 7.
- Someone has moved or deleted the item, usually by accident. This could be either the owner or a delegate. Mailbox audit logging can help you investigate this possibility, if audit logging is correctly configured. Mailbox audit logging is explained in chapter 12. If mailbox audit logging is not enabled, then you can still perform searches of the mailbox within Outlook to locate the missing item.
- Junk mail filtering has moved the item to the Junk E-Mail folder.
- An inbox rule has moved or deleted the message.

You can check for inbox rules by running the *Get-InboxRule* cmdlet.

```
[PS] C:\> Get-InboxRule -Mailbox alan.reid | Select Name,Description | fl

Name        : Report
Description : If the message:
                  the message includes specific words in the subject 'Report'
              Take the following actions:
                  copy the message to the following folder: 'Reports'
```

Delivery Restrictions

Recipients can have delivery restrictions applied that control who can send email to that recipient. Delivery restrictions are often used for very large distribution groups to prevent misuse within the organization. For example, the "All Staff" distribution group of a company can be configured so that only the executive team and approved communications officers can send email messages. Aside from preventing misuse this also prevents "reply all" storms on very large distribution lists.

For distribution groups the **delivery management** settings can be used to control whether a group can be used by internal senders only, or by internal and external senders (Figure 6-12). Alternatively, individual users can be added so that only those approved users can send to the group.

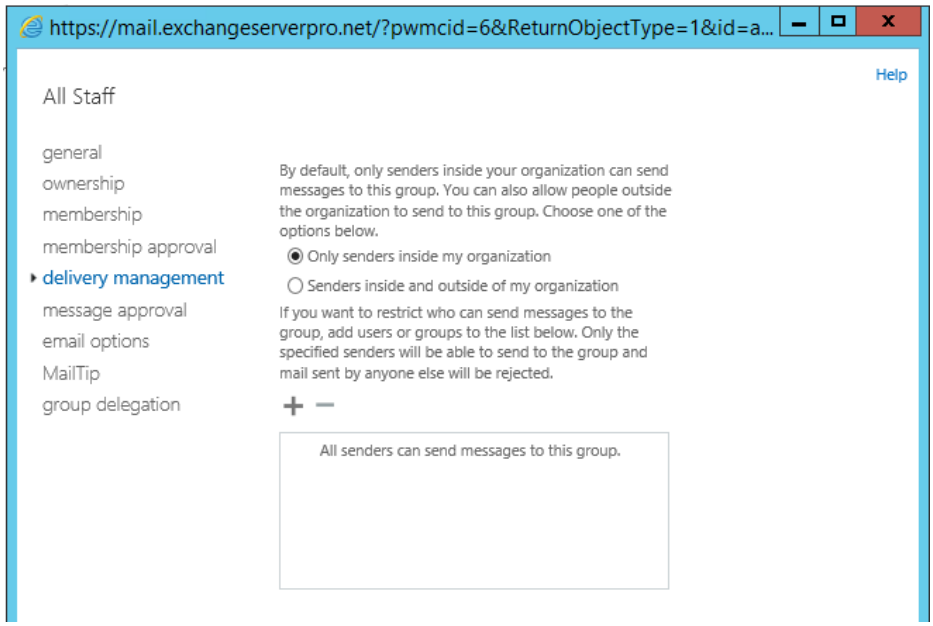


Figure 6-12: Delivery management settings for a distribution group.

For mailboxes, similar controls are available in the **Message Delivery Restrictions** settings (Figure 6-13).

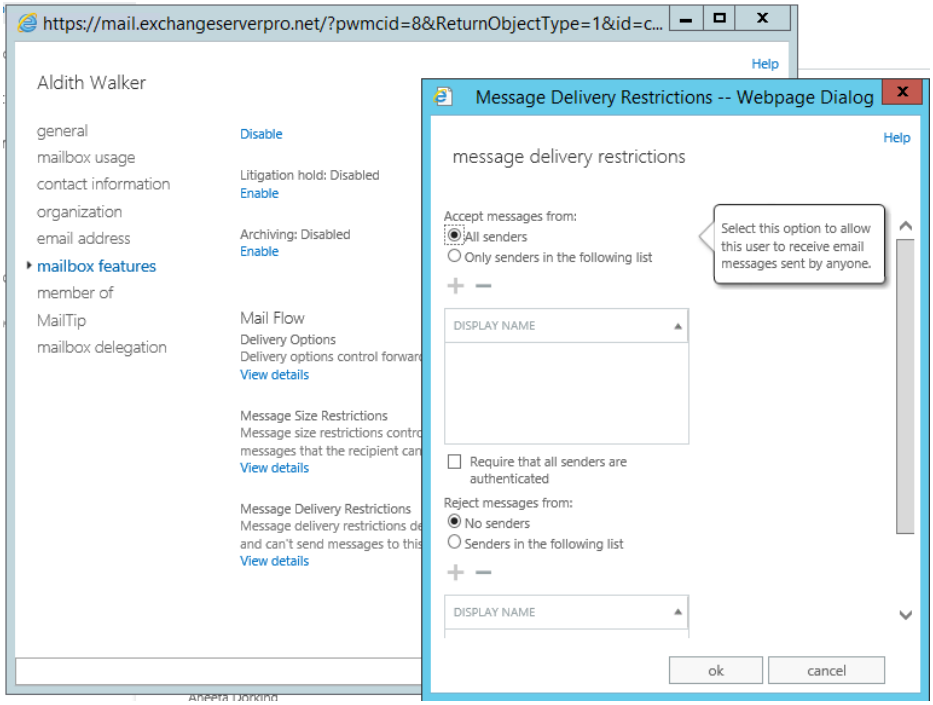


Figure 6-13: Delivery restrictions for a mailbox.

Mailboxes also have an option to require that all senders are authenticated. In effect this is the same setting as the distribution group option to restrict internal vs external senders. Both options configure the same attribute on the object, which is visible in the Exchange Management Shell.

```
[PS] C:\> Get-Mailbox Alan.Reid | fl RequireSenderAuthenticationEnabled

RequireSenderAuthenticationEnabled : False


[PS] C:\>Get-DistributionGroup AllStaff | fl  RequireSenderAuthenticationEnabled

RequireSenderAuthenticationEnabled : True
```

The default setting for mailboxes is to not require authenticated senders. In other words, a mailbox can receive email from anybody inside or outside of the organization. Groups have a default setting that requires the sender be authenticated, which effectively means only internal users can send to the group. Groups created prior to Exchange Server 2007 did not have this restriction enabled by default, so it's possible that the oldest groups in your organization are open to external senders.

You can use *Set-Mailbox* or *Set-DistributionGroup* to reconfigure the recipient's restrictions to suit your requirements.

Real World: Groups are often used to receive email from external senders, so the correct solution is not necessarily to disable external senders for all groups. However, if you're troubleshooting misuse of distribution groups by spammers or external senders, you can consider locking the group down to internal senders only if it has no external communication requirements. Alternatively, you can change the SMTP address for the group. However that is not 100% effective, and may impact other legitimate senders who are using the SMTP address.

Send-As versus Send on Behalf

With Exchange there are two methods that can be used to grant a user the ability to send email from another address:

- Send-As – this effectively allows impersonation of a mailbox, with the recipient seeing the message as being sent by the mailbox that the real sender sent it as. This is often used with shared mailboxes, such as a Help Desk team who need to send email from "Help Desk" and not their personal mailboxes.
- Send on Behalf – this allows one user to send email on behalf of another, with the recipient seeing the email as being from "Person A on behalf of Person B". This is often used with delegates.

When send-as and send on behalf are configured for the same person, they'll likely receive a non-delivery report each time they try to send as the shared mailbox or other user. The reason is that send-as and send on behalf are incompatible rights, and can't co-exist. A user should be granted either send-as, or send on behalf permissions, depending on what they need to do.

When troubleshooting email delivery problems when you're sure that either send-as or send on behalf are configured, make sure you quickly check for the presence of the other option as well.

Mailbox Storage Quotas

Mailboxes can be configured with storage quotas to prevent them from growing to an unmanageable size. By default, mailboxes inherit the storage quotas configured on the mailbox database that hosts the mailbox. The default storage quotas for Exchange 2013 and above are about 2GB, give or take a few hundred MB depending on the threshold in question. There are three thresholds:

- Warning quota – the user is warned that their mailbox is nearly full.
- Prohibit send quota – the mailbox user is no longer able to send email messages, whether internal or external. However, they can continue to receive emails.
- Prohibit send receive quota – the mailbox user is unable to send or receive emails. This prevents unrestricted growth, particularly for abandoned or unmonitored mailboxes.

When an email has been rejected due to a mailbox storage quota, that reason will be provided in the non-delivery report.

Individual mailboxes can have storage quota settings that are different from the database that hosts the mailbox. Individual mailbox storage quotas will only take effect if the **UseDatabaseQuotaDefaults** option is set to *True*. While that option is True, no amount of modifying a mailbox's individual storage quotas will have any effect.

Additional reading

Mailboxes

- Unexpected Permissions Appearing on Exchange Server Mailboxes
- How to Grant Read Only Access to an Exchange Mailbox
- How to Remove Auto-Mapping in Outlook
- Deleted Delegates Still Receive Meeting Invites for Other Mailbox Users

Calendars

- How to Change and Correct the Time Zone Values and Working Hours for a Conference Room

- Understanding Calendar Repair
- Managing the Resource Booking Attendant with Set-CalendarProcessing
- Shared Calendar Permissions and Nested Groups

Groups

- Important Information about Group Expansion for In-Place Holds
- Pros and Cons of Using Separate Security and Distribution Groups
- Remember the Basics When Working with Dynamic Distribution Groups

Email Delivery

- IMCEAEX Non-Delivery Report When You Send Email Messages to an Internal User
- The Attribute, the Myth, the LegacyExchangeDN

Chapter 7: Troubleshooting Clients

Paul Cunningham

Exchange clients cover the set of applications and devices that connect to Exchange to send and receive messages or otherwise interact with mailboxes and mailbox items. Exchange supports multiple client connection protocols. Some are hosted by Client Access services, and some are hosted by Transport services. I refer to them as services here because the services hosted by different server roles has changed over different versions of Exchange, so it is often simpler to refer to services instead.

The most common clients you will encounter in the real world are Outlook (desktop), mobile devices or apps, and web browsers. But of course, there are many other types of clients that you will also see from time to time, such as POP/IMAP clients, SMTP clients, or custom-developed Exchange Web Services (EWS) clients.

Outlook

Outlook is the primary desktop application used to connect to Exchange mailboxes. The first thing to keep in mind is that not all versions of Outlook will work with every version of Exchange. Exchange 2013 is supported for Outlook 2007 SP3 or later, as long as the latest updates are also applied to Outlook. Exchange 2016 dropped Outlook 2007 support entirely and requires clients to run Outlook 2010 SP1 or later, again as long as the latest updates are also applied to Outlook.

For Mac users there is Outlook for Mac 2014, Outlook for Mac 2011, and Entourage 2008 (EWS), which are supported for both Exchange Server 2013 and 2016. The Mac clients connect to Exchange using Exchange Web Services (EWS), which is discussed later in this chapter. Outlook for Mac 2016 is available from Office 365 and can also be used to connect to an on-premises server.

Outlook for Windows connects to Exchange 2013 and Exchange 2016 using one of two protocols:

- RPC over HTTP (also known as Outlook Anywhere)
- MAPI over HTTP (also known as MAPI/HTTP), introduced in Exchange 2013 Service Pack 1

Both protocols connect to Exchange over HTTPS (TCP port 443). In fact, all Outlook connectivity to Exchange now occurs over HTTPS, with the exception of one Autodiscover lookup method that will try on port 80. This greatly simplifies the client port requirements for connecting to Exchange, which means simpler configurations for firewalls that sit on the network between clients and servers.

Autodiscover

Outlook configuration and connectivity relies heavily on Autodiscover. The Autodiscover service running on Exchange provides information to clients about how to connect to Exchange services. When a domain-joined Outlook client starts for the first time, an Active Directory lookup is used to determine the email attributes of the user, as well as locate an Autodiscover Service Connection Point (SCP) to query for more information. Each Exchange 2013 or earlier Client Access server, or every Exchange 2016 Mailbox server, registers an SCP using the server's fully qualified domain name. The administrator then needs to reconfigure the SCP for each server to the appropriate namespace for that site (all servers within a site should have the same SCP URL configured).

In the case of domain-joined clients, the Autodiscover SCP URL does not need to match the domain name of the user's primary SMTP address. If Autodiscover is successful using the SCP the Outlook profile will be automatically configured (Figure 7-1).

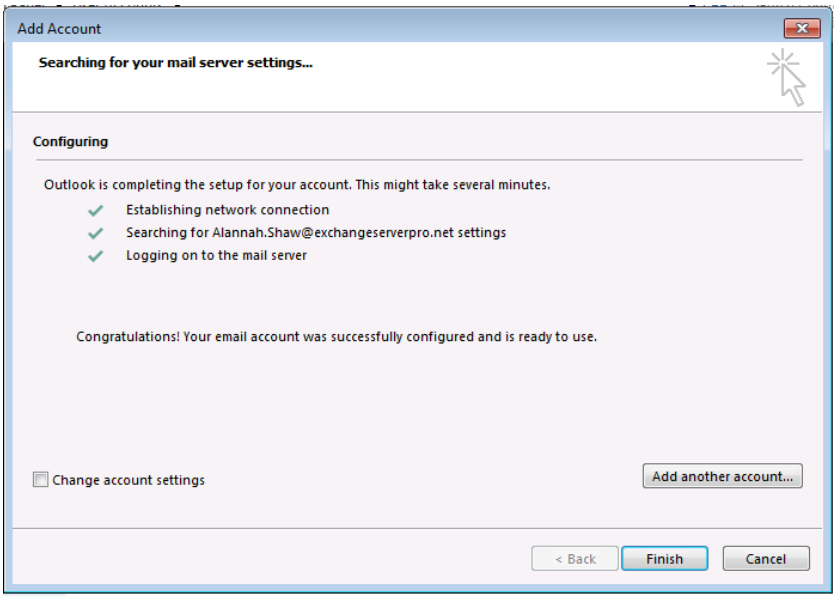


Figure 7-1: A successful Autodiscover and auto-configuration in Outlook 2013

If the Autodiscover SCP does not exist or is invalid in Active Directory, the Autodiscover process continues using the same sequence as clients that are not domain-joined, which is explained next.

Outlook clients running on non-domain joined computers still use Autodiscover. Instead of looking up the SCP in Active Directory, they use a sequence of DNS lookups against the domain name of the user's primary email address.

1. HTTPS root domain query (e.g. https://contoso.com/Autodiscover/Autodiscover.xml)
2. HTTPS Autodiscover domain query (e.g. https://autodiscover.contoso.com/Autodiscover/Autodiscover.xml)
3. Local XML file
4. HTTP redirect method (e.g. http://autodiscover.contoso.com/Autodiscover/Autodiscover.xml, expecting a redirect to a HTTPS URL)
5. SRV DNS record query (looking for an SRV record called _autodiscover._tcp.contoso.com)
6. Cached URL in the Outlook profile (if a previous Autodiscover URL was found and cached)

Although Outlook will stop searching for Autodiscover as soon as it succeeds with one of the options in the sequence above, it might (depending on the client version and whether a policy disables it) perform multiple lookups simultaneously. For example, Outlook 2013 will perform an Autodiscover SCP and root domain lookup at the same time, which can have unintended consequences if the root domain resolves to a server that has a HTTPS website running on it.

Outlook then authenticates to Autodiscover with the user's credentials, and posts a query to determine to which Exchange namespace it should connect to for that user's mailbox. This process of automatic configuration is dependent on several factors:

- The Autodiscover SCP, defined as the *AutodiscoverServiceInternalUri* of the Client Access server, is able to be resolved in DNS and is functioning
- The Outlook client is able to make a connection on port TCP 443 to the Autodiscover service

- The HTTPS session is able to be established, which relies on the SSL certificate passing validity checks. SSL certificates are discussed in Chapter 3.

Real World: The number one cause of Autodiscover failures I encounter in the field is the Autodiscover SCP not being changed from the default value. The number two cause is the Autodiscover SCP being set to a name that doesn't exist on the Exchange server's SSL certificate.

Outlook caches Autodiscover information in the user's profile path. You will find the file located in *C:\Users\<username>\AppData\Local\Microsoft\Outlook*, with a file name of *<GUID> – Autodiscover.xml* (Figure 7-2).

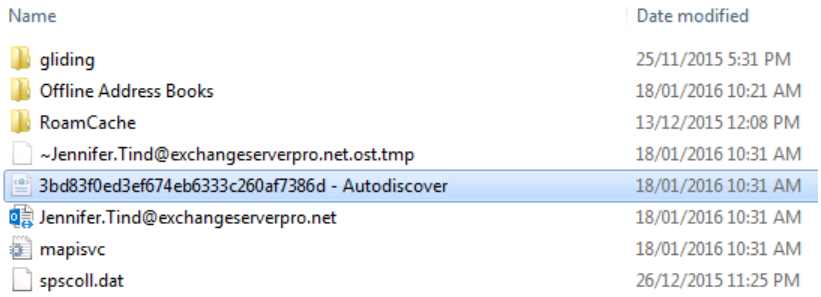


Figure 7-2: Autodiscover information cached in the user's profile

If you ever suspect that the client is using incorrect cached Autodiscover details, you can simply delete that file and restart Outlook. Autodiscover can be tested from within Outlook by holding the CTRL key on your keyboard and right-clicking on the Outlook icon in the system tray, then choosing **Test E-mail AutoConfiguration** (Figure 7-3).

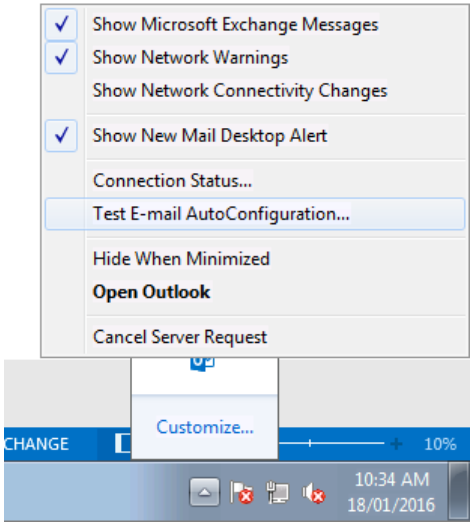


Figure 7-3: Accessing the email auto-configuration test in Outlook

Enter the email address and password for the account, de-select the two **Guessmart** options, and then click **Test** (Figure 7-4).

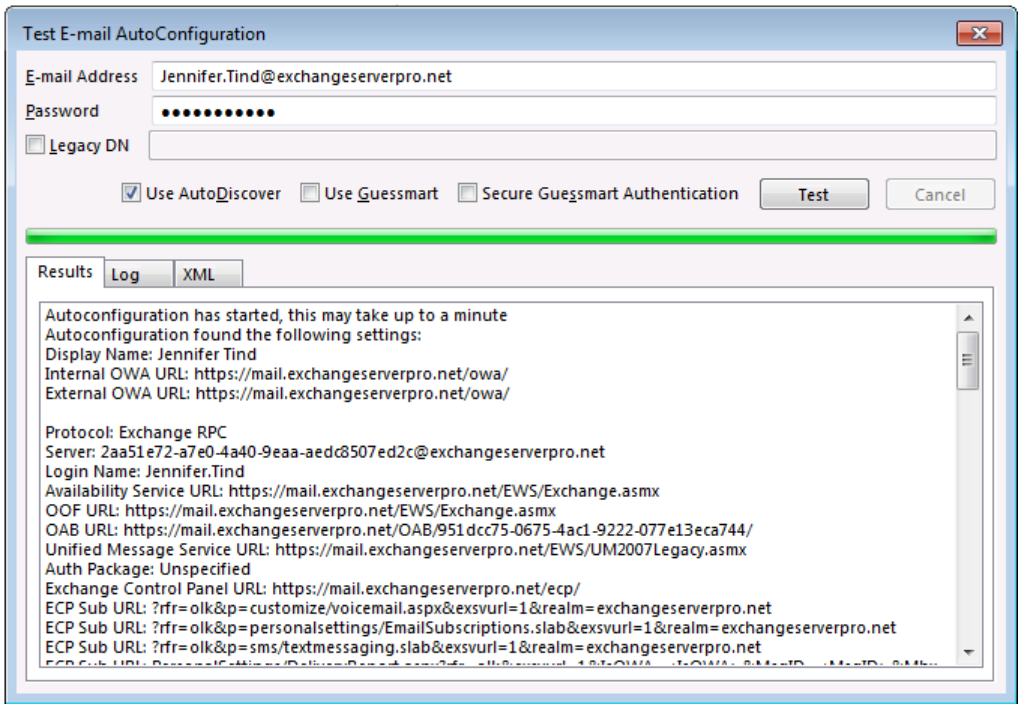


Figure 7-4: Auto-configuration test results in Outlook

The test should complete within a few seconds, and you can review the URLs that have been returned for each service to make sure they align with the URLs you've configured on your Client Access servers and SSL certificates. If they don't match your settings immediately, wait a little longer and try again in case there is simply a delay with Active Directory replication. These URLs can only come from Autodiscover. You can't override them or assign them manually for an Outlook profile, so it's important that Autodiscover works correctly.

Depending on your Autodiscover namespaces and DNS entries, Outlook may be redirected to a server name that it does not trust, and will display a warning to the end user (Figure 7-5). This warning may also appear if Outlook attempts to connect to the well-known URL of "autodiscover", even if you have a different namespace in use for Autodiscover in your environment.

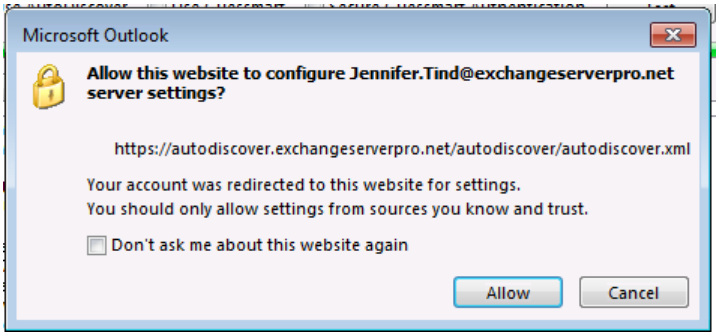


Figure 7-5: Outlook Autodiscover redirect warning dialog

Even though the user can click to allow the redirection, and suppress warnings for that server name in future, this is not a good user experience. Fortunately, the warning can be avoided by adding your Autodiscover namespace to Outlook's list of trusted names. The list of redirect servers is stored in the registry in one of two locations:

- HKEY_CURRENT_USER\Software\Policies\Microsoft\Office\xx.0\Outlook\Autodiscover\RedirectServers
- HKEY_CURRENT_USER\Software\Microsoft\Office\xx.0\Outlook\Autodiscover\RedirectServers

In the registry paths shown above "xx" is the version of Outlook, for example 15.0 for Outlook 2013. Add string values for any additional server names that you want to suppress the redirect message for (Figure 7-6).

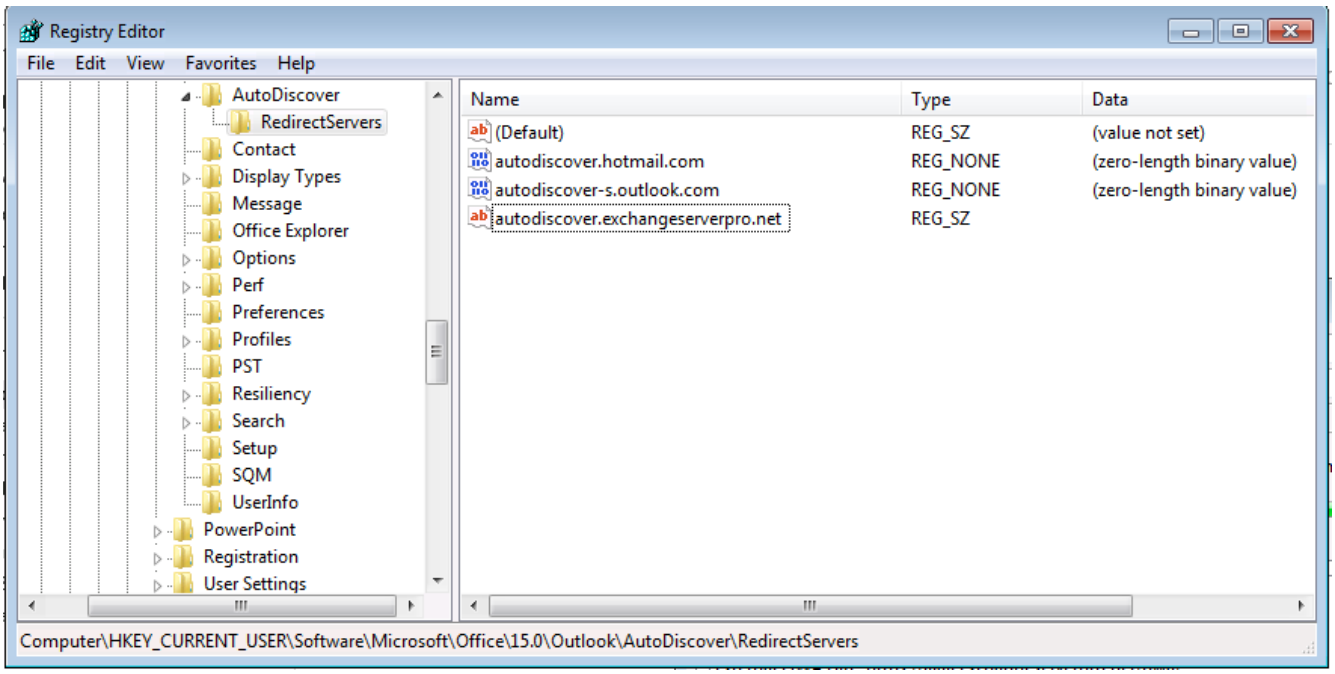


Figure 7-6: Registry entries that control which servers are trusted for Autodiscover

To deploy the registry key to multiple computers it is easier to use a Group Policy.

Outlook Connectivity

After successfully retrieving Autodiscover information and configuring the Outlook profile, Outlook then connects to Exchange using either Outlook Anywhere or MAPI/HTTP. Again, this is an SSL-encrypted connection over HTTPS, so SSL certificates are important at this stage as well. If there are any SSL certificate validity problems, then Outlook will fail to connect (Figure 7-7).

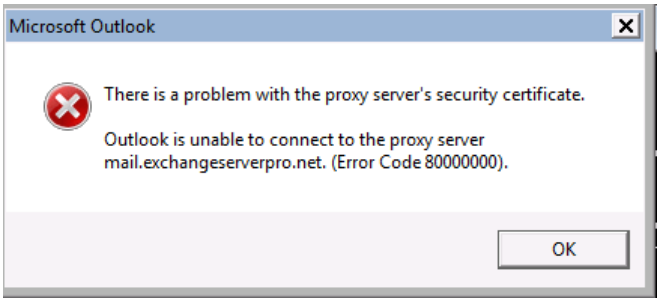


Figure 7-7: Outlook fails to connect when a certificate problem is detected

With two available protocols for Outlook connections (Outlook Anywhere, and MAPI/HTTP), you may be wondering how you can tell which one is being used by an Outlook client. First, you can check whether MAPI/HTTP is enabled for the Exchange organization.

```
[PS] C:\> Get-OrganizationConfig | select *mapi*

MapiHttpEnabled : False
```

From the client-side you can also check the Outlook Connection Status dialog, available by right-clicking the Outlook icon in the system tray. The **Protocol** column will show RPC/HTTP for Outlook Anywhere (Figure 7-8).

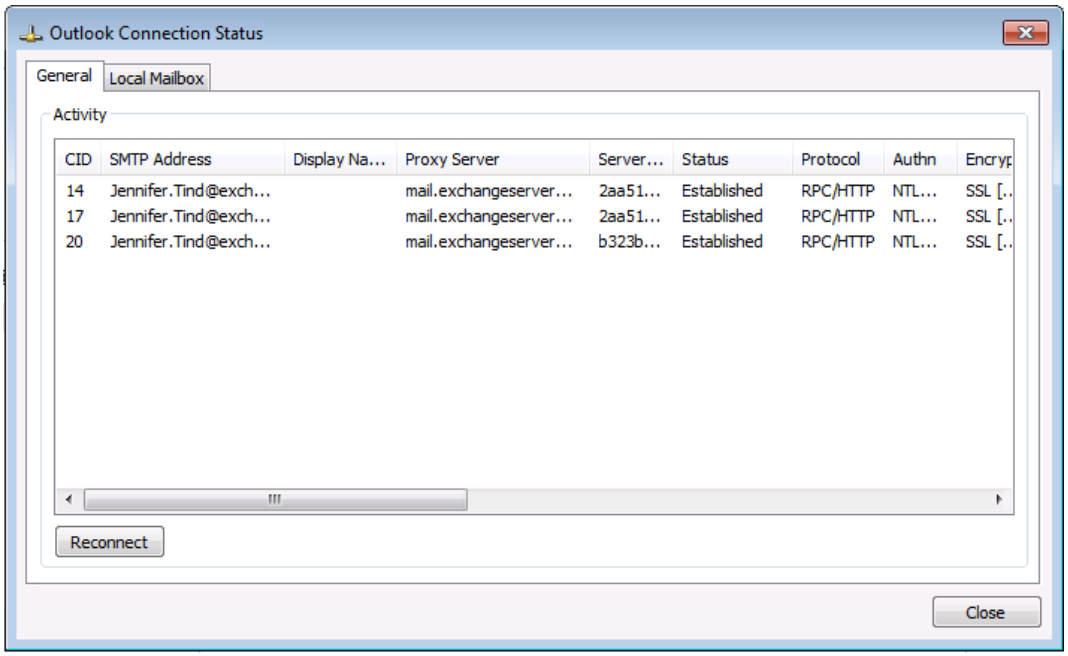


Figure 7-8: An Outlook client connecting using RPC/HTTP

The MAPI/HTTP configuration for the organization can be enabled by running *Set-OrganizationConfig*.

```
[PS] C:\> Set-OrganizationConfig -MapiHttpEnabled $true
```

Supported clients (Outlook 2013 SP1 or later) will not instantly switch to using MAPI/HTTP. You are likely to see some delay before the change becomes affected, but once it is, the **Protocol** column of the Outlook Connection Status dialog will display "HTTP" instead of "RPC/HTTP".

That is, unless the client has been disabled for MAPI/HTTP in the registry. This setting is located in *HKEY_CURRENT_USER\Software\Microsoft\Exchange*, in a DWORD value named *MapiHttpDisabled*. If that DWORD value is present, and is set to 1, then MAPI/HTTP will be disabled for that client.

Outlook Web App/Outlook on the web

Exchange has a web-based email client that has gone by several names over the years:

- Outlook Web Access (OWA) in Exchange 2007
- Outlook Web App (OWA) in Exchange 2010 and 2013
- Outlook on the web in Exchange 2016 (and Office 365)

For the sake of simplicity, I'll refer to it as OWA. Web browsers can connect to OWA assuming the following conditions:

- The end user knows the OWA URL to connect to. The OWA URL for on-premises Exchange is not automatically discovered by web browsers using Autodiscover.
- The OWA URL can be resolved in DNS.
- The OWA URL can be accessed on HTTPS (TCP port 443).
- The HTTPS session can be established.
- OWA is enabled for the end-user (it's enabled by default)

The most common issues encountered with OWA are:

- SSL certificate problems. If the SSL certificate fails a validity check, then the web browser will display a warning to the user. SSL certificates are discussed in Chapter 3.
- Login issues due to incorrect username syntax.
- Mailbox access issues during co-existence due to incorrect OWA namespace configuration, or load-balancing.
- Mailbox access issues due to Active Directory permissions.

OWA Login

Several methods can be used to authenticate a user connection with OWA. The recommended logon format is *User principal name (UPN)*, and best practice is to match a user's UPN to their primary SMTP address (Figure 7-9). In effect this makes it seem to the user like they log on with their email address, which is simpler for them to remember than any other method.

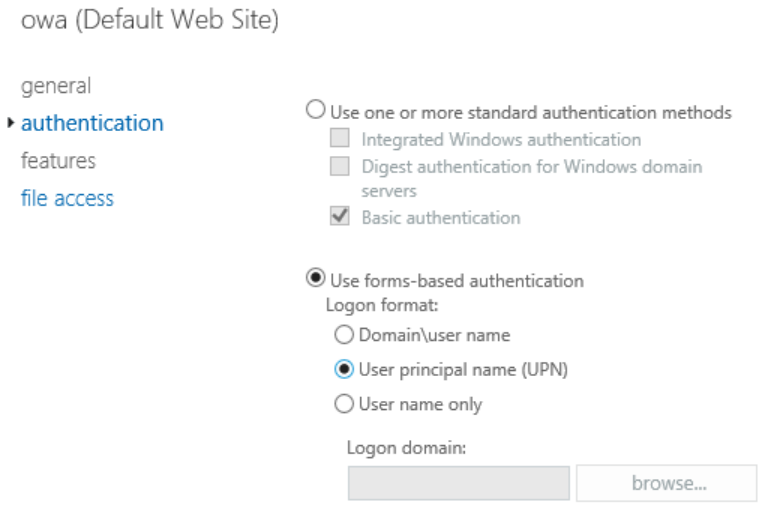


Figure 7-9: OWA authentication options

Using one of the forms-based authentication methods for OWA logon means that users see a user-friendly logon page when they access the OWA URL with a browser. In fact, when the OWA logon format is set to use the UPN the login page will display "Email address" instead of "User name" (Figure 7-10), to make it even friendlier for the user to understand what to enter in the form.

Figure 7-10: The OWA login page with the user-friendly "email address" prompt

Standard authentication methods of *Integrated Windows*, *Digest*, and *Basic* are also available. Those options present a less friendly dialog box. The exception is when *Integrated Windows* is used in combination with adding the OWA URL to the Intranet Zone of the user's Internet Explorer security settings. Under those conditions the user will be automatically logged on to OWA with the domain credentials that they have logged on to their computer with.

When multiple servers exist it is recommended to configure the OWA login settings to be the same, because mismatched OWA logon settings on Exchange servers will cause problems for end users. The users will either be unable to login, or will have their sessions unexpectedly terminated.

You can review the OWA virtual directory settings using the *Get-OWAVirtualDirectory* cmdlet.

```
[PS] C:\> Get-OwaVirtualDirectory | select Server,FormsAuthentication,LogonFormat

Server     FormsAuthentication   LogonFormat
------     -------------------   -----------
EX2013SRV1                True PrincipalName
EX2010SRV1                True    FullDomain
EX2016SRV1                True      UserName
EX2016SRV2                True    FullDomain
```

In the example above you can see that the logon formats are different for each virtual directory. To align them all with the desired setting, use *Set-OWAVirtualDirectory*. For example, to configure all of the servers in the example above to use the UPN logon format, the following command is used.

```
[PS] C:\> Get-OwaVirtualDirectory | Set-OwaVirtualDirectory -FormsAuthentication $true -LogonFormat
PrincipalName
```

Because the *Get-OWAVirtualDirectory* and *Set-OWAVirtualDirectory* cmdlets need to make RPC calls to IIS on each server for this configuration, you'll notice that the command runs very slowly, particularly in larger environments.

Note: When you modify the OWA logon format there are some warnings thrown by the PowerShell cmdlets. On each server IIS must be restarted by running *IISreset*, and the ECP virtual directory also needs to be reconfigured to match the OWA logon format by using *Set-ECPVirtualDirectory*.

OWA Namespaces

Each Exchange server that hosts an OWA virtual directory has two URLs configured; the internal URL, and the external URL. You can view the namespaces configured on OWA virtual directories by running the *Get-OWAVirtualDirectory* cmdlet.

```
[PS] C:\> Get-OwaVirtualDirectory -ADPropertiesOnly | Select Server,InternalURL,ExternalURL

Server     InternalUrl                          ExternalUrl
------     -----------                          -----------
EX2013SRV1 https://mail.exchangeserverpro.net/owa https://mail.exchangeserverpro.net/owa
EX2010SRV1 https://mail.exchangeserverpro.net/owa https://mail.exchangeserverpro.net/owa
EX2016SRV1 https://mail.exchangeserverpro.net/owa https://mail.exchangeserverpro.net/owa
EX2016SRV2 https://mail.exchangeserverpro.net/owa https://mail.exchangeserverpro.net/owa
```

As mentioned earlier, browsers do not use Autodiscover to find the OWA URL. In fact, the user can type anything into their browser, even an IP address, and as long as it resolves to the Exchange server they will connect to OWA. But that doesn't mean that URLs should not be configured on the OWA virtual directories. There are still two ways in which the configured OWA URL comes into play. The first is the in the Outlook

backstage view, where the OWA URL is exposed next to the users account settings options (Figure 7-11). Outlook clients do retrieve the OWA URL using Autodiscover, even though web browsers don't.

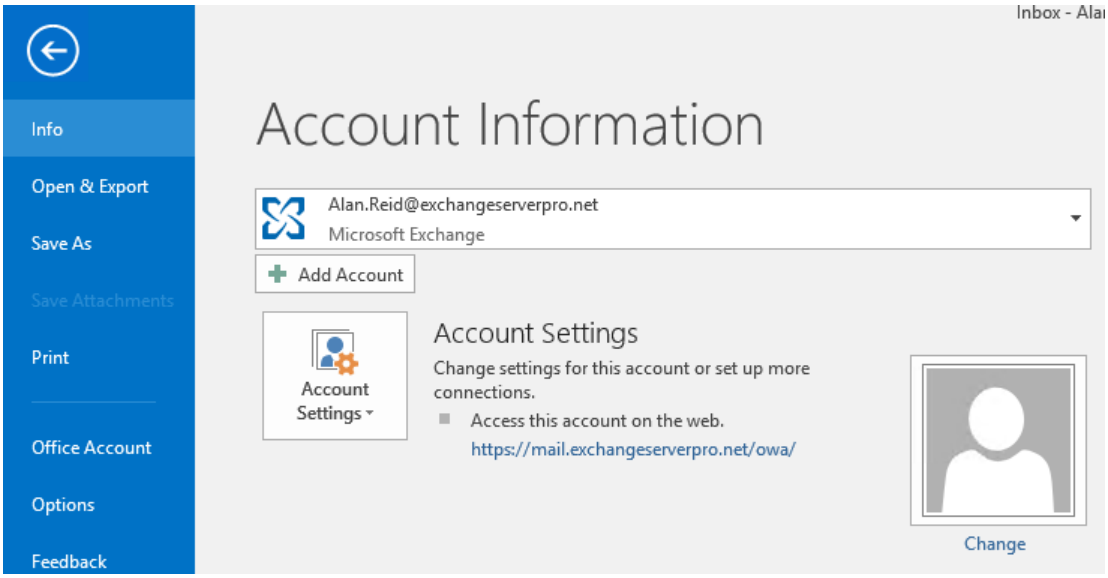


Figure 7-11: The OWA URL visible in Outlook's backstage view

The second is during some co-existence scenarios. In particular, when Exchange 2013 and 2007 are in co-existence, OWA uses a redirect to a legacy URL when Exchange 2007 mailbox users connect to the Exchange 2013 server (Figure 7-12).

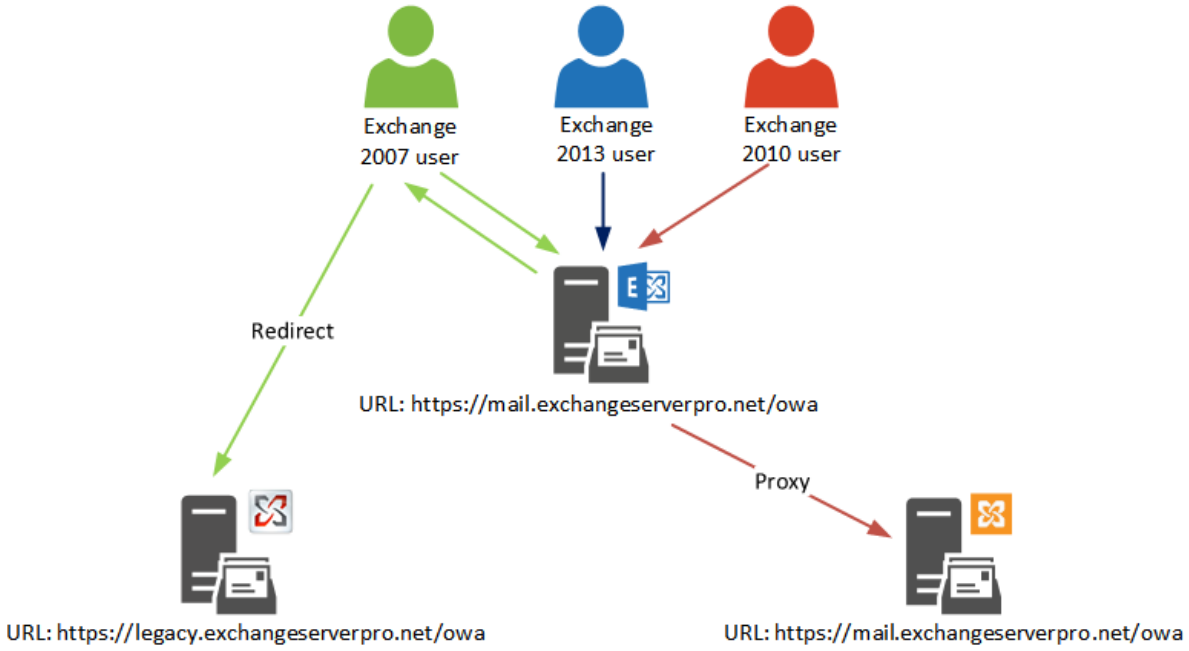


Figure 7-12: OWA redirection during Exchange 2007 and 2013 co-existence

Real World: Even though you can theoretically connect to OWA using the wrong URL, or even using the IP address, if the connection goes through any type of application-aware load balancer or proxy server that is checking for specific host names, then the connection will fail.

OWA Active Directory Permissions

When a user logs on to OWA the Exchange server computer account needs to access the user object in Active Directory to read the mail attributes. If the Exchange server computer account doesn't have read access to those attributes, then the OWA login will fail. The most obvious sign of this problem is the user seeing an error message similar that shown in Figure 7-13. Active Directory permissions are discussed in a later section of this chapter.

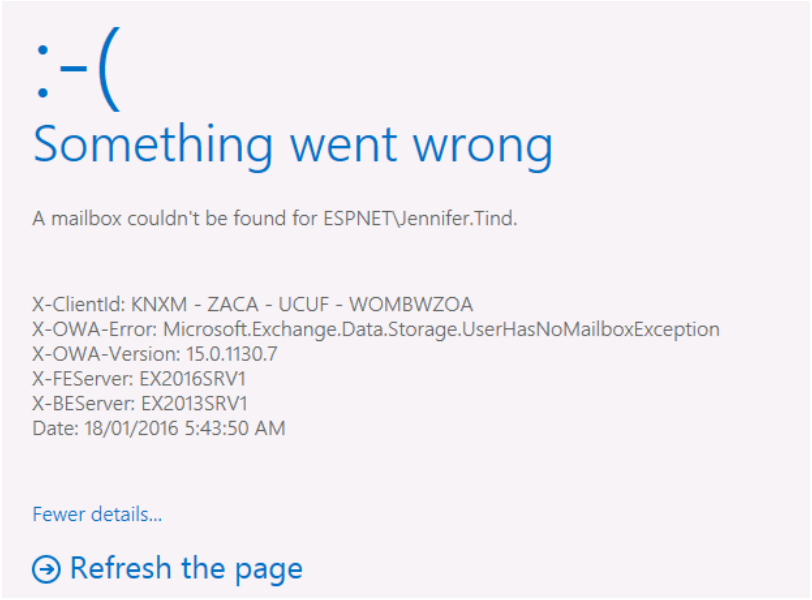


Figure 7-13: OWA login failure

ActiveSync

Exchange ActiveSync (EAS) is the protocol used by mobile devices and applications to access email, calendar, and other items in Exchange mailboxes. Troubleshooting mobile devices can be a complex task, because there are so many factors that influence whether a mobile device will be able to successfully connect to Exchange. For example, these are just some of the common factors that influence mobile connectivity to Exchange:

- Mobile networks
- DNS
- Autodiscover
- Firewalls
- TCP session timeouts
- Reverse proxies and load balancers
- SSL certificates
- Username format in Active Directory
- EAS policies
- EAS mailbox settings
- Active Directory permissions
- Exchange server health

ActiveSync Connectivity

The most obvious requirement for a mobile device is network connectivity, and a route between the device and the Exchange server. Mobile devices typically connect over the internet, and EAS connectivity from the

mobile device to Exchange occurs over HTTPS (TCP 443) in order to encrypt and protect the network traffic between device and server (Figure 7-14).

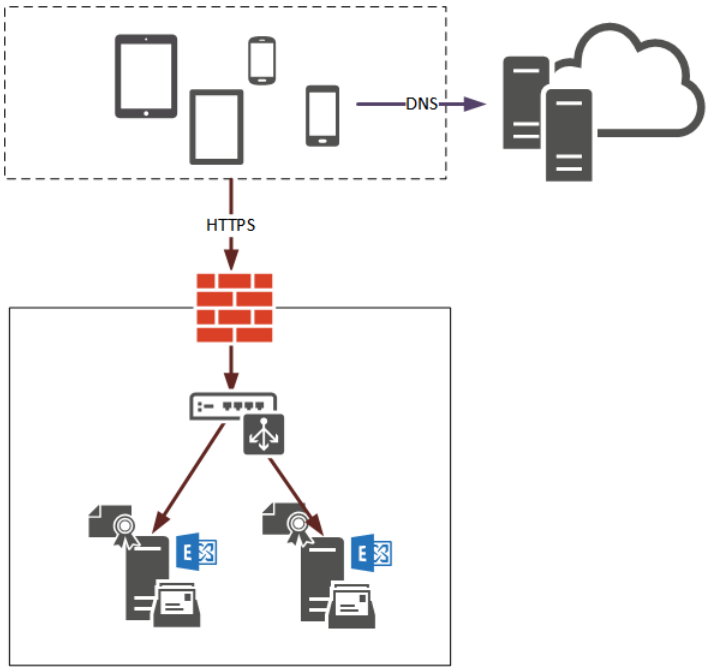


Figure 7-14: Mobile device connectivity

Before the device can connect to Exchange, it first uses Autodiscover to locate the ActiveSync namespace to which to connect. To enable the Autodiscover process, a series of DNS records are created in the public DNS zone for the domain. One or more of those DNS records will be used by the mobile device for Autodiscover. Different devices and applications can have a different Autodiscover lookup order, but generally speaking the order will be:

1. HTTPS root domain query (e.g. https://contoso.com/Autodiscover/Autodiscover.xml)
2. HTTPS Autodiscover domain query (e.g. https://autodiscover.contoso.com/Autodiscover/Autodiscover.xml)
3. HTTP redirect method (e.g. http://autodiscover.contoso.com/Autodiscover/Autodiscover.xml, expecting a redirect to a HTTPS URL)
4. SRV DNS record query (looking for an SRV record called _autodiscover._tcp.contoso.com)

Real World: A recommended practice is to implement DNS records for all of the Autodiscover lookups that different devices and applications might perform, for maximum compatibility. However, this means you should always make sure that you update all of the different record types any time there is a change, such as a migration or hybrid scenario. If you forget to update one of the record types, you can expect mobile devices, as well as other applications such as Skype for Business or Outlook, to fail in unexpected ways.

If the Autodiscover endpoint is located, and the user's credentials are correct, then Autodiscover will return the ActiveSync namespace that is configured on the Client Access services in the Active Directory site where the user's mailbox is hosted. The ActiveSync namespace can be seen by running the *Get-ActiveSyncVirtualDirectory* cmdlet.

```
[PS] C:\> Get-ActiveSyncVirtualDirectory -ADPropertiesOnly | Select Server,InternalUrl,ExternalUrl

Server      : EX2013SRV1
InternalUrl : https://mail.exchangeserverpro.net/Microsoft-Server-ActiveSync
ExternalUrl : https://mail.exchangeserverpro.net/Microsoft-Server-ActiveSync
```

```
Server      : EX2010SRV1
InternalUrl : https://mail.exchangeserverpro.net/Microsoft-Server-ActiveSync
ExternalUrl : https://mail.exchangeserverpro.net/Microsoft-Server-ActiveSync

Server      : EX2016SRV1
InternalUrl : https://mail.exchangeserverpro.net/Microsoft-Server-ActiveSync
ExternalUrl : https://mail.exchangeserverpro.net/Microsoft-Server-ActiveSync

Server      : EX2016SRV2
InternalUrl : https://mail.exchangeserverpro.net/Microsoft-Server-ActiveSync
ExternalUrl : https://mail.exchangeserverpro.net/Microsoft-Server-ActiveSync
```

The mobile device or application then looks up the namespace in DNS, makes a connection over HTTPS to the namespace, authenticates the user, and then begins to perform folder sync and other operations. All of this relies on DNS, firewall access, load balancers (if they are deployed in the environment), certificates, and authentication. The connection will also depend on the Active Directory security on the user object, similar to what was described earlier in this chapter for OWA access to mailboxes. Active Directory permissions are discussed in a later section of this chapter.

Note: The PowerShell output above is an example of using split DNS. Both the internal and external URLs are the same, but resolve to different internal and external IP addresses respectively by hosting an internal DNS zone for internal clients to use, and an external DNS zone for clients on the internet to use.

Allow/Block/Quarantine for ActiveSync

When a mobile device or application makes an ActiveSync connection to an Exchange mailbox, it goes through a process known as Allow/Block/Quarantine (ABQ), which assesses the following criteria:

1. Is the mobile device authenticated?
2. Is the user enabled for ActiveSync?
3. Does the device comply with the ActiveSync mailbox policy in effect for that user?
4. Does the user have a personal exemption that blocks the mobile device?
5. Does the user have a personal exemption that allows the mobile device?
6. Is the device blocked by a matching device access rule?
7. Is the device quarantined by a matching device access rule?
8. Is the device allowed by a matching device access rule?
9. Is the device allowed, blocked, or quarantined by the default access level specified for the organization?

The ABQ process can be represented by the workflow diagram in Figure 7-15.

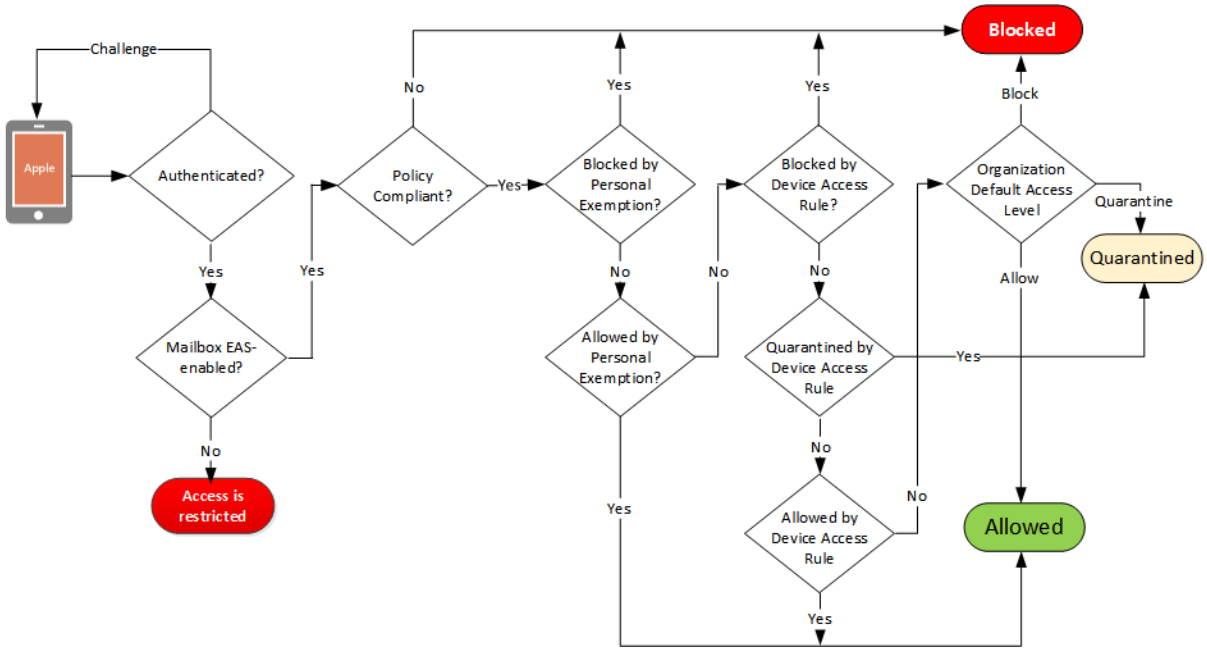


Figure 7-15: The ActiveSync ABQ process

At any stage during the ABQ workflow a decision can be made by the server to allow, block or quarantine the device. For example, if a device is not policy compliant, then the block decision is made at that stage of the process, and no further criteria are checked.

Authentication is the first requirement. If the user can't authenticate with their credentials, then they will not be able to access their mailbox. Credentials are typically their email address and password, if you've configured users with User Principal Names (UPNs) that match their primary SMTP address, which is the recommended practice. If the credentials are incorrect, or the account is disabled or locked, the user can't log on from the mobile device or application.

Real World: Disabling a user account will not prevent them from connecting if they already have an established ActiveSync connection, due to caching of information by Internet Information Services (IIS) on the Exchange server. It may take several hours for the user to be unable to connect, unless IIS is restarted, which is disruptive for other users also connected to that server. If you need to immediately block a user from connecting to ActiveSync, then you should disable the protocol on their mailbox.

The ActiveSync protocol is enabled by default on all mailboxes, and can be disabled on a per-mailbox basis. You can see the current protocol status for a user's mailbox by running the *Get-CASMailbox* cmdlet.

```
[PS] C:\> Get-CASMailbox alan.reid@exchangeserverpro.net | select ActiveSyncEnabled

ActiveSyncEnabled : True
```

To disable ActiveSync, run *Set-CASMailbox*.

```
[PS] C:\> Set-CASMailbox alan.reid@exchangeserverpro.net -ActiveSyncEnabled $false
```

The mobile device or application must also be policy compliant. Each mailbox user is assigned a mobile device mailbox policy in Exchange that defines the required security configuration for any devices that the user wants to use to connect to their mailbox.

The mobile device mailbox policies for the Exchange organization can be seen by running *Get-MobileDevicePolicy*.

```
[PS] C:\> Get- MobileDeviceMailboxPolicy
```

The PowerShell output for mobile device mailbox policies is sometimes difficult to interpret, so a clearer view can be seen by using the Exchange Admin Center to view the properties of each policy (Figure 7-16).

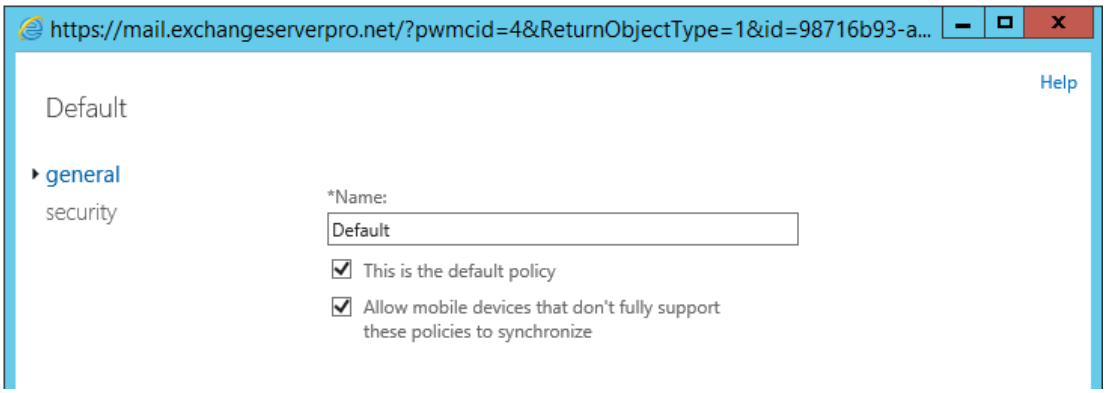


Figure 7-16: The default mobile device mailbox policy

You can also view the mobile device mailbox policy assigned to a user by running *Get-CASMailbox*.

```
[PS] C:\> Get-CASMailbox alan.reid@exchangeserverpro.net | Select ActiveSyncMailboxPolicy*

ActiveSyncMailboxPolicy            : Default
ActiveSyncMailboxPolicyIsDefaulted : True
```

New mailboxes are automatically assigned to the mobile device mailbox policy that is marked as the default policy.

Note: In the example above the policy name is "Default", and Alan Reid is configured to use the default policy (which happens to be named "Default"). However, if another policy was set as the default later, Alan would begin using the new default policy, instead of the policy named "Default". To avoid confusion with this situation it is best to leave the policy named "Default" as the default, configured with the policy settings you want to be the defaults for new mailbox users, and then create additional non-default policies with different names for specific needs.

If the device settings comply with the policy for that mailbox, then the ABQ process next looks at whether the device ID has been explicitly blocked or allowed for that mailbox user, in that order. The *Get-CASMailbox* cmdlet shows us which device IDs fall into either of those categories. In this example, Vik Kirby has a device ID that is allowed, so it will skip all subsequent ABQ checks and be allowed to connect to the mailbox.

```
[PS] C:\ >Get-CASMailbox Vik.Kirby | select ActiveSyncAllowed*,ActiveSyncBlocked*

ActiveSyncAllowedDeviceIDs : {F04016EDD8F2DD3BD6A9DA5137583C5A}
ActiveSyncBlockedDeviceIDs : {}
```

Personal block or allow exemptions of that nature are only created by administrator action. An administrator can pre-populate the device ID into the block or allow list if the device ID is known beforehand, otherwise they can do it after the user has tried to connect the device (and perhaps been quarantined, as well discuss shortly).

If no block or allow exemptions exist, then ABQ next looks at device access rules. Device access rules are created by administrators to either block, quarantine, or allow devices and applications that meet the specific criteria of the rule. The available criteria are:

- Device Type (or Family)
- Device Model
- Device OS

- User Agent

Each device access rule can have a single characteristic assigned, which makes them very precise, but also quite cumbersome to deal with if there are dozens or even hundreds of device characteristics that you need to build rules for. The device access rules can be seen by running *Get-ActiveSyncDeviceAccessRule*.

```
[PS] C:\> Get-ActiveSyncDeviceAccessRule

QueryString       : NastyOS
Characteristic    : DeviceOS
AccessLevel       : Block
Name              : NastyOS (DeviceOS)
AdminDisplayName  :
ExchangeVersion   : 0.10 (14.0.100.0)
DistinguishedName : CN=NastyOS (DeviceOS),CN=Mobile Mailbox Settings,CN=Exchange Server
Pro,CN=Microsoft
                    Exchange,CN=Services,CN=Configuration,DC=exchangeserverpro,DC=net
Identity          : NastyOS (DeviceOS)
Guid              : 9065415f-0602-42e0-9b36-4c813e1b132e
ObjectCategory    : exchangeserverpro.net/Configuration/Schema/ms-Exch-Device-Access-Rule
ObjectClass       : {top, msExchDeviceAccessRule}
WhenChanged       : 2/5/2016 12:39:51 AM
WhenCreated       : 2/5/2016 12:39:51 AM
WhenChangedUTC    : 2/4/2016 2:39:51 PM
WhenCreatedUTC    : 2/4/2016 2:39:51 PM
OrganizationId    :
Id                : NastyOS (DeviceOS)
OriginatingServer : S1DC1.exchangeserverpro.net
IsValid           : True
ObjectState       : Unchanged
```

When multiple device access rules match a device they are assessed in the following order:

1. Block rules
2. Quarantine rules
3. Allow rules

In effect this means that the most restrictive rule (a block) will override a less restrictive, or non-restrictive rule. That just makes good security sense.

If no device access rules match the device, then ABQ finishes by applying the default access level for the organization. There are three levels as you would expect; allow, block, or quarantine. You can view the default access level by running the *Get-ActiveSyncOrganizationSettings* cmdlet.

```
[PS] C:\> Get-ActiveSyncOrganizationSettings | Select DefaultAccessLevel

DefaultAccessLevel : Allow
```

With so many different stages of the ABQ process that can cause a device to be blocked, quarantined, or allowed, it may seem to be impossible to troubleshoot. However, there are a few techniques that will make your job as an Exchange administrator a lot easier.

The first is using the *Get-MobileDevice* cmdlet to look at the device access state, and the device access state reason. This information will reveal which devices are blocked, quarantined, or allowed, and why they are in that state.

```
[PS] C:\> Get-MobileDevice | Select DeviceId,DeviceType,DeviceModel,DeviceAccessState*

DeviceId               : Appl87941C1N3NS
DeviceType             : iPhone
DeviceModel            : iPhone2C1
DeviceAccessState      : Blocked
DeviceAccessStateReason : Policy
```

```
DeviceId                : Appl87941C1N3NS
DeviceType              : iPhone
DeviceModel             : iPhone2C1
DeviceAccessState       : Blocked
DeviceAccessStateReason : Individual

DeviceId                : 704294541
DeviceType              : TestActiveSyncConnectivity
DeviceModel             : TestActiveSyncConnectivity
DeviceAccessState       : Allowed
DeviceAccessStateReason : Individual

DeviceId                : ApplDLXH8DELDVGJ
DeviceType              : iPad
DeviceModel             : iPad3C3
DeviceAccessState       : Allowed
DeviceAccessStateReason : Global
```

In the output above you can see that two devices are blocked. One is blocked because of a policy (it doesn't meet the requirements of the mobile device mailbox policy for that user), and the other is blocked by a personal exemption. Similarly, two mobile devices are allowed. One is allowed by a personal exemption, and the other is allowed because of the global (or default) access level for the organization.

The second useful technique is to run the Exchange Remote Connectivity Analyzer (ExRCA), and perform its ActiveSync test. The ExRCA test will simulate a real mobile device on the internet, giving you a clear view of the complete, end to end connectivity and ABQ process for mobile devices and applications that are connecting to your Exchange servers. If you can't see a mobile device associated with a mailbox user in Exchange at all, then there's a good chance that a connectivity problem is occurring before the ABQ process can start. The ExRCA test will help you to identify those types of problems.

Real World: It's also useful to keep some spare mobile devices around for testing with different mobile operating systems and applications.

Exchange Web Services

Exchange Web Services (EWS) is an API that enables client applications to communicate with Exchange to access information about mailboxes or their contents. Applications can use EWS to retrieve information from Exchange services, or to interact with data in Exchange mailboxes. For example, an EWS application can retrieve information about calendar items from a room mailbox to determine which items might have an organizer who no longer works for the company.

EWS is used by Microsoft Outlook for calendar free/busy information, Out of Office settings, calendar sharing, and other features such as MailTips. Mac clients rely entirely on EWS for all communications. When EWS is not working, those features will stop working as well, which end users are likely to notice as symptoms such as not seeing the availability of other users they are inviting to a meeting.

EWS uses its own namespace which can be seen by running the *Get-WebServicesVirtualDirectory* cmdlet.

```
[PS] C:\> Get-WebServicesVirtualDirectory -ADPropertiesOnly | Select Server,InternalUrl,ExternalUrl

Server      : EX2013SRV1
InternalUrl : https://mail.exchangeserverpro.net/EWS/Exchange.asmx
ExternalUrl : https://mail.exchangeserverpro.net/EWS/Exchange.asmx

Server      : EX2010SRV1
InternalUrl : https://mail.exchangeserverpro.net/EWS/Exchange.asmx
ExternalUrl : https://mail.exchangeserverpro.net/EWS/Exchange.asmx
```

```
Server      : EX2016SRV1
InternalUrl : https://mail.exchangeserverpro.net/EWS/Exchange.asmx
ExternalUrl : https://mail.exchangeserverpro.net/EWS/Exchange.asmx

Server      : EX2016SRV2
InternalUrl : https://mail.exchangeserverpro.net/EWS/Exchange.asmx
ExternalUrl : https://mail.exchangeserverpro.net/EWS/Exchange.asmx
```

As long as the EWS namespace is resolvable in DNS, can be reached over the network by the client, and the server is using a valid and trusted SSL certificate, then not many things can go wrong with EWS. Remember that EWS settings can only come from Autodiscover, so the EWS namespace can only be located when Autodiscover is configured correctly. However, it is also possible that applications will be blocked from accessing EWS.

There are multiple controls in place for allowing or blocking EWS. At the organization level, EWS can be configured to either enforce a block list (which will block any applications listed in the block list), or enforce an allow list (which will block any application except those that are listed in the allow list).

Real World: In June 2013, LinkedIn was found to have implemented a feature that invited users to enter their corporate email credentials on the LinkedIn website. LinkedIn then connected to the person's mailbox and scraped it for email addresses to suggest them as possible contacts that should be invited to connect on LinkedIn. This connection used EWS to access Exchange server mailboxes.

The organization configuration can be viewed by running the Get-OrganizationConfig cmdlet. By default, there is no policy set and nothing is explicitly allowed or blocked, which in effect means that any application can access EWS.

```
[PS] C:\> Get-OrganizationConfig | select Ews*

EwsAllowEntourage          :
EwsAllowList               :
EwsAllowMacOutlook         :
EwsAllowOutlook            :
EwsApplicationAccessPolicy :
EwsBlockList               :
EwsEnabled                 :
```

To block an application from access EWS the EWS application policy must first be set, and then the allow list or block list populated with the user agents of applications you want to control. Using the example of LinkedIn as mentioned above, to block the LinkedIn agent you would run the following commands.

```
PS C:\> Set-OrganizationConfig -EwsApplicationAccessPolicy EnforceBlockList

PS C:\> Set-OrganizationConfig -EwsBlockList @{add='LinkedInEWS'}
```

EWS can also be enabled or disabled on a per-mailbox basis, and individual mailboxes can have their own EWS application access policy, block list, or allow list.

```
[PS] C:\> Get-CASMailbox alan.reid | select Ews*

EwsEnabled                 :
EwsAllowOutlook            :
EwsAllowMacOutlook         :
EwsAllowEntourage          :
EwsApplicationAccessPolicy :
EwsAllowList               :
EwsBlockList               :
```

The same steps are used to allow or block EWS applications for mailboxes as are used for the organization-wide configuration, replacing *Set-OrganizationConfig* with *Set-CASMailbox*.

Note: In the examples above you'll notice that some specific applications such as Outlook, Outlook for Mac, and Entourage have their own individual EWS access controls that are separate to the general EWS application access policy. These properties use Boolean values – such as $true or $false.

POP and IMAP

Post Office Protocol (POP) and Internet Mail Access Protocol (IMAP) are older mail protocols that are not as commonly used today as the other client protocols discussed in this chapter. In fact, the POP and IMAP services are disabled by default on newly installed Exchange servers. POP and IMAP are independent protocols and services, and you do not need to enable both of them to be able to use one of them. However, due to the similarities in configuration and troubleshooting of POP and IMAP, this section will generally refer to both of them together.

POP and IMAP clients can connect to mailboxes as long as:

- The end user knows the POP/IMAP server name to connect to. The POP/IMAP server name is not automatically discovered using Autodiscover.
- The POP/IMAP server name can be resolved in DNS.
- The POP/IMAP services are running (they are not enabled by default)
- The server can be accessed on the correct network ports (the default ports are 110 and 995 for POP and secure POP, 143 and 993 for IMAP and secure IMAP).
- The POP/IMAP protocol is enabled for the mailbox (they are both enabled by default).
- The user is able to securely authenticate with the server.

One of the first tasks for an administrator who needs to make POP or IMAP available is to start those services on the Exchange server, and enable them for automatic start up for future restarts.

There are two POP services, and two IMAP services that need to be enabled. For POP the service names are:

- Microsoft Exchange POP3 (MSExchangePOP3)
- Microsoft Exchange POP3 Backend (MSExchangePOP3BE)

For IMAP the services are:

- Microsoft Exchange IMAP4 (MSExchangeIMAP4)
- Microsoft Exchange IMAP4 Backend (MSExchangeIMAP4BE)

For Exchange Server 2013 the front end service (e.g. Microsoft Exchange POP3) runs on the Client Access server role, while the backend service runs on the Mailbox server role. If both roles are installed on the same server, then both services exist on the server. For Exchange Server 2016, both services always run on the same server.

Note: In earlier versions of Exchange only one service existed for each of POP and IMAP, and there were no backend services for either of them.

The POP or IMAP services need to be enabled on any server that the client will be connecting to for POP or IMAP, and on any server that can host an active database copy for the mailbox users that will be connecting.

Although their use is less common, when POP or IMAP are used it's important that clients connect securely. This is because the POP and IMAP protocols both pass all communications, including credentials, in clear text. POP and IMAP connections are only secure if they occur over an SSL/TLS encrypted connection. Unfortunately, it is the secure logon process that causes most connectivity issues for POP and IMAP.

POP and IMAP can each be independently assigned an SSL certificate. They do not need to use the same SSL certificate as the HTTPS services. By default, POP and IMAP use the self-signed certificate that is installed on the Exchange server during setup. The self-signed certificate will not be trusted by clients connecting to POP or IMAP, and should be replaced with a valid third-party SSL certificate. To change the POP and IMAP services to use a different certificate that is a SAN or single name certificate, run the *Enable-ExchangeCertificate* cmdlet.

```
[PS] C:\> Get-ExchangeCertificate -Thumbprint 4E26BD30B489DA13167AD5C6FFEB26FDDB63EE1D | Enable-
ExchangeCertificate -Services POP,IMAP
```

If the certificate you want to use is a wildcard certificate, then you need to use *Set-POPSettings* and *Set-IMAPSettings* instead. For example, to set the POP service to use a full-qualified domain name of *mail.exchangeserverpro.net*, we would run the following commands.

```
[PS] C:\> Set-PopSettings -Server EX2016SRV1 -X509CertificateName mail.exchangeserverpro.net

WARNING: Changes to POP3 settings will only take effect after all Microsoft Exchange POP3 services are
restarted on server EX2016SRV1.

[PS] C:\> Restart-Service MSExchangePOP3
```

The same process is used with the Set-IMAPSettings cmdlet to set the IMAP service to use a wildcard certificate.

```
[PS] C:\> Set-IMAPSettings -Server EX2016SRV1 -X509CertificateName mail.exchangeserverpro.net

WARNING: Changes to IMAP4 settings will only take effect after all Microsoft Exchange IMAP4 services
are restarted on server EX2016SRV1.

[PS] C:\>Restart-Service MSExchangeIMAP4
```

With the correct certificate configuration in place the POP or IMAP clients will be able to securely transmit their logon credentials to the server. Secure login is the default setting for POP and IMAP services.

Warning: If the Exchange server has an SSL certificate that exactly matches the X509certificatename value that you provide in the examples above, then that certificate will be bound to the POP or IMAP service instead of the wildcard certificate. The wildcard certificate is only able to be used when no other certificates contain the fully-qualified domain name that you specify when configuring the POP or IMAP settings.

```
[PS] C:\> Get-ClientAccessServer | Get-PopSettings

UnencryptedOrTLSBindings  SSLBindings                    LoginType
------------------------  -----------                    ---------
{[::]:110, 0.0.0.0:110}   {[::]:995, 0.0.0.0:995}        SecureLogin
{[::]:110, 0.0.0.0:110}   {[::]:995, 0.0.0.0:995}        SecureLogin
{[::]:110, 0.0.0.0:110}   {[::]:995, 0.0.0.0:995}        SecureLogin
{[::]:110, 0.0.0.0:110}   {[::]:995, 0.0.0.0:995}        SecureLogin
```

However, if you do find that you have POP or IMAP clients that can't support secure login for some reason, or you want to disable secure login temporarily to test whether a problem is certificate-related or password-related, then you can set plain text login for POP and IMAP. For example, to set plain text authentication for the POP service you can run the following command.

```
[PS] C:\> Set-PopSettings -Server EX2016SRV1 -LoginType PlainTextAuthentication

WARNING: Changes to POP3 settings will only take effect after all Microsoft Exchange POP3 services are
restarted on server EX2016SRV1.

[PS] C:\> Invoke-Command -ComputerName EX2016SRV1 {Restart-Service MSExchangePOP3}
```

Both POP and IMAP services also have protocol logging available for troubleshooting. Protocol logging for POP and IMAP works similarly to protocol logging on send connectors and receive connectors, and is not enabled by default. To enable protocol logging for POP on a server you can run the following command.

```
[PS] C:\> Set-PopSettings -Server EX2016SRV1 -ProtocolLogEnabled $true

WARNING: Changes to POP3 settings will only take effect after all Microsoft Exchange POP3 services
are restarted on server EX2016SRV1.

[PS] C:\> Invoke-Command -ComputerName EX2016SRV1 {Restart-Service MSExchangePOP3}
```

The protocol logs are stored by default in *C:\Program Files\Microsoft\Exchange Server\V15\Logging\Pop3* or *\Imap4*. Much like the protocol logs for Transport connectors, the POP and IMAP protocol logs are simple text files in CSV format that can be read in Notepad, Excel, or parsed using a text-analysis tool.

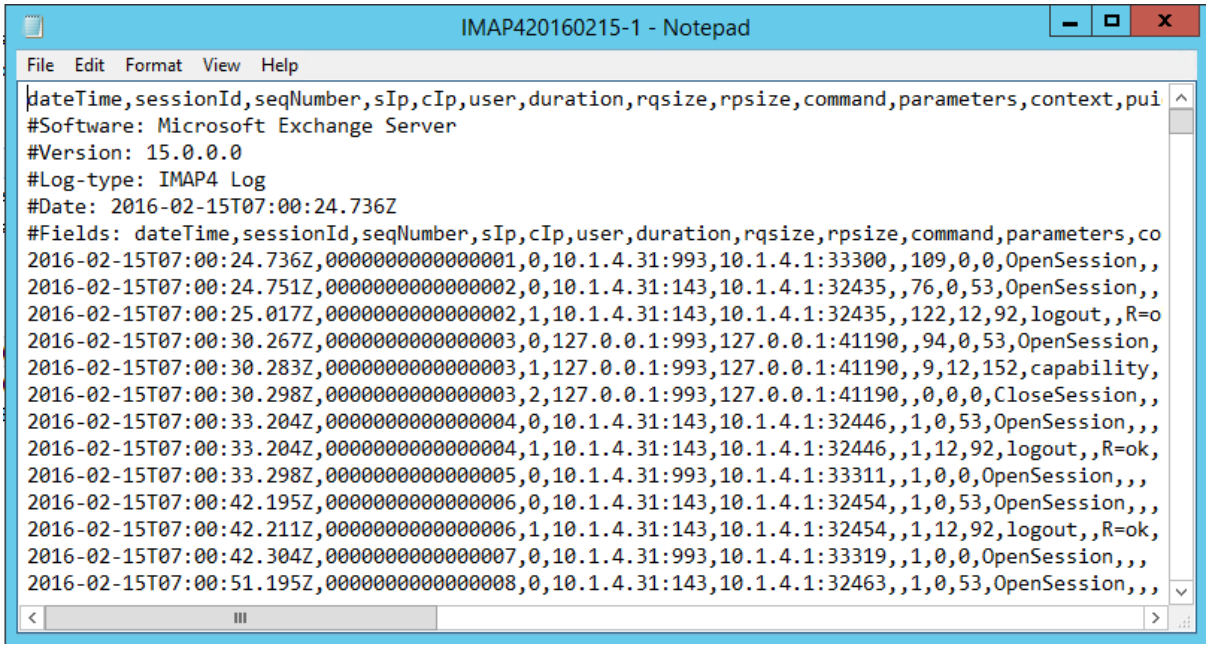


Figure 7-17: An IMAP protocol log

If you have a load balancer in the environment that is performing health checks for the POP or IMAP services, then you'll see a moderate amount of protocol logging each day due to those probe connections from the load balancer.

Warning: Protocol logging for POP and IMAP does not have the same retention capabilities as protocol logging for Transport. The protocol log files will not be automatically removed when they exceed a certain size or age limit, and will eventually consume all available disk space on the volume. For this reason, it is recommended to only enable protocol logging for POP or IMAP temporarily while troubleshooting a problem, and then disable it again when the troubleshooting has been completed.

POP and IMAP protocols are only used to download email from mailboxes. To send email a client needs to use SMTP, which we'll cover in the next section of this chapter.

SMTP

Simple Mail Transfer Protocol (SMTP) is the protocol used in an Exchange environment to send email messages between servers, and also between some clients and servers. SMTP is not used by Outlook clients that are connecting via Outlook Anywhere or MAPI/HTTP, mobile clients connecting via ActiveSync, or other clients connecting via Exchange Web Services. However, SMTP is used by POP/IMAP clients (which are only

mail-access protocols), as well as by many devices and applications in a typical network environment, to send messages. It's quite common to use an Exchange server as the SMTP provider for scanners, photocopiers, UPSs, hardware appliances, as well as numerous applications that need to send email notifications.

SMTP clients connect to the closest matching receive connector on the Exchange server when they are sending an email message. In chapter 4 of we've already looked at how receive connectors work, and how you can troubleshoot them.

For "dumb" devices a simple, unencrypted, unauthenticated SMTP connection is used, and the Exchange server can accept those connections on its default receive connector, or on a custom receive connector that is configured to allow SMTP relay from specific IP addresses. However, for many SMTP clients there is often additional considerations when authentication and encryption of the SMTP connection is required. POP and IMAP clients (discussed in the previous section of this chapter) are a good example.

If a POP/IMAP client tries to authenticate an SMTP connection of an unencrypted connection, a message is received with words to the effect of "*The outgoing server (SMTP) mail.exchangeserverpro.net does not support the selected authentication method.*" To resolve this, the client needs to be configured to use STARTTLS. Depending on the email client being used the settings will look different, but here's an example from Mozilla Thunderbird's outgoing server settings (Figure 7-18).

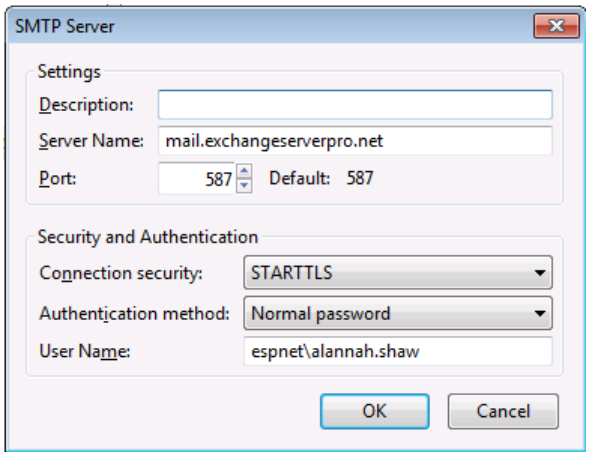


Figure 7-18: Mozilla Thunderbird's outgoing SMTP server settings

Note: In the example above port 587 is used, which is configured during Exchange setup on the "Client Frontend" receive connector on the Exchange server. 587 is the SMTP port that should be used by authenticated clients.

The TLS encrypted connection requires a trusted and valid SSL certificate on the server. Certificates are covered in chapter 3. With a trusted, valid SSL certificate installed and enabled for SMTP on the Exchange server, the SMTP client may still receive errors or warnings when trying to establish a TLS encrypted session with the server. The reason for this is that the receive connector has not been configured with the TLS certificate name.

The TLS certificate name value is made by combining the issuer and subject into a single string. This is achieved by using PowerShell. First, determine the thumbnail value for the certificate you want to use. In this example I'm going to use my wildcard certificate, which is already enabled for SMTP.

```
[PS] C:\> Get-ExchangeCertificate

Thumbprint                                Services   Subject
----------                                --------   -------
D8C33B1E0FDFE180920C5CEED0612B95269FA1E7  IP.WS..    CN=EX2016SRV2
31E5D6D7E6BD77FC20FA4F490983C6945631CB6C  .......    CN=WMSvc-EX2016SRV2
```

```
DE67EC3C8D679AA35D17678FEC51907272B1BAE2  ...WS..    CN=*.exchangeserverpro.net, OU=IT, O=LockLAN
Systems Pty Ltd, L...
```

Capture the certificate as a variable, specifying the thumbprint of the certificate that will be used for the receive connector.

```
[PS] C:\> $cert = Get-ExchangeCertificate -Thumbprint DE67EC3C8D679AA35D17678FEC51907272B1BAE2
```

Now, declare a new variable that combines the issuer and subject values for the certificate.

```
[PS] C:\> $tlscertificatename = "<i>$($cert.Issuer)<s>$($cert.Subject)"
```

Finally, set the *TlsCertificateName* property on the server's receive connector. This step is repeated for as many servers as you need to configure.

```
[PS] C:\> Set-ReceiveConnector "EX2016SRV1\Client Frontend EX2016SRV1" -TlsCertificateName
$tlscertificatename
```

SMTP clients are also subject to message rate limiting on the receive connectors for the Exchange server they are connecting to. You can view the message rate limits for a server by running the *Get-ReceiveConnector* cmdlet.

```
[PS] C:\> Get-ReceiveConnector -Server EX2016SRV1 | select name,messageratelimit,messageratesource

Name                               MessageRateLimit MessageRateSource
----                               ---------------- -----------------
Default EX2016SRV1                 Unlimited               IPAddress
Client Proxy EX2016SRV1            5                            User
Inbound from Office 365            Unlimited               IPAddress
Outbound Proxy Frontend EX2016SRV1 Unlimited               IPAddress
Client Frontend EX2016SRV1         5                            User
```

As you can see, most of the connectors have an unlimited message rate limit. However, the receive connector used by SMTP clients has a message rate limit of 5 (per minute), based on the user. For humans sending email this is likely to be enough for their needs, and indeed most email clients will handle rate limiting without issue, and will simply retry any messages that were not able to be sent on the first attempt.

However, some applications that use SMTP may not handle rate limiting very well, and will simply drop the messages that could not be sent. Ideally this is addressed in the application's code, but if you have a need to set a higher rate limit you can do so by running *Set-ReceiveConnector*.

```
[PS] C:\> Set-ReceiveConnector "EX2016SRV1\Client Frontend EX2016SRV1" -MessageRateLimit 100

[PS] C:\> Set-ReceiveConnector "EX2016SRV1\Client Proxy EX2016SRV1" -MessageRateLimit 100
```

In the case of the "Client Frontend" connector, be sure to set the new limit on the corresponding "Client Proxy" connector as well, as shown in the example above.

Real World: A high value for the message rate limit is preferable to an "unlimited" rate limit, to protect the server against a rogue application that may try to send thousands or even millions of messages through the Exchange server.

What Else Can Cause Client Issues?

Clients are subject to a wide variety of conditions that can cause connectivity, authentication, and performance problems that require troubleshooting. Aside from the items already discussed in this chapter, here's some additional things to be aware of.

Throttling Policies

Exchange Server uses throttling policies to ensure that clients are deliberately or inadvertently overloading the server, and to ensure that resources are being shared proportionally. Throttling policies cover a broad range of scenarios, from maximum concurrent OWA sessions, to maximum number of ActiveSync device associations, and even the maximum number of discovery keywords that can be used.

A default throttling policy is created in every organization, with generous limits for most settings, and even many limits that are set to "*Unlimited*". You can view the throttling policy by running the *Get-ThrottlingPolicy* cmdlet.

As a general rule, you should not modify the default throttling policy. Some specific software products that need to integrate with Exchange in a way that would breach the default throttling policy limits will often come with instructions from the third party vendor as to how to configure a new throttling policy specifically for that product.

Active Directory Permissions and Privileged Accounts

When users log on to some client protocols such as OWA and ActiveSync, the Exchange server computer account needs to access the user's mail attributes in Active Directory. If the Exchange server computer account doesn't have read access to those attributes, then the login will fail.

To check for the cause of this issue, open the properties of the user object in Active Directory Users & Computers, select the **Security** tab, and then click the **Advanced** button. Note that if you don't see the **Security** tab then you need to first enable **Advanced Features** in the **View** menu of the Active Directory Users & Computers console.

Make sure that permissions inheritance is enabled for the user object, so that the correct ACLs apply for Exchange server access to the user. All that should be required by you is to click the **Enable Inheritance** button (Figure 7-19), and apply the changes.

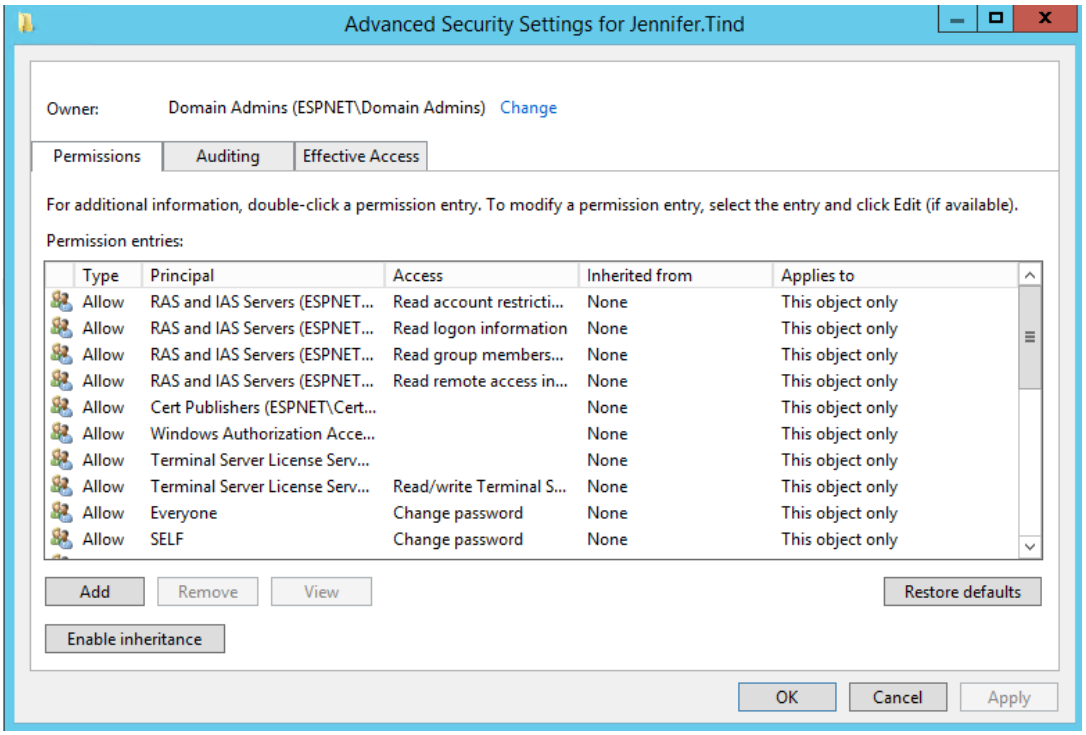


Figure 7-19: An Active Directory user object with permissions inheritance disabled

However, you may find that after performing this step the permissions inheritance is found to be disabled again a short time later. This is due to the concept of protected groups and AdminSDHolder. A user that is a member of a protected group, such as Domain Admins, Enterprise Admins, or Account Operators, will be automatically disabled for permissions inheritance by Active Directory.

The solution to this should be obvious – don't use members of protected groups for day to day access to mailboxes over OWA and ActiveSync. Doing so risks those credentials being exposed, even if the connections are occurring over a HTTPS connection. Use a separate administrative account for administrative tasks, and keep your day to day account as low privilege as possible.

Certificates

As most client connectivity occurs over encrypted connections, SSL certificates play an important role in allowing a successful connection to occur. Certificates have been discussed in several of the sections of this chapter, and are explained in more detail in Chapter 3.

Load balancing

In a high availability deployment, load balancers are often used to distribute client traffic across multiple Exchange servers. Client connectivity is therefore subject to the configuration and availability of the load balancer itself. In addition, if the load balancer is sending traffic to inactive or unhealthy servers, then the client experience will suffer.

> **Real World:** Even a minor setting in a load balancer configuration can have a big impact on the client connection, or the user experience. In one troubleshooting case I worked on it was determined that a compression setting on the load balancer was causing some of the JavaScript code for OWA to be corrupted in transit, which broke much of the OWA interface for end users.

Recipient configurations

Even a successfully connected client may experience issues that are not client-related, but rather recipient-related. Recipients are discussed in more detail in chapter 6.

Performance

The client experience can easily be impacted by the performance of the Exchange server itself. Even if all network, certificate, authentication and other factors are working perfectly, if the client is connected to an overloaded Exchange server then the experience will be poor. Server performance is discussed in more detail in chapter 8.

Additional Reading

Autodiscover

- Exchange 2013 Autodiscover Service
- New Behavior in Outlook 2013 Causing Certificate Errors in Some Environments

Client Connectivity

- Network Ports for Clients and Mail Flow in Exchange 2013

- Provisioning, policies, remote wipe, and the Allow/Block/Quarantine list in Exchange ActiveSync
- Exchange Remote Connectivity Analyzer
- MAPI over HTTP in Exchange 2016

Other

- AdminSDHolder, Protected Groups, and SDPROP

Chapter 8: Troubleshooting Performance

Andrew Higginbotham

Proper performance troubleshooting and analysis skills are the most transferrable skillset of any IT support professional. I've personally known support engineers who were subject matter experts in Windows performance move to an equivalent role with Exchange, SQL, or Azure. All without making a 180-degree change in career path, but instead taking their core performance and issue analysis skills and simply building upon them. While this chapter focuses on Exchange-specific performance troubleshooting, let's take a moment to discuss the troubleshooting skills that span technologies.

Establishing a baseline

As previously mentioned in Chapter 1, when you have performance issues, the best defense can often be a good offense. In other words, the preemptive collection of performance data and historical trends can be your most valuable resource when you eventually meet a challenging performance issue that's affecting production. Having the following information will assist in determining how an Exchange server has deviated from the baseline:

- Number of mailboxes
- Messages Sent/Received daily
- Average Message Size
- Expected IOPS per Database/Disk/Server
- Average Disk Latency
- Average Memory consumption
- Average CPU Utilization
- Average connections per client type at the Client Access Layer (typically the Load Balancer)

With this information, you're not gathering data and making assumptions, you're simply looking for divergence from the baseline to current state. How can you possibly know what poor performance is without first knowing what acceptable performance is? The important thing is to avoid personal opinion and bias by having raw data readily available to compare against. It doesn't matter if you use the built-in Microsoft toolsets or a third-party monitoring solution to collect the data. Select a performance monitoring and testing solution that allows you the best combination of usability, functionality, feature set, and price.

You support the solution end-to-end

When you're responsible for the performance of an application, it means you're responsible for it end-to-end. If your solution runs on physical servers with local hardware then you must understand how to measure performance within the application, in the operating system, and down to the physical hardware and through the network stack. If you virtualize an application and use Storage Area Network (SAN) storage then you must be capable of measuring performance within the application, in the operating system, in the hypervisor, through the network stack, through the SAN network fabric, and at the SAN. The mindset of "that's not my department so it's not my responsibility" is a cancer to performance troubleshooting. Every performance support case I've encountered where the teams operated in silos and remained distant from each other, thus

creating an "us vs them" mentality and resulting in issues lasting exponentially longer than necessary. "Operational maturity" is commonly used in the industry to describe an organization's effectiveness in managing the technology they choose to implement. An organization that has close ties with their application/operating system/hypervisor/storage vendors, runs according to best practices, and has excellent inter-departmental efficiency and communication is said to be operationally mature. Whereas a company that's new to a technology, has invested little in training, does not follow vendor guidance, and has poor intra-organizational communication and collaboration is said to be operationally immature. The operationally immature company is going to encounter more support incidents and performance issues than their mature counterparts. Your job as an Exchange performance specialist is to understand what's necessary to maintain a successful and well-performing implementation. For this, you must learn how to track and measure performance at each level of your Exchange infrastructure:

- Exchange metrics/statistics (messages per day, etc.)
- Windows Performance Counters
- Network monitoring data (dropped packets, retransmits, etc.)
- Hypervisor Performance Counters (%RDY, Memory/CPU utilization, etc.)
- Storage Area Network monitoring data (IOPS, replication metrics, etc.)

Whether or not you're a member of the SAN team or the networking team, if your application (Exchange in our case) is performing poorly as a result of another team, you're still accountable until you can prove otherwise. It is for this reason you must educate yourself on every aspect of the Exchange deployment, as you cannot always rely on others to show your same level of dedication to the performance of your application.

Trust but Verify

When you are asked to help resolve a performance issue, there will always be a period of information gathering. Some information will be facts-based while some will be conjecture. It's your responsibility to verify the validity of all statements or data. The nature of humans is that we like to have an explanation for the unknown. If a system begins performing poorly and it happens to correspond to a recent update, then it would only make sense that the update was the cause. If an Exchange Administrator encounters instability and performance degradation following an update of Active Directory to use Windows Server 2012 R2 Domain Controllers, they will certainly think that the blame probably lies with the Directory Services Team. Each of these issues may have no relation to each other whatsoever, but they must still be investigated. Correlation does not imply causation is a phrase commonly used in statistics and the sciences to encourage others not to jump to conclusions. It also applies to information technology.

When troubleshooting Exchange performance, all data must be considered. However, we must not arrive at a conclusion before first analyzing the data, developing a theory, and testing. A common mistake is making your own conclusion and manipulating the data to fit your conclusion (because we as humans do not like being wrong) instead of allowing the data to carry us to the correct conclusion.

If you're looking for good questions to get you started on the correct path to troubleshooting a performance issue, consider the following:

- What component is experiencing the symptom? (Outlook/ActiveSync/OWA)
- When did you first notice the symptoms?
- Has the symptom been constant, sporadic, or periodic?
- If a baseline is present, how does the symptom deviate from it? (for example: Baseline of 20ms Avg. disk latency vs current 200ms Avg. disk latency)
- What is the scope of the performance problems? (single user/disk/database/server vs system wide)
- Does the affected scope share common components? (same SAN/Virtual Host/Switch)

- If there are other systems not experiencing the issue, what is different on the non-affected systems? (update level/# of users/connection quality)
- Is the symptom getting worse, improving, or stable?

Validate the solution and use validation data for troubleshooting

Many Exchange performance issues are due to improper sizing of some component and therefore could have been avoided if proper care had been taken when the infrastructure was designed. Yet even if a solution is properly sized, it's still possible to encounter performance issues if the environment drastically deviates from its origins. When discussing Exchange performance validation, two tools come to mind:

- Jetstress – Tool for validation of Exchange storage. Uses Exchange ESE to generate IO with the goal of generating the maximum number of IOPS while staying within acceptable disk latency thresholds. This information is used to confirm that the storage solution will deliver at least as many IOPS as required for the solution. Exchange is not required to be installed to run Jetstress.
- LoadGen – Exchange user simulation tool to place synthetic workload on an Exchange server with the goal of stressing not only disk but RAM and CPU as well.

Another important tool to this process is the Exchange Server Role Requirements Calculator which is used to determine the goals the tests strive to achieve. While the goal of this book is not to teach how to size Exchange, understanding what components are required to make an Exchange environment run at a satisfactory level is vital to recognizing an improperly sized environment. We will therefore discuss at a high level the steps required to properly size and validate an Exchange environment. Afterwards, we'll discuss how this data can potentially be used for performance troubleshooting in the future. Some steps have been omitted for simplicity, such as "Determine Goals", "List Constraints", and so on. The steps are as follows:

Gather Inputs

- Number of mailboxes using profile tiers
- Messages Sent/Received per day for mailboxes in each tier
- Average Message Size for mailboxes in each tier
- Mailbox and Archive mailbox size for mailboxes in each tier
- Deleted Item Retention Period for mailboxes in each tier

The calculator allows you to place this data into User Mailbox Tiers, assuming not every user in an environment uses email in the same manner. Figure 8-1 shows a theoretical environment with 90,000 mailbox, 80,000 of which use email in a different way than the remaining 10,000. This fact led to the decision to categorize them in their own tier, thus helping in our sizing decisions. Ideally these 90,000 mailboxes will be evenly distributed amongst the Exchange servers to achieve balance and predictable performance load distribution.

Tier-1 User Mailbox Configuration	Value
Total Number of Tier-1 User Mailboxes / Environment	80000
Projected Mailbox Number Growth Percentage	0%
Total Send/Receive Capability / Mailbox / Day	200 messages
Average Message Size (KB)	75
Initial Mailbox Size (MB)	2048
Mailbox Size Limit (MB)	10240
Personal Archive Mailbox Size Limit (MB)	0
Deleted Item Retention Window (Days)	14
Single Item Recovery	Enabled
Calendar Version Storage	Enabled
Multiplication Factor User Percentage	100%
IOPS Multiplication Factor	1.00
Megacycles Multiplication Factor	1.00
Desktop Search Engines Enabled (for Online Mode Clients)	No
Predict IOPS Value?	Yes
Tier-1 User IOPS / mailbox	0.00
Tier-1 Database Read:Write Ratio	3:2

Tier-2 User Mailbox Configuration	Value
Total Number of Tier-2 User Mailboxes / Environment	10000
Projected Mailbox Number Growth Percentage	0%
Total Send/Receive Capability / Mailbox / Day	50 messages
Average Message Size (KB)	50
Initial Mailbox Size (MB)	512
Mailbox Size Limit (MB)	5120
Personal Archive Mailbox Size Limit (MB)	10240
Deleted Item Retention Window (Days)	90
Single Item Recovery	Enabled
Calendar Version Storage	Enabled
Multiplication Factor User Percentage	100%
IOPS Multiplication Factor	1.00
Megacycles Multiplication Factor	1.00
Desktop Search Engines Enabled (for Online Mode Clients)	No
Predict IOPS Value?	Yes
Tier-2 User IOPS / mailbox	0.00
Tier-2 Database Read:Write Ratio	3:2

Figure 8-1: User Mailbox Tiers in the Exchange Server Role Requirements Calculator

These are typically the most impactful inputs that will determine both your performance and sizing needs. As a general rule, Messages Sent/Received per day will have the greatest impact on the amount of performance you will need out of the system. Typically, changing this value will adjust the IOPS required per server value on the Role Requirements tab of the calculator. Whereas Avg. Message Size, mailbox size and Deleted Item Retention will determine the amount of storage capacity needed by the infrastructure. The number of mailboxes will obviously have an impact on both performance as well as capacity requirements.

These inputs are relevant to performance troubleshooting because changes to these values can drastically alter the required resources for running Exchange properly. I worked on a very high profile escalation for a large customer where the hardware was mistakenly to blame for Exchange performance issues. Their original environment was **sized** as follows:

- 70,000 mailboxes
- 100 Messages Sent/Received Per Day (per mailbox)
- Avg. Message Size of 75 KB

The customer was experiencing severe performance issues, culminating in server hangs and Outlook clients disconnecting. At first glance, disk latency was extremely high (Avg. of 200ms for extended periods) and it was thought that the hardware was not performing properly across all 8 servers. Upon inspection of Performance Monitor, it was determined that the servers were generating almost twice the IOPS than the hardware was sized for and capable of delivering. This was determined by looking at the Disk Transfers/sec counter. It was at this point where I asked for the customer's Exchange Sizing Calculator from the original design exercise. After reviewing the calculator output as well as messaging statistics, the following was determined:

- Customer currently had closer to 90,000 mailboxes
- Current Messages Sent/Received per day (per mailbox) was 150

After further discussion, it was determined the customer had acquired another company (responsible for the 30% mailbox count growth) and their Exchange Administrator had recently left the company. So the customer's inputs had changed and there was not sufficient monitoring in place to detect this growth and change in usage. The change from 100 Messages Sent/Received per day to 150 actually resulted in a 60% growth in required IOPS per Server. This value is present on the Role Requirements tab of the Exchange Calculator. Specifically, the "Total Database Required IOPS/Server" value. (Figure 8-2). Note that only database required IOPS per server (2,111) are relevant to performance planning as log IOPS are sequential and not considered burdensome on rotational disk storage

Host IO and Throughput Requirements	/ Database	/ Server	/ DAG	/ Environment
Total Database Required IOPS	21	2111	33768	33768
Total Log Required IOPS	5	447	7145	7145
Database Read I/O Percentage	60%	--	--	--
Background Database Maintenance Throughput Requirements	1.0 MB/s	100 MB/s	1600 MB/s	1600 MB/s

Figure 8-2: IOPS requirement as determined by the Exchange calculator

The hardware was actually over performing rather than underperforming. The ultimate solution was to scale out the environment by adding additional Exchange Servers. During this process the Exchange Calculator was again used to properly size the environment.

This is an excellent lesson in how accurate and up-to-date inputs are vital to sizing an Exchange environment, as well as troubleshooting its performance. I recommend all customers keep the spreadsheets generated by the Exchange Server Role Requirements Calculator for the life of the Exchange solution and regularly compare current inputs to the original specifications.

Storage

With any storage solution, I often find myself explaining to customers the level of performance they can expect to achieve from the hardware in question. Due to the various factors involved, the most accurate answer would be to say the solution can achieve whatever performance rating that it has actually been validated for. Meaning, place your workload on the storage solution and see what you can achieve; tuning performance as necessary. If the goal is to determine how many IOPS a solution can achieve then it greatly depends on the workload being performed against it. For example, if the individual IO is very small (4 KB) it would be much easier to achieve more IO per second (IOPS) than if the IO size were large (1 MB). This is a trick that vendors often use to inflate their achievable IOPS numbers, forgetting to mention that the achievable IO greatly depends on the application that generates IO against the storage.

So while the best answer (the Consultant answer) is often "it depends" when asked about achievable IOPS, there are some general statements around disk performance which can be used for guidance. Each disk type has a given amount of IOPS that it can expect to provide within acceptable latency thresholds. The Exchange Server Role Sizing Calculator has a very helpful hidden table which provides estimates for achievable IOPS per disk type (Figure 8-3). In this instance, for Exchange sizing, we care only about Random IO.

Disk Drive	Random Disk I/O	Sequential Disk I/O
5.2K RPM SATA 2.5"	55	300
5.4K RPM SATA 2.5"	55	300
5.9K RPM SATA 2.5"	55	300
5.2K RPM SATA 3.5"	50	300
5.4K RPM SATA 3.5"	50	300
5.9K RPM SATA 3.5"	50	300
5.2K RPM SAS 3.5"	52.5	300
5.4K RPM SAS 3.5"	52.5	300
5.9K RPM SAS 3.5"	52.5	300
7.2K RPM SATA 2.5"	60	300
7.2K RPM SATA 3.5"	55	300
7.2K RPM SAS 2.5"	62.5	300
7.2K RPM SAS 3.5"	57.5	300
10K RPM SAS 2.5"	165	300
15K RPM SAS 2.5"	230	300
10K RPM FC/SCSI/SAS 3.5"	130	300
15K RPM FC/SCSI/SAS 3.5"	180	300

Figure 8-3: Expected IOPS per disk type

As you can see, there's a significant difference in the expected IOPS of a 7.2K NL SAS disk (~55 IOPS) compared to a 15K SAS disk (~180 IOPS). You'll notice SSD drives are not mentioned, as they are not recommended for Exchange deployments due to cost and small size, but they can achieve IOPS in the many thousands.

In troubleshooting Exchange storage performance, it's important to understand the type of disk storage being used. This will give you a rough estimate of the achievable IOPS of the overall solution. For example, a server with 12 7.2K NL SAS hard drives should be able to achieve roughly 700 IOPS. It's certainly not an exact science due to the various potential size of IO as well as sequential vs random IO, but it's usually good enough for fair estimations. For example, if the Performance Monitor \PhysicalDisk\Disk Transfers/Sec counter (which effectively tracks IOPS) displays over 2,000 IOPS on the same 12 disk system previously mentioned, it would be a fair assumption that the workload was generating more IOPS than the storage solution can support. So it would be expected to see disk latency at unacceptably high values on this solution (disk latency over Avg. 20ms is considered poor).

Note: It's preferred to use object PhysicalDisk, to ensure we look at the closest number to the Disk Subsystem. When using Bitlocker for instance, IOPS will increase up to 8 fold from original estimate at the LogicalDisk object level.

I often use this chart to set expectations with customers for the number and type of disks they have in the system. It's important to understand that scientifically speaking, there is going to be a performance limit of any storage solution. Knowing that as you move nearer this limit, latency will increase, is critical for troubleshooting storage performance.

Note: With most storage solutions, placing a faster disk in the same array as slower disks will not deliver the desired results. As an example, adding a 10K disk in an array full of 7.2k disks. In some cases, it can even introduce latency as the array compensates for the differing disk speeds. Therefore, if you choose to add faster media, do so for the entire array/virtual disk.

Now you might ask "why not simply purchase faster drives and achieve greater IOPS per spindle?" The answer is simply cost per gigabyte. The goal of the Exchange Product Team has been to provide users with the biggest and fastest mailbox for the cheapest price. With reduced IOPS requirements, faster disks are no longer

required. Also, faster drives historically grow in capacity the slowest, while slower drives (such as SATA or NL SAS) increase in capacity at a much faster rate. This Geek out with Perry session explains the intent behind the design decisions made for Exchange when it comes to how storage is used.

> **Note:** NL SAS disks are really just SATA disks with SAS logic. This means that the drives typically perform at SATA speeds. Generally speaking, a large (2-8TB) 7.2K SAS drive is synonymous with NL SAS.

Memory and CPU

The Exchange Product Team has never been shy about their recommendations when it comes to deployment and sizing guidance. The most recent guidance (June 2015) says that Exchange servers should not exceed 24 CPU cores and 96GB of RAM. Instead, customers should scale out their infrastructure as they grow, instead of scaling up. It's important to understand the reasoning for these recommendations, as they were partially formed from support cases related to poor performance.

Many of these recommendations revolve around .NET performance. Microsoft Support has encountered cases where very dense servers, with many CPU cores, encountered performance issues with .NET garbage collection. This issue is described in further detail in this TechEd Europe session on Exchange 2013 performance monitoring and tuning. It is for these same reasons that the Exchange Product Team recommends disabling CPU hyperthreading on physical servers, as giving Exchange the impression that twice the CPU cores are available than actually are can result in poor .NET performance. As you probably guess, Exchange health is dependent upon .NET health. Therefore it's critical the recommended version of .NET is deployed on Exchange 2010/2013/2016 servers.

This post from a Microsoft Support Escalation Engineer outlines excellent CPU sizing and performance troubleshooting tips for Exchange. One tip in the article I strongly recommend for all Exchange deployments is to change server power plans to "High Performance" mode or an equivalent setting. Many servers have BIOS power settings which controller system and processor power states. These are typically utilized to save power, encourage green computing, and reduce power consumption costs. However, these solutions are not always optimal for systems performing work where timing is critical, such as Exchange or SQL. It's for this reason that Microsoft recommend disabling these power saving features to ensure optimal processor performance.

Lastly, while I feel it's important to take Microsoft's CPU and Memory sizing recommendations into consideration, they are still only recommendations and are not support statements. They are based on best practices developed through trial and error gained from running Exchange Online servers inside the Office 365 cloud service. Servers with 24 cores and 96GB of RAM have been battle-tested and proven to be excellent at running Exchange Server in Microsoft's datacenters. Therefore, deploying a similar platform is likely to yield a stable and well-performing Exchange solution.

Storage Validation

Knowing that the underlying storage has been validated to handle the expected workload is an important part of being able to reliably support the actual workload. It's why Microsoft Support has long stated that validating storage with Jetstress is one of the requirements for being "supported". The job of Jetstress is fairly simple: place a pre-determined IO load on the disk subsystem and verify the latency of the storage is acceptable to run Exchange. I highly suggest that you read the Jetstress Field Guide to better familiarize yourself with how to properly use this tool. In short, the process is as follows:

- Gather inputs to be used in the Exchange Server Role Requirements Calculator
- Use Exchange Server Role Requirements Calculator to determine # of Database IOPS per Server

- Run Jetstress with goal of achieving predetermined IOPS value, while achieving a "Pass"
- Optional: Determine max achievable IOPS on storage solution while still achieving a "Pass"

I often joke saying many of my Exchange performance cases involve servers where Exchange hasn't yet been installed. This is due to customers having issues during the validation (Jetstress) phase of their deployment. These cases can be broken into two categories:

- Unfamiliarity with the Jetstress tool and the proper testing methodology
- An actual hardware, configuration, or sizing issue

Reading through the Jetstress Field Guide usually resolves the former issue. People who are unfamiliar with Jetstress testing will often install Jetstress, crank the thread count to an unreasonably high value, and blame the hardware when it fails. My answer is that I can theoretically make Jetstress fail on any storage solution. As for the latter, the resolution is typically one of the following (most which will be covered later in this chapter):

- Controller or hard disk firmware issue
- Controller cache misconfiguration
- Hardware issue (loose cable, predictive failure drive, etc.)
- SAN NIC/HBA (firmware or configuration)
- SAN LUN configuration
- SAN Network configuration
- Anti-Virus or other file system filter driver

Having the knowledge that a storage solution passed Jetstress means that at one time, it did function properly. To a troubleshooter, this means something has changed in the environment causing it to not function as expected. Passing a Jetstress test also gives an indication of the IOPS that can actually be delivered by the solution. Upon completion, Jetstress generates an .HTML results file giving the Pass/Fail result of the test as well as the latency values measured on each drive. This is also a useful file to keep as it can help troubleshooting efforts if problems subsequently develop. It is not unheard of to have a firmware, operating system, or anti-virus update change the latency within an Exchange configuration. If drastically different results are observed compared with when the storage was originally validated, this is an indication that something fundamental has changed, assuming the IO load is the same.

Understanding controller caching and Exchange performance

One of the primary talking points of my Storage Configuration Options for Exchange session given at the IT Dev Connections conference in 2015 was around the use of JBOD storage with Exchange and what that definition means to various people. Since Exchange 2010 and the advent of the Database Availability Group, the idea of deploying Exchange onto JBOD has spread like wildfire. Starting with laughs and jeers from the IT community at the mere idea of placing production workloads onto non-RAID protected direct-attached storage, evolving to the largest Exchange deployment in the world (Office 365) running on JBOD storage. Not only does Exchange Online run on JBOD, but mailboxes in the service do not have traditional backups performed against them. Instead relying upon Exchange Native Data Protection. Quite a shocking fact, especially if you were to present it to an Exchange Admin around the year 2009.

For the correct deployment architecture, JBOD actually makes a lot of sense once you understand the performance and High Availability improvements made in the product. Exchange 2013/2016 requires ~90% fewer IOPS than Exchange 2003, which makes large/slow/cheap disks such as 6TB 7.2K NL SAS a viable

deployment option for even the largest infrastructures. The reduction in IOPS also removed the requirement for expensive SAN storage with optimized RAID configurations to achieve acceptable performance. In addition, with a Database Availability Group (DAG) being able to support up to 16 geographically diverse copies of a mailbox database (although 3-4 is a more practical number), there's no need to waste drives on RAID when the application itself can handle your data availability and redundancy.

Although I'm not here to tout the capability of Exchange High Availability (that topic is addresses in its own book), I do want to discuss a common misconception around the hardware requirements for an Exchange JBOD solution (as specified in Microsoft's Preferred Architecture). There's no doubt that if you get storage wrong, it results in poor Exchange performance and probably leads to investigation of the hardware. In every such case that I have been involved with, the root cause was not the deployed hardware which was at fault, but rather an inappropriate hardware configuration.

The definitive source of information regarding Exchange storage is the TechNet article describing the Exchange Storage Configuration Options. The article details the various supported configurations, storage media, and best practices for Exchange storage; such as:

- Media types and expected speed
- Database and Log File location recommendations
- Requirements for when JBOD is acceptable (opposed to RAID)
- RAID array stripe size
- Controller cache settings
- Maximum database size recommendations
- Supported file systems
- Encryption supportability and recommendations
- Deduplication supportability

The following guidance around controller caching settings is found in the "Supported RAID types for the Mailbox server role" section:

Storage array cache settings:

- *The cache settings are provided by a battery-backed caching array controller.*
- *Best practice: 100 percent write cache (battery or flash-backed cache) for DAS storage controllers in either a RAID or JBOD configuration. 75 percent write cache, 25 percent read cache (battery or flash-backed cache) for other types of storage solutions such as SAN. If your SAN vendor has different best practices for cache configuration on their platform, follow the guidance of your SAN vendor.*

This guidance is important when it comes to deciding on the detailed configuration for a JBOD-based Exchange infrastructure. Unfortunately, many incorrectly assume that if you deploy Exchange on JBOD, a caching controller is not required to achieve the necessary performance. Perhaps this misconception is due to the notion that JBOD is simply a disk connected to a server, with no intelligence or sophisticated controller whatsoever. With Microsoft advertising the performance improvements in Exchange storage, it's easy to see how the mistaken impression that a caching controller is not required. This is absolutely incorrect.

Although Microsoft technically supports the lack of a caching disk controller in Exchange JBOD-based configurations, no guarantees are extended that the solution will be able to provide the performance necessary to run Exchange. This is why running Jetstress is such an important step in an Exchange deployment. The only reason I feel the lack of a caching controller is tolerated is so that Storage Spaces (which require the absence of a caching controller) are supported. However, that's purely my own speculation. So what's the big deal you might ask? If Microsoft does not require a caching controller in a JBOD solution from a supportability

perspective, why is it so vital to a successful and well-performing Exchange solution? Let's start by defining the types of cache and their recommended settings.

On-Disk Cache

Hard drives have a Disk Buffer, often called a Disk Cache, used to cache writes to disk. Caching occurs when the operating system issues a write to disk, but before the data is actually written to a platter, the drive firmware acknowledges to the OS that the write has been committed to disk, thus allowing the OS to continue working instead of waiting for the data to actually be committed. The potential period of latency between caching and commitment can be significant on slower rotational media. Caching increases performance with the known risk that the system loses power before a write is committed to disk, but after the OS has received acknowledgement of committal, data loss/corruption will occur. This is why a UPS is required in such a configuration.

Unfortunately, the cache on NL SAS disks is notoriously small (typically 64-128 MB) and unreliable, which means that they really are not suitable for enterprise workloads. The cache can be easily overwhelmed or susceptible to data loss, also known as a lost flush. If you use a low-end RAID controller which does not contain a cache, they can only rely on the on-disk cache for write performance.

Controller Cache

Disk Array Controllers typically have a much larger (512 MB-2 GB) and more robust cache that is capable of delivering a high write performance. In many situations, the controller cache is the single biggest contributor to delivering the necessary disk write performance for enterprise-level deployments. In fact, the topic of controller caching (or lack thereof) is one of the most common call drivers in hardware vendor storage support.

As an example, let's consider a scenario that happens far too often. A company builds a server with 96 GB of RAM, 16 CPU cores, and 10 TB of storage, but skimps on the RAID controller by purchasing one without cache. A low-end RAID controller may save a few hundred dollars but will turn an otherwise robust system into one incapable of sustaining a storage-intensive workload. This is because the on-disk cache, which I previously mentioned is easily overwhelmed, is all that stands between you and degraded storage performance.

On several occasions I've dealt with Exchange performance escalations where a low-end controller was used on the assumption that an Exchange "JBOD" solution did not require one. The issue was often discovered during Jetstress testing, but in some occasions, the customer had already begun the deployment/migration because they chose to forgo Jetstress testing.

Even some modern high-end controllers with > 1 GB of cache can encounter a performance problem when not properly configured. Because of modern solutions like Storage Spaces, which require no cache, some controllers offer a non-RAID/Pass-through/HBA/JBOD (name varies by vendor) mode. This feature allows selected disks to be presented to the OS as raw disks, with no RAID functionality whatsoever. In other words, no cache. Again, because of misconceptions, I've encountered customers who used this mode for an Exchange Server JBOD deployment because they incorrectly assumed it was appropriate. What makes the situation even more unfortunate is that not enabling the write cache or purchasing a low-end controller are fairly easy problems from which to recover. You either reconfigure the cache (does not even require a reboot) or upgrade the controller (which will import the RAID configuration from the disks), neither of which involves data destruction. However, if a customer deployed Exchange using the Pass-through option, the drive would have to be rebuilt/reformatted. This is an issue that you really hope to discover during Jetstress testing and not after migrating mailboxes.

The question therefore arises as to how do we properly configure JBOD storage for Exchange? I personally like to use the term RBOD for this, because you're really creating a server full of single-disk RAID 0 virtual disks/arrays. This effectively functions as a JBOD solution, which also allows you to reap the benefits of the controller's cache. According to some of my Microsoft contacts, this is how JBOD is implemented in Office 365.

When creating this array/virtual disk for Exchange JBOD, the following settings should be used:

- Write Cache: Use 100% of controller cache used for writes
- Read Cache: Use 0% of controller cache used for reads
- Stripe Size: 256K or 512K (follow vendor guidance)
- On-Disk Cache: Disabled

There are a few things to note regarding this list. The on-disk cache should be disabled to avoid double-caching (caching at the controller as well as the disk) as well as the possibility of overwhelming the on-disk cache. If this is left enabled, the risk of a Lost Flush is increased. In addition, each vendor uses different terminology for caching settings. For example, Dell does not use a percentage value for configuring their cache, instead it's either Enabled or Disabled. For example:

- Write-Back=Write caching enabled
- Write-Through=Write caching disabled
- Read Ahead=Read caching enabled
- No Read Ahead=Read caching disabled
- Disk Cache Policy Enabled/Disabled=On-Disk Cache

Follow your vendor's guidance when configuring any of these settings.

What if I don't use a caching controller

What if you still decide against using a controller with caching functionality? Or to use Pass-through mode? Or to not enable the write cache? Technically, such a solution is supported by Microsoft. However, realistically the only solution which would pass Jetstress would be one where users had a very low IO profile. Maybe users who only sends/receives a few emails daily, and therefore would not require many IOPS per server.

I performed a quick test on my lab server. A physical system with 64 GB of RAM, 12 CPU cores, and a RAID controller with 1 GB of cache. My plan was to use Jetstress on the system with Write-Cache enabled, note the time it took to create the test databases and the achievable IOPS, then repeat the test with the Write-Cache disabled. I expected the testing to be much slower with caching disabled, but even I was surprised with how drastically different the results were.

Testing Parameters:

- 1 Database (40 GB)
- Dedicated virtual disk/Windows partition for database and logs
- Disk Subsystem Throughout Test
- Performance Test
- Autotuning=Enabled
- Duration=15min (.15)

> **Note:** The same virtual disk/array was used for both tests; the only change was the caching setting

Results with Disk-Cache Enabled:

- Time to create 40GB test database=8 minutes
- Jetstress Result=Pass
- Achieved IOPS=300
- Thread count used=6 (via Autotuning)

Results with Disk-Cache Disabled:

- Time to create 40GB test database=4 hours
- Jetstress Result=Fail (due to write latency)
- Achieved IOPS=50
- Thread count used=1

> **Note:** Tests in my lab with caching disabled would always fail with Autotuning enabled. Therefore, I had to manually configure the thread count. After several tests (all failing), I configured the test to only 1 thread, which still failed due to log write latency.

Needless to say, I was surprised. The test went from taking 8 minutes to create a 40 GB file to 4 hours! It was more latent by a factor of 30! Now imagine instead of a Jetstress test, this was a production Exchange server. Maybe the administrator or consultant thought the initial install was taking longer than expected, maybe the mailbox moves were much slower than anticipated, and maybe the client experience was so slow it was unusable. This is the point where the product or the hardware are usually blamed by the users and upper management. It often takes an escalation to Microsoft and/or the hardware vendor to explain the importance of a caching controller. The need to spend for success is never more apparent when it comes to good disk controllers with solid caching.

> **Note:** Always follow hardware vendor guidance. Also, this guidance was specifically for Exchange direct attached storage solutions. For SAN or converged solutions, contact your vendor for guidance. Lastly, always run Jetstress to validate an Exchange storage solution before going into production.

Using Performance Monitor

Windows Performance Monitor or PerfMon, has been used for Windows performance monitoring and analysis for over 20 years. With its rich history comes a huge amount guidance and references on how best to use the tool. From that guidance, I'll summarize what I feel is the most useful to the Exchange performance troubleshooter.

The Interface

PerfMon can be launched via the following common methods:

- Navigating to Administrative Tools and selecting Performance Monitor
- Start>Run>PerfMon
- Executing *perfmon* within a PowerShell or CMD window

The default view brings you to a line graph representing data points captured at configured intervals over time (Figure 8-4).

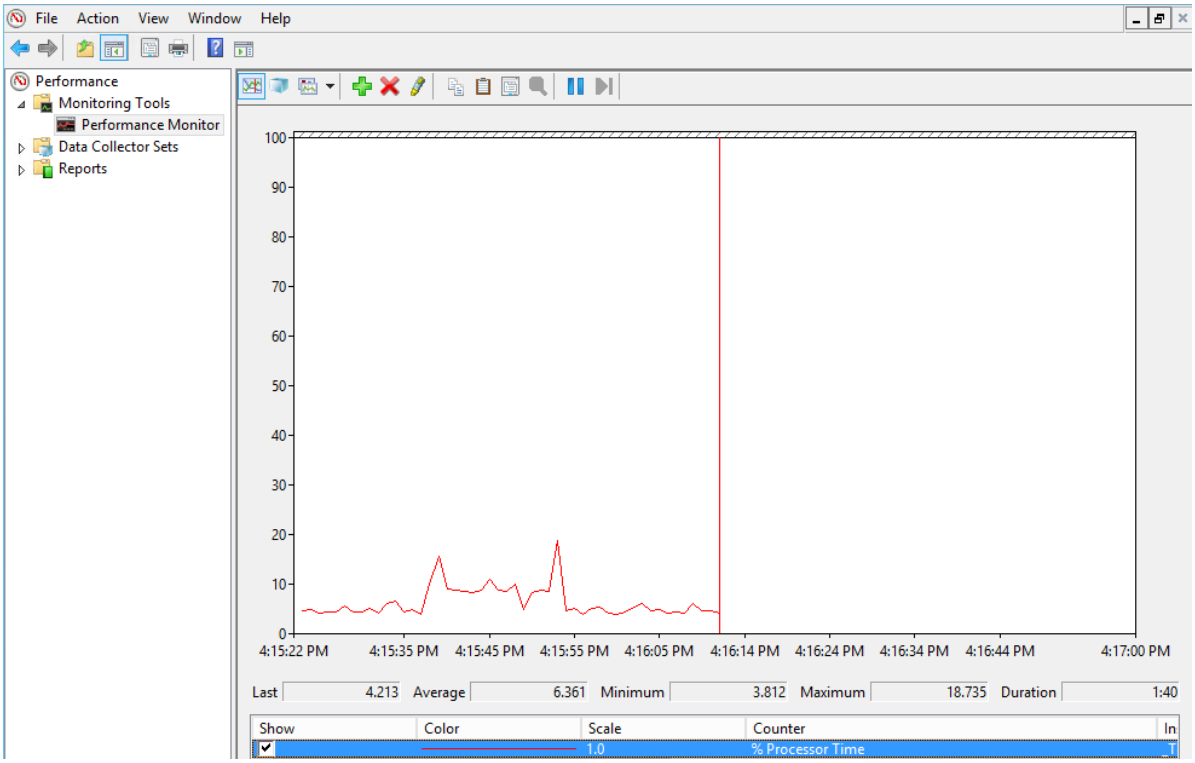


Figure 8-4: Line graph of % Processor Time

Live data provides the default view when you open PerfMon. Counters can be added or removed using the green plus or red X buttons respectively. When adding each counter, a brief description can be viewed detailing what each counter captures (Figure 8-5).

> **Note:** A performance object is an entity for which performance data is available. Performance counters define the type of data that is available for a performance object. An application can provide information for multiple performance objects. Performance objects can contain either single instance counters or multiple instance counters. A single instance object returns a single set of counter values (Reference).

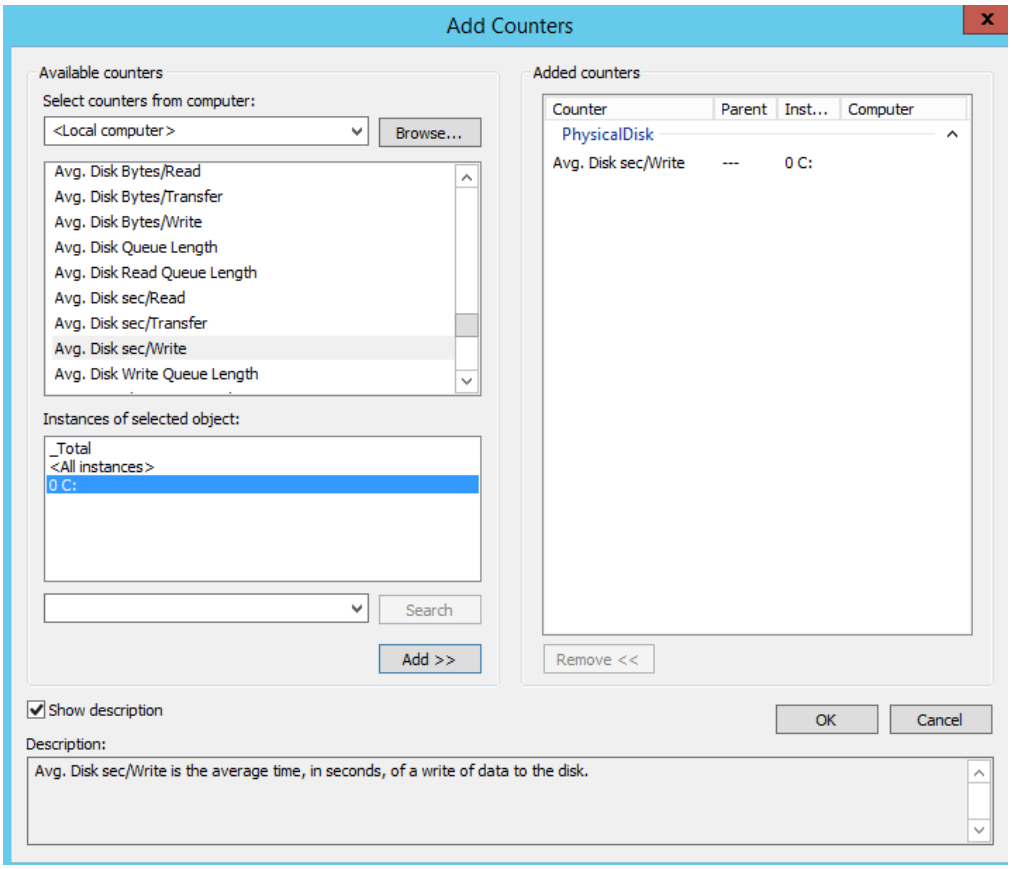


Figure 8-5: Adding the *\PhysicalDisk\Avg. Disk sec/Write* counter with description

When working with multiple counters, I recommend using the Highlight (Pencil) button to display the currently selected counter in bold on the graph. Once you have added the counters you wish to view, I recommend selecting all counters and scaling them (Figure 8-6) so they can all be displayed on the same graph.

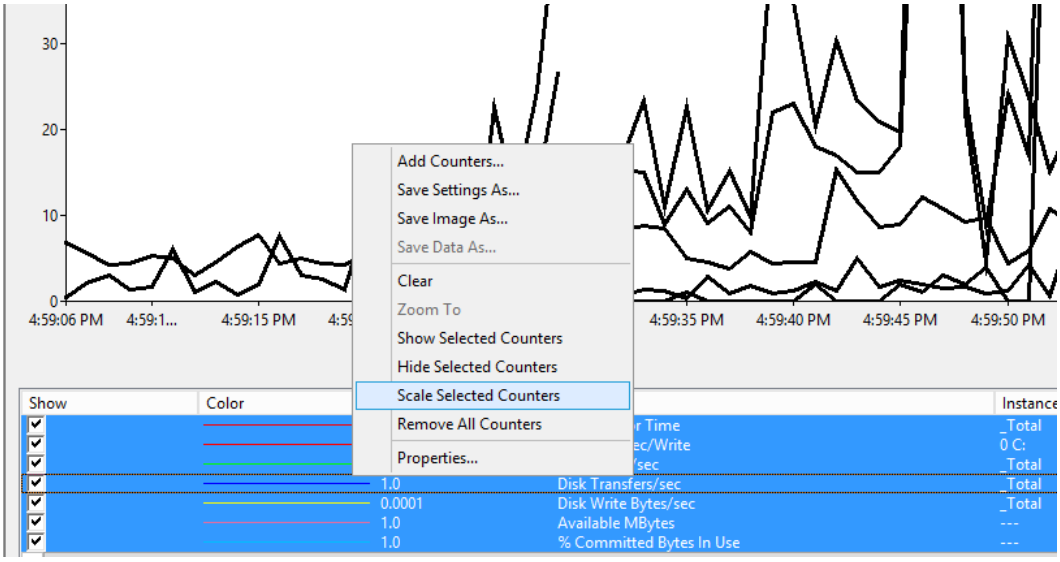


Figure 8-6: Selecting counters you wish to scale and fit onto the same graph

While this allows you to view all counters on the same graph, it's important to understand that due to the scaling, a value may not necessarily be larger than others just because it appears larger on the graph. For example, disk latency for the C drive may appear higher on the graph than the E drive. However, if the counter for C is scaled to 1.0 and the counter for E is scaled to .01, disk latency values for E may actually be much higher. The scaling allowed both values to be displayed in the same simplistic view, but do not directly reflect

measured values. Instead, I recommend letting the scaled graph uncover trends and outliers, while using the actual measured data points to uncover the true values (Figure 8-7).

Figure 8-7: Actual values measured at each data point

Viewing captured data

In most cases, performance data is viewed after the fact. For instance, a support representative has requested a PerfMon capture on the affected server, preferably for the time period when the issue is occurring. When viewing captured data, there is one cardinal rule that any Windows performance expert will tell you; **never double-click a Performance Monitor (.BLG) file**. These .BLG files can be hundreds of megabytes or even several gigabytes, and double-clicking them will almost certainly result in your system hanging. Instead, open PerfMon and click the "View Log Data" button or press Ctrl+L (Figure 8-8).

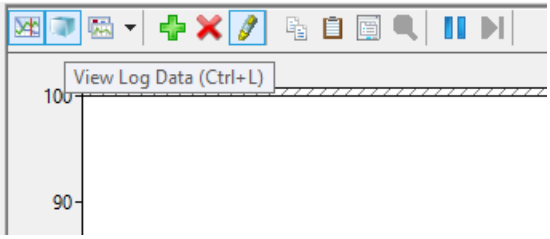


Figure 8-8: The "View Log Data" button in the top-left corner of the PerfMon window

You can then select the .BLG file you wish to view (Figure 8-9). This method is preferred over double-clicking the .BLG outside of PerfMon. With this method no counters will be loaded initially, whereas double-clicking the file will open all counters, resulting in slowness and a visually busy graph.

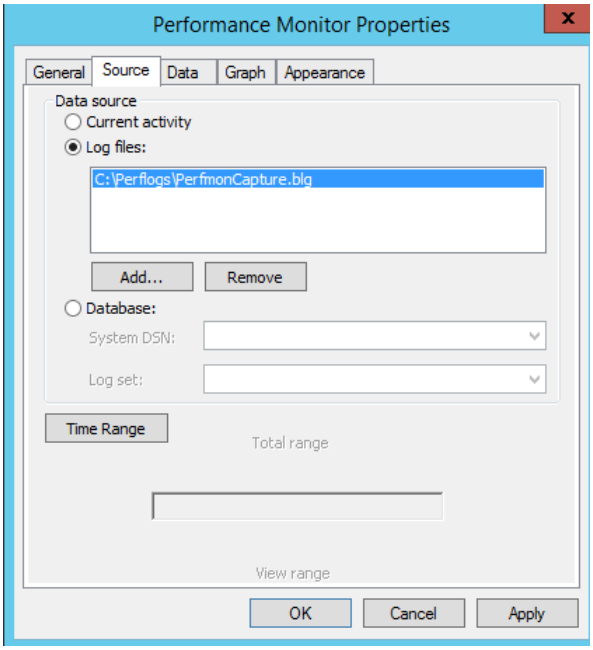


Figure 8-9: Selecting a performance monitor file to view

After the .BLG file is open, the desired counters can be added using the green plus button. The graph will represent the period of time the data capture was running, giving you the ability to zoom to a specific time period (Figure 8-10). This is especially useful if the capture was for a particularly long time period. In some cases, a spike can only be seen on the graph when zoomed into a short time window.

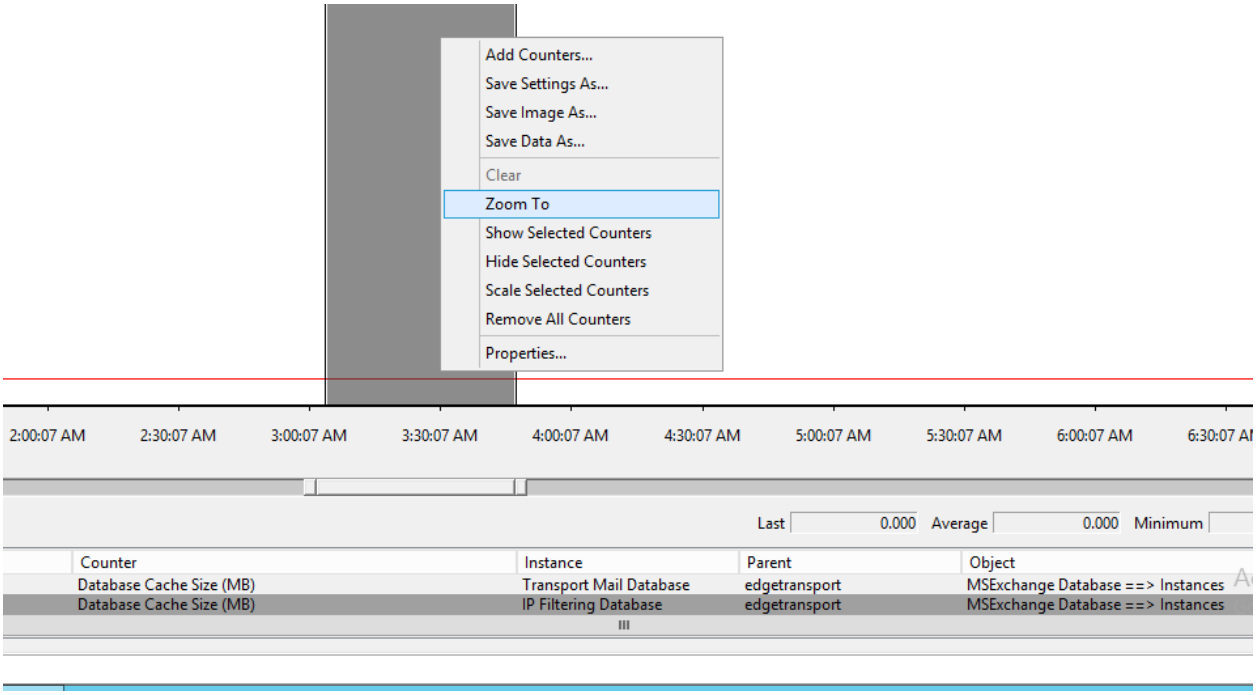


Figure 8-10: Zooming to a specific time period of a PerfMon capture

Capturing data

Performance data can be captured by creating a Data Collector Set, which can be used to define the following parameters:

- Performance counters captured
- Sample interval
- Location of .BLG file
- Schedule
- Duration
- Maximum file size
- Circular Enabled/Disabled
- File Format

While you can certainly create a user defined Data Collector Set using the Performance Monitor GUI, this isn't really practical when adding many counters or requesting someone to gather data for your analysis. However, you can generate an XML configuration file to be used as a template to make this process easier. I prefer using the command line for this operation. The example shown below illustrates how to create a data collector set using logman.exe:

```
C:> logman.exe create counter Perf-Counter-Log -f bincirc -v mmddhhmm -max 2048 -c
"\Cache(*)\*" "\LogicalDisk(*)\*" "\Memory\*" "\NetLogon(*)\*" "\PhysicalDisk(*)\*"
"\Process(*)\*" "\Processor(*)\*" "\Server\*" "\System\*" -si 00:00:05
```

This command creates a Data Collector Set named "Perf-Counter-Log" with a binary circular log that will be a maximum of 2048 MB in size and have a sample interval of 5 seconds. This means the file will grow to 2 GB and then continuously roll over until the Data Collector Set is manually stopped. This is useful if you wish to capture data until an issue occurs, at which point you can stop the capture. The "-c" switch defines which counters we will gather. As an alternative, a configuration file could also be used.

Run the following command to start the capture:

```
C:> logman.exe start Perf-Counter-Log
```

Run the following command to stop the capture:

```
C:> Logman.exe stop Perf-Counter-Log
```

Another option is to use the PowerShell Get-Counter command to gather both local and remote server data. This can obviously be scripted to your heart's desire and is easily scalable using PowerShell Remoting.

There are also two well-known automated methods of Exchange performance log collection:

- Performance Analysis of Logs (PAL)
- Experfwiz

PAL is especially useful as it provides templates for various Microsoft applications, including Exchange, as well as thresholds for each counter collected. It produces an HTML file for easy viewing of collected counters as well as color-coded indicators for when thresholds are exceeded. While it's a great starter tool for the novice, it can sometimes be misleading by presenting a wall of red text. This can happen when spikes occur but are not necessarily indicative of a systemic issue.

Lastly, one extremely underappreciated feature of PerfMon is the ability to save counters to an HTML file which can then be used to paste those same counters into a different PerfMon session on a different server.

Exchange Performance Counters

The architecture of the Exchange product has evolved over the years. Services have been renamed, server roles added/removed, and functionality altered. In turn, the evolution of the product has led to changes in performance counter names and their expected values during normal operation. While common Windows counters related to disk, memory, and processing have remained constant throughout the years, Exchange-specific counters have changed from version to version. Below you'll find a listing of the various counters relevant to Exchange 2013/2016 as well as their expected values where applicable.

> **Note:** Much of these values were taken from the sources cited below. Please use them as a reference for further explanation of the counters.

Exchange 2013 Performance Counters

Tools and Techniques for Exchange Performance Troubleshooting

Exchange Server 2013 Performance Recommendations

Active Directory Connectivity

MSExchange ADAccess Domain Controllers(*)\LDAP Read Time	<50ms on Avg.
MSExchange ADAccess Domain Controllers(*)\LDAP Search Time	<50ms on Avg.
MSExchange ADAccess Processes(*)\LDAP Read Time	<50ms on Avg.
MSExchange ADAccess Processes(*)\LDAP Search Time	<50ms on Avg.
\Netlogon\Semaphore Waiters	See References
\Netlogon\Semaphore Holders	See References

\Netlogon\Semaphore Acquires	See References
\Netlogon\Semaphore Timeouts	See References
\Netlogon\Average Semaphore Hold Time	See References

General Client Access/HTTP

\Web Service(Default Web Site)\Current Connections	See References
WebService(_Total)\Connection Attempts/sec	See References
\MSExchange HttpProxy\Accepted Connection Count	See References
\Web Service(Exchange Back End)\Current Connections	See References
\MSExchange HttpProxy(*)\Outstanding Proxy Requests	See References
MSExchange HttpProxy(*)\MailboxServerLocator Average Latency	See References
MSExchange HttpProxy(*)\Average Authentication Latency	See References
MSExchange HttpProxy(*)\Average ClientAccess Server Processing Latency	See References
MSExchange HttpProxy(*)\Mailbox Server Proxy Failure Rate	See References
MSExchange HttpProxy(*)\Proxy Requests/Sec	See References
MSExchange HttpProxy(*)\Requests/Sec	See References

AutoDiscover

MSExchangeAutodiscover\Requests/sec	See References
\MSExchange HttpProxy(autodiscover)\Outstanding Proxy Requests	See References

RPC

\MSExchange RpcClientAccess\RPC Averaged Latency	<250ms
MSExchange RpcClientAccess\RPC Requests	<=40
MSExchange RpcClientAccess\Active User Count	See References
MSExchange RpcClientAccess\Connection Count	See References
\MSExchange RpcClientAccess\RPC Operations/sec	See References
\MSExchange HttpProxy(rpchttp)\Outstanding Proxy Requests	See References
MSExchange RpcClientAccess\User Count	See References

ActiveSync

\MSExchange ActiveSync\Requests/sec	See References
\MSExchange ActiveSync\Current Requests	See References
MSExchange ActiveSync\Sync Commands/sec	See References

\MSExchange HttpProxy(eas)\Outstanding Proxy Requests	See References

Exchange Web Services

\MSExchangeWS\Average Response Time	See References
\MSExchangeWS\Requests/sec	See References
MSExchange Availability Service\Availability Requests (sec)	See References
\MSExchange HttpProxy(ews)\Outstanding Proxy Requests	See References

Outlook Web App

\MSExchange OWA\Average Response Time	See References
\MSExchange OWA\Average Search Time	See References
\MSExchange OWA\Requests/sec	See References
MSExchange OWA\Current Unique Users	See References
\MSExchange HttpProxy(owa)\Outstanding Proxy Requests	See References

Exchange Admin Center

\MSExchange HttpProxy(ecp)\Outstanding Proxy Requests	See References

MAPI over HTTP

\MSExchange HttpProxy(mapi)\Outstanding Proxy Requests	See References

Offline Address Book

\MSExchange HttpProxy(oab)\Outstanding Proxy Requests	See References

POP3

\MSExchangePop3(*)\Average LDAP Latency	See References
\MSExchangePop3(*)\Average RPC Latency	See References
\MSExchangePop3(*)\Request Rate	See References

IMAP4

\MSExchangeImap4(*)\Average LDAP Latency	See References
\MSExchangeImap4(*)\Average RPC Latency	See References
\MSExchangeImap4(*)\Request Rate	See References

PowerShell

\MSExchangeRemotePowershell\Current Connection Sessions	See References
\MSExchangeRemotePowershell\Current Connected Unique Users	See References
\MSExchange HttpProxy(powershell)\Outstanding Proxy Requests	See References

Network

Network Interface(*)\Packets Outbound Errors	0
Network Interface(*)\Packets Received Discarded	0
TCPv4\Connections Reset	See References
TCPv6\Connections Reset	See References

Information Store

\MSExchangeIS Store(*)\RPC Average Latency	<50ms
\MSExchangeIS Store(*)\RPC Operation/Sec	See References
MSExchangeIS Client Type\RPC Requests	<70ms
\MSExchangeIS Client Type(*)\RPC Average Latency	<50ms
\MSExchangeIS Client Type(*)\RPC Operation/Sec	See References

Memory

\Memory\% Committed Bytes in Use	<80%
\Memory\Available MBytes	>5% of RAM
.NET CLR Memory(*)\% Time in GC	<10% on Avg.

Storage/Mailbox

\MSExchange Active Manager(_total)\Database Mounted	Balanced
\MSExchange Database ++> Instances(*)\I/O Database Reads (Attached) Average Latency	<20ms
\MSExchange Database ++> Instances(*)\I/O Database Writes(Attached) Average Latency	<50ms
\MSExchange Database ++> Instances(*)\I/O Log Writes Average Latency	<10ms
\MSExchange Database ++> Instances(*)\I/O Database Reads (Recovery) Average Latency	<200ms
\MSExchange Database ++> Instances(*)\I/O Database Writes(Recovery) Average Latency	<Reads

Note: For an excellent insight into Windows disk counters, as well as how to measure disk latency, see the below two TechNet posts from my friend and former colleague Flavio Muratore:

- Windows Performance Monitor Disk Counters Explained
- Measuring Disk Latency with Windows Performance Monitor (Perfmon)

CPU

\Processor(_Total)\% Processor Time	<75% on Avg.
\Processor(_Total)\% Privileged Time	(Kernel) <75% on Avg.
\Processor(_Total)\%User Time	<75%

\Process (*)\% Processor Time	Process dependent
System\Processor Queue Length (all instances)	<=5 per processor

ASP.NET

ASP.NET\Application Restarts	0
ASP.NET\Worker Process Restarts	0
ASP.NET\Request Wait Time	0
ASP.NET Applications(*)\Requests In Application Queue	0
ASP.NET Applications(*)\Requests Executing	See References
ASP.NET Applications(*)\Requests/Sec	See References

Workload Management

MSExchange WorkloadManagement Workloads(*)\ActiveTasks	See References
MSExchange WorkloadManagement Workloads(*)\CompletedTasks	See References
MSExchange WorkloadManagement Workloads(*)\QueuedTasks	See References

Exchange Built-In Data Collector Sets

Starting with Exchange 2013, there are two built-in Data Collector Sets which constantly monitor all relevant Exchange performance counters (Figure 8-11). The data collected by these counters is used by Managed Availability in determining the health of Exchange on the local server.

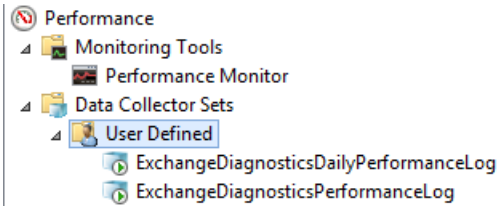


Figure 8-11: Default Exchange Data Collector Sets

The .BLG files generated by these Data Collector Sets are located within the *<Install Drive>\Microsoft\Exchange Server\V15\Logging\Diagnostics\DailyPerformanceLogs* directory. These files, along with accompanying Exchange logs within the *Logging* directory, can consume a considerable amount of disk space. While there are means to purge excess logs, having a historical view of an Exchange Servers' performance can prove quite useful. Within this directory (Figure 8-12) contains 7 days worth of Performance Monitor logging data.

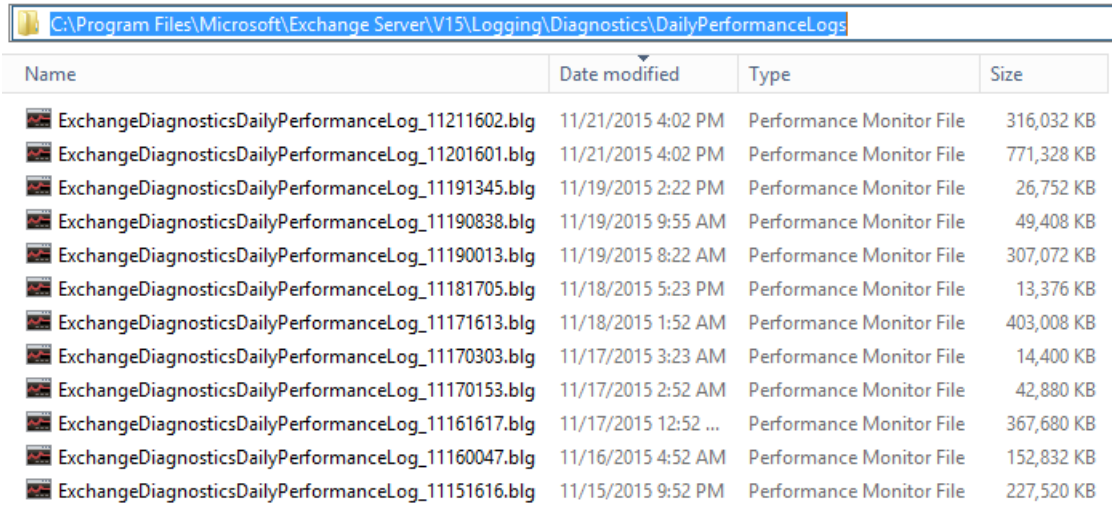


Figure 8-12: PerfMon files containing Exchange-related counter data

Analyzing Exchange Performance Monitor Data

Having discussed Performance Monitor, its usage, and the relevant Exchange performance counters, we now move on to review the proper methodology for using this information to diagnose an Exchange issue.

Identify frequency

Is the issue currently happening? If not, did the performance issue occur only once? If it repeats, is it sporadic or periodic? Knowing these answers will determine the most appropriate counters that you should measure and what data is gathered.

If the issue is currently occurring, then I recommend the following:

- Use Task Manager to Processes Tab to sort by Memory and CPU utilization. Identify the set of Exchange processes and their percentage of overall utilization. Compare this to the Performance Tab to determine overall system utilization. Task Manager does not include disk performance information, so instead use Windows Resource Monitor to measure each processes disk utilization.
- For deeper analysis, open PerfMon and the relevant counters. I recommend starting down one of three paths based on initial analysis:
    - Disk
    - Memory
    - CPU
- Once a hypothesis is made that the issue is related to one of the three aforementioned resources, then add the related Exchange counters

If the issue occurred only once then we're effectively performing root cause analysis, where there is no guarantee of success. This is because success is dependent upon data/logs still being present. Days can pass before an issue is reported, especially if the issue occurred before the weekend or a holiday. It's important to set proper expectations when performing root cause analysis, and let the stakeholders know it may be impossible to determine the cause if sufficient date is not present.

With that in mind, the built-in Exchange Data Collector Sets are extremely useful to view the performance of a system several days in the past. You should first copy the .BLG file to an alternate location before it is deleted

by background Exchange cleanup processes (after 7 days). You can then open Performance Monitor, open the file, and add the Disk, Memory, and CPU counters to begin the analysis. I also recommend immediately exporting both the Application and System event logs, as Windows will purge these files when capacity is reached.

The most critical piece of information for root cause analysis is the timeframe of the initial issue. This will determine if the data you have available to you is of any use. Obviously, if an issue occurs Monday night, logs from Tuesday to Friday are effectively useless for analysis. However, if you have the logs that cover the period in question, knowing the precise window when an issue occurred allows you to investigate the event logs and performance data leading up to the event. Often this data is critical to uncovering the root cause. It is common to find that a backup or Anti-Virus scan will commence immediately before a performance issue, allowing you to "follow the bread crumbs" so to speak and uncover the culprit.

If the issue is repeating (either sporadic at random intervals or periodic at predictable intervals), you'll want as many data captures as possible. A range of data captures will allow you to confirm that the same behavior is observed during each incident as well providing the evidence to detect a pattern. For example, if a particular process is utilizing a significant amount of system resources during each capture or an upward trend in usage is detected, a possible memory leak might be the cause.

A full list of activities (processes) running on the server outside Exchange is also extremely useful when troubleshooting repeating issues. Depending on the environment's architecture, the following should be considered:

- Execution time of backup jobs
- SAN replication schedules
- Anti-Virus full system scan schedule
- Hypervisor maintenance or automation schedules

Start with common counters

As you may have already noticed, I prefer looking at common Windows performance counters first, before moving on to Exchange-specific counters. Having spoken with several Microsoft Support Engineers and Premier Field Engineers, this is a common preference amongst performance troubleshooters. Whether the system is running Exchange, SQL, SharePoint, Hyper-V, or third-party applications, understanding the values of Disk, CPU, Memory, or Network counters is an excellent first step. For example, if Physical Disk Avg. Disk sec/Write (which measures disk write latency) exceed an average of 70ms (20ms or less is considered acceptable), this might lead us to investigate Exchange disk/storage-related counters to determine what Exchange is doing at this time. Although it's possible a hardware or environmental issue is causing the performance issue, Exchange itself could be placing excessive load onto the system. This is where the Exchange counters can prove useful. Issues such as excessive transaction log generation, mobile device bugs, or third-party add-ins can routinely cause Exchange to generate excessive load and burden the server hardware.

The problem with beginning your investigation with the Exchange counters is there are many of them, and their thresholds are not as well-known as the basic Windows counters are. This is why I recommend a monitoring solution to track the proper counters and respond accordingly. Whether this is Microsoft's SCOM, or a third-party solution, is a matter of preference, cost, and functionality. However, in my experience, these typically lead you to Disk, CPU, Memory, or Network regardless. They just may streamline the process and allow you to scale your monitoring and diagnosis much easier.

Look for patterns, trends and anomalies

When you analyze performance counters, the detection of certain patterns within a graph can lead to the identification of specific undesirable behaviors. People often associate an upward ascending line (like a ramp) with a memory leak. More specifically, if you're monitoring the memory utilization of a specific process, it's possible such a behavior could indicate a memory leak which would consume all system memory. However, it's important to note the counter you're looking at when making such a claim. Obviously, the System Uptime counter is expected to have such an upward slope, as uptime increases as long as the server remains running. On the other hand, a downward descending line (like a ski slope) might also indicate a memory leak if the counter is "Memory\Available Mbytes". Another example is that "% Processor Time" in a descending trend means CPU resources are becoming available. These points illustrate the truth of knowledge of the counter being measured makes all the difference in the world when observing its progress. A possible and often overlooked anomaly is seeing "gaps" or absence of data in PerfMon output (Figure 8-13).

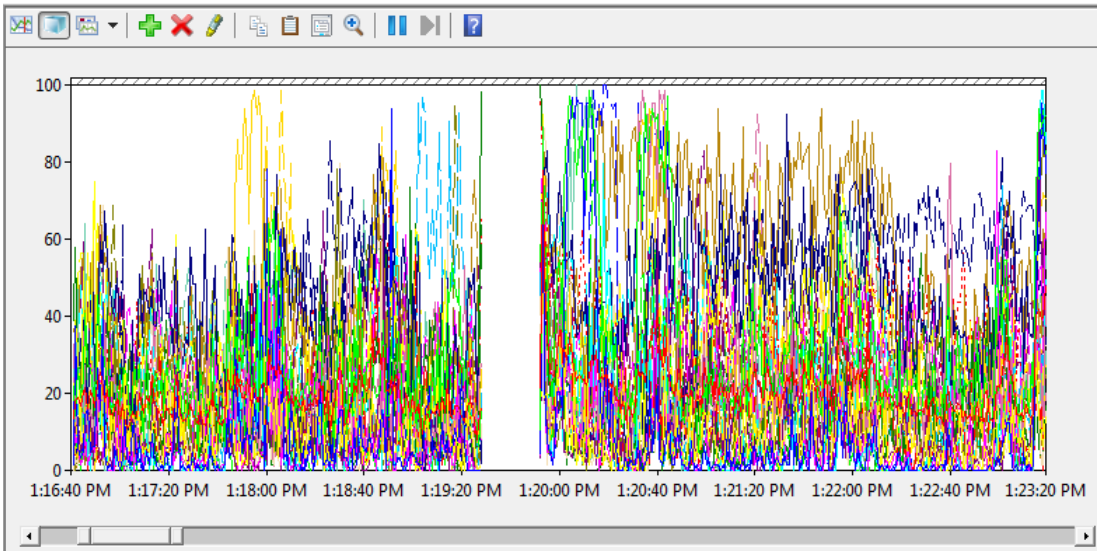


Figure 8-13: A "gap" or period of no data collection in a PerfMon log file

Seeing these gaps is never a good sign and is usually indicative of a severe system performance issue. The above gap lasted approximately 20 seconds and was not even visible when first opening the .BLG file. This is common when you have a capture which is particularly large and covers a long period of time. Only upon closer inspection (and zooming into a span of <10min) can we see this gap, which means that during this 20 second period, the Operating System was unable to log any performance monitoring data, likely due to it being woefully short on system resources. In my experience, these gaps can be caused by:

- Insufficient CPU resources for the Operating System to log performance data
- Insufficient Memory resources for the Operating System to log performance data
- Period of high Disk IO causing a resource issue

Scenarios which can lead to this are:

- Backup software causing a system to hang during a backup operation
- Excessive client access load being placed on the server
- Working set trimming operation causing a system hang

The effects of these gaps are typically:

- Complete Operating System hang
- Network packet loss due to insufficient resources

- Cluster failovers resulting from loss of network connectivity
- Period of heavy disk activity as all pending/quiesced IO flushes to disk after the gap
- Blue Screen of Death from the system itself or Exchange Managed Availability via a specific responder

CPU considerations and Hyper-threading

Having already discussed CPU sizing considerations for Exchange, let's now discuss a few important CPU configuration considerations that can have a significant impact on performance.

Receive Side Scaling

Receive Side Scaling (RSS) is a networking technology which allows network processing to be distributed amongst processors. If this feature is disabled, network processing will primarily occur over the first CPU (CPU 0), leading to potential network bottlenecks or dropped packets. RSS should be enabled by default, both in Windows and for the Network Interface Card (NIC). However, due to past issues with the Windows Scalable Networking Pack and NIC drivers/firmware, many adopted the habit of disabling these features by default. Or, at the first sign of system performance issues they would disabled the features regardless of whether they were actually the cause. With this feature disabled, network performance can be significantly impacted, which is especially impactful for Exchange Database Availability Groups that rely heavily on the health of the network stack. See the below articles for references on this issue.

- Network Settings, Network Teaming, Receive Side Scaling (RSS) & Unbalanced CPU Load
- Time to revisit recommendations around Windows networking enhancements usually called Microsoft Scalable Networking Pack
- A reminder on real life performance impact of Windows SNP features
- Processor 0 increased CPU utilization

Hyper-threading

Hyper-threading is a simultaneous multithreading technology with the goal of improving processing performance. In simplistic terms, for every physical processor core, hyper-threading allows the appearance of an additional logical core. However, while the Operating System views two logical processors, only one physical processing core exists with a singular set of execution resources.

While going through VCP training, my instructor used an analogy that's stuck with me for a while. People have this misconception about hyper-threading that it's actually giving you twice the cores, when in fact it is not. Say you're eating food with one hand. You're losing the opportunity to put more food in your mouth in the time that it takes to reach down and grab another bite. Now if you could eat with both hands (enabling HT) then during that time when one hand is getting more food, the other hand could be placing food in your mouth. However, you still ultimately have only one mouth to eat with. And most importantly, the faster you're eating (moving your hand back and forth) then the less benefit that other hand is giving you. So on a server, the more utilized the processors are, the less benefit HT will give you. So while it's not accurate to say that hyper-threading gives you twice the cores, it can increase performance in some situations.

Historically, the risk of enabling hyper-threading was typically associated with virtualization. Even then, it wasn't a technical limitation but more of an operational one. People incorrectly assume they have twice the processing power than they actually possess, which results in rampant CPU oversubscription which ultimately leads to. This should not be an issue in an operationally mature environment which is properly sized. However, the Exchange Product Team recommends against enabling hyper-threading on physical Exchange servers. The reasoning is that the processing gains were not worth the costs. By cost, they are referring to memory allocations per logical processor via .NET.

As a quick reference, terms like Virtual Processor, Logical Processor, and Physical Processor are sometimes mistakenly used interchangeably. Here are brief descriptions of each, in my own words:

- Physical Processor: Physical processing core per chip/socket
- Logical Processor: Logical threads per ship/socket
- Virtual Processor (vCPU): Assigned virtual cores when virtualizing

Example: The Intel i7-5960x processor has 8 Physical Processor cores, 16 Logical Processor cores (with HT enabled), and if the system with this processor were to be made a hypervisor, it could assign up to 16 Virtual Processor cores per virtual machine (VM).

Understanding these concepts is vital in both sizing as well as when troubleshooting CPU performance. It's important to understand how to recognize CPU contention in virtual environments, as well as understand how hyper threading can potentially impact physical deployments as well.

Exchange 2013/2016 and .NET

During development of Exchange 2013, it was decided to re-write much of the code base into managed code, meaning Exchange 2013 became a .NET application. This change effectively means that a healthy Exchange installation depends upon a healthy installation of .NET. There have been a few instances (additional reference) where Exchange performance was impacted by an underlying issue with .NET. With this knowledge, understand that it's extremely important to run Exchange on a supported version of .NET.

> **Note:** .NET updates cause Windows to recompile all the Exchange assemblies. This affects CPU performance for up to 20 minutes after reboot. See this article by Exchange MVP Jeff Guillet for a description and a script to help speed this up by making the recompile multi-threaded.

Exchange Virtualization

Exchange virtualization is an extremely complicated topic. And not just for technical reasons, but also political, economic, and preferential reasons. For the sake of time and the conservation of space, I'll tighten my focus to matters relevant to troubleshooting virtualized Exchange servers and avoid the discussion on whether I feel Exchange should or should not be virtualized (although a simpler solution is always easier to troubleshoot, and virtualization is an added complexity).

My reasons for including Virtualization in the Performance chapter is simple; numbers. In troubleshooting, there are purely Exchange issues and purely virtualization issues. A Venn Diagram of Exchange and Virtualization issues would intersect with many Performance issues. In my opinion, the majority of these issues could be avoided with proper sizing and adherence to best practices. With that in mind, let's briefly discuss Exchange virtualization best practices.

Additional References:

- Best Practices for Virtualizing and Managing Exchange 2013 (Microsoft)
- Exchange 2013 on VMware Best Practices Guide (VMware)

Memory

Hyper-V Dynamic Memory is not supported in production for Exchange Servers. However, it can be used in a lab environment. The reasoning for this is simple enough in that an operating system running an application with a transactional database (Exchange) should never have memory taken away from it while the application

is still running. This is especially true for Exchange as the sudden reduction in available memory can result in significant performance issues, such as specific Exchange processes forced to reduce their working set, or a significant flushing to disk of transactions formerly held in database cache. For these reasons, an Exchange virtual machine must use Static Memory when running on Hyper-V. This means when the virtual machine starts it will have the same amount of RAM until it powers off. In effect, memory simulates what is found on a physical machine.

If the Exchange virtual machine is being hosted on a VMware hypervisor (ESX), a similar memory management rule must be enforced (using different terminology). A production Exchange virtual machine must have a 100% Memory Reservation.

CPU

There are many well-written articles on CPU sizing when virtualizing, most of which are from the VMware perspective:

CPU Over commitment and Its Impact on SQL Server Performance on VMware

Virtual Machine Performance – CPU Ready

How to successfully Virtualize MS Exchange – Part 1 – CPU Sizing

Hyper-V CPU Scheduling–Part 1

> **Note:** Much of this module is taken from my blog post entitled CPU Contention and Exchange Virtual Machines

All of these articles provide excellent advice on the topic. In summary, in vendor-neutral terminology, on a given host, the total number of assigned processor cores on all of your virtual machines can potentially be greater than the total number of actual cores on the physical host.

Example:

On a virtual host with two Intel Xeon E5-2640 2.5GHz CPU's, each with 6 cores (hyper threading disabled for this example), you have a total of 12 physical cores. If you host three virtual machines, each with 4 virtual cores, then you would have assigned 12 total cores, or a ratio of 1:1 (virtual cores : physical cores). If you increase the number of cores per VM from 4 to 8 then your ratio changes to 2:1 and at least some level of CPU contention is introduced.

> **Note:** You should also assign processor cores to the Hypervisor OS, but for this example I excluded that measure for simplicity.

Again, in vendor-neutral layman's terms, when your ratio exceeds 1:1 (2:1, 3:1, 4:1, etc.) the likelihood that a virtual machine will have to wait for a physical core to become available for its use increases. Check out the above links for detailed explanations of how this happens.

Is overcommitting CPUs good or bad? It depends. In many cases, overcommitting CPUs is perfectly acceptable and recommended. In fact, I'd say it's one of the biggest advantages of virtualization. However, overcommitting CPUs is dependent on the workload and solutions like VDI are capable of functioning perfectly happily beyond 12:1 (depending on which vendor you speak to) while still achieving acceptable performance.

But in the Exchange world, Microsoft makes their stance pretty clear. They support a ratio no greater than 2:1 of virtual processors to physical cores but 1:1 is strongly recommended. VMware echoes this advice in their guidance. Also, hyper-threading can be enabled on the host, but when sizing you should only take into account the physical cores. In short, size as if HT is not enabled.

If this guidance is not followed, the following symptoms might occur on virtual Exchange Servers:

- Messages may sit in the mail queue longer than expected
- Responsiveness of OWA may be negatively impacted
- "Trying to connect…" message in Outlook client
- General slowness or unresponsiveness (hang) of the Management Tools and Operating System
- "Time gaps" in performance monitor logs
- The VM does not produce network or disk traffic
- The VM console is unresponsive or hangs
- Database failovers or BSOD of Exchange Servers in extreme cases
- The VM is unresponsive over the network

The most interesting thing about troubleshooting a system that's experiencing CPU contention is that when viewed from within the VM itself, CPU utilization may be minimal while things are painfully slow. Much of this will depend on the load of the host, workload within your VM, and the sizing of the other VMs. A few very dense VMs, or many small VMs with only 1-2 vCPUs can have differing behaviors.

The important thing to remember is not to jump to conclusions and start adding cores because you think a virtual machine needs them. If you've been paying attention, you'll understand that this solution will only make matters worse as the VM may have to wait even longer for its additional vCPUs to be scheduled against physical cores that are heavily utilized. Simply put, if your CPU is fully utilized within the VM, then you might look at increasing vCPUs. If you suspect processor but the vCPU within the VM is not fully utilized, then investigate possible CPU contention.

It's important to realize how easily this can happen without proper operational controls. I once encountered a support case where an Exchange 2010 customer experienced a period of heavy mail flow but the mail queues were not draining. The root cause was when a junior virtualization admin moved several dense VMs to the same host as Exchange and caused the CPU ratio to move from a 2:1 core ratio to 6:1. Massive CPU contention caused the problem.

Storage

Many of the principles of troubleshooting Exchange storage transfer to troubleshooting Exchange virtualized storage. The sweet spot for disk latency remains <20ms, vendor best practices for caching update levels should still be followed, and Jetstress must still be run to validate the storage. The difference lies in additional layers of complexity, failure points for performance, and storage virtualization technologies being used, such as Thin Provisioning, Snapshots, Dynamic Disks, and Tiered Storage. Several of these technologies are not unique to virtualization. For example, you can have physical servers and utilize thin provisioning or tiered storage. However, as these technologies are typically utilized in virtualization scenarios I'm addressing them now. Let's discuss how each technology can "go wrong" or be encountered in an Exchange troubleshooting scenario.

Thin Provisioning: Using this technology, you could potentially have a scenario where an Exchange volume exhausts available storage, while Exchange and the Operating System believes free space is still available. This can initially result in transport issues due to Back Pressure and at worst, an Exchange database dismounting in a Dirty Shutdown state. This can be particularly harmful as Exchange has built-in mechanisms to gracefully dismount its databases when it detects available disk space is nearing zero. In some thin provisioning

scenarios an Operating System could believe there is over 1 TB of free space on a drive, when the backend storage solution has been completely allocated. The key here is to fix this problem before it ever happens by closely monitoring your thin provisioned storage to enable additional capacity to be added before an outage occurs.

Virtual Machine Snapshots: VM snapshots are not supported in production for Exchange virtual machines. This is primarily due to the many Active Directory tie-ins with Exchange. If an Exchange VM diverges from the Active Directory configuration database after a snapshot is used to revert the system, the Exchange organization will be in an inconsistent state. Also, if the reverted Exchange VM is a member of a Database Availability Group, its database copies will probably require to be reseeded.

Dynamic Disks (Dynamic VHDX): Until Exchange 2013 and Server 2012 R2, Dynamic VHDX files were not supported for Exchange in production. However, due to write performance improvements, this is now a fully supported option. While VHX is supported, I recommend scheduling any expansion of disks to off-peak hours to help mitigate the performance impacts.

Tiered Storage: This technology has different names depending on the vendor (Tiered Storage, Data Progression, FAST, Virtual Storage Tier, Hierarchical Storage Management, etc.). While it is supported, if data is automatically moved by storage software, it is not recommended by the Exchange Product Team for Exchange storage. The most common time when issues occur is during Jetstress validation. If you choose to use tiered storage with Exchange, then the recommendation is to use LoadGen instead of Jetstress to validate the storage configuration. However, if you must run Jetstress, engage your storage vendor as it will likely require custom configuration for the test to pass. In troubleshooting, if you experience latency when running Exchange on tiered storage, consider moving it to static/dedicated disks where the data blocks will not be moved. If this is not an option, consider increasing Tier 1 storage capacity so more Exchange data can be held within Tier 1.

Exchange Diagnostic Logging

A quick note on Exchange Diagnostic Logging. This is the feature that automatically gathers information about different parts of the product (including the protocols used). The information is valuable and not only for troubleshooting performance issues. The parameters to control diagnostic logging can be set via Exchange Management Shell or the Exchange Management Console in 2007/2010. The data are not placed into any particular folder or file, but instead are in the Windows Event Viewer. By default, all processes are set to "Lowest", which generates the level of Event Viewer Application logging seen on every Exchange Server upon installation. These can be changed to one of the following values:

- Lowest
- Low
- Medium
- High
- Expert

When experiencing an issue with a particular process in Exchange but the Application logs do not provide sufficient details, the logging level can be modified to hopefully shed additional light on the cause.

Additional Tools

Fiddler: Useful for capturing and analyzing HTTP traffic

- Tools and resources for troubleshooting EWS applications for Exchange

- Unified Communications with Microsoft
- Using Fiddler to troubleshoot HTTP protocol issues on Windows Phone
- Using fiddler to troubleshoot address book download issues
- Announcing Fiddler Inspector for Exchange Server ActiveSync

Exchange Server User Monitor:

- Exchange Server User Monitor
- Using the Microsoft Exchange Server User Monitor (ExMon) tool
- Exchange 2013 and 2016 Exmon tool is now available

Log Parser Studio:

- Introducing: Log Parser Studio
- Log Parser Studio 2.0 is now available

Network Monitor:

- How to take Network traces between clients and Exchange servers

Exchange Workload Management: Throttling of Exchange tasks based on system resources

- Exchange Workload Management

Windows Storport Tracing: Measuring latency of individual IO at the lowest point in the Windows IO stack. Can be used to bypass filter drivers in Windows such as Anti-V or backup software.

- Deciphering Storport Traces 101
- Storport ETW Logging to Measure Requests Made to a Disk Unit
- Tracing with Storport in Windows 2012 and Windows 8

Additional reading

Overview

- IOPS
- Correlation does not imply causation

Validate the solution and use validation data for troubleshooting

- Announcing the Jetstress 2013 Field Guide
- Extensible Storage Engine
- Loadgen 2013
- Released: Exchange Server Role Requirements Calculator
- Windows Performance Monitor Disk Counters Explained
- I/O Request Size
- Understanding I/O: Random vs Sequential
- Geek Out with Perry - Managing Storage Efficiently
- Hard disk drive nearline storage
- Serial ATA
- Serial attached SCSI
- Ask the Perf Guy: How big is too BIG?
- Garbage Collection

- Monitoring and Tuning Microsoft Exchange Server 2013 Performance
- Hyper-threading
- Exchange 2016 system requirements
- Troubleshooting High CPU utilization issues in Exchange 2013
- Processor states

Understanding controller caching and Exchange performance

- Storage Configuration Options for Exchange 2013
- JBOD
- Exchange Native Data Protection
- Exchange Server 2013 Architecture
- High Availability and Site Resilience De-Mystified
- The Preferred Architecture
- Exchange 2013 storage configuration options
- Disk array controller
- Disk cache
- Storage Spaces Overview
- Disk buffer
- Uninterruptible power supply
- Lost Flush

Using Performance Monitor

- Windows Performance Monitor
- System Monitor
- Object and Counter Design
- Create a Data Collector Set to Monitor Performance Counters
- Using A PAL Template To Easily Capture Performance Data
- Two Minute Drill: LOGMAN.EXE
- Using PowerShell To Gather Performance Data
- Retrieving Packets Received Discarded Perfmon Counter From Multiple Servers
- Windows PowerShell: Dive Deep into Remoting
- Performance Analysis of Logs (PAL) Tool
- Experfwiz (Exchange Performance Data Collection tool)
- Performance Monitor Tips and Tricks

Exchange Performance Counters

- Exchange 2013 Performance Counters
- Tools and Techniques for Exchange Performance Troubleshooting
- Exchange Server 2013 Performance Recommendations
- Windows Performance Monitor Disk Counters Explained
- Measuring Disk Latency with Windows Performance Monitor (Perfmon)

Exchange Built-In Data Collector Sets

- Exchange 2013 diagnostic and performance log files
- Managed Availability
- Exchange 2013 Logging and Space Requirements
- Removing Old Exchange 2013 Log Files

Analyzing Exchange Performance Monitor Data

- Task Manager (Windows)
- Resource Monitor
- Archive an Event Log
- Analyzing Exchange Transaction Log Generation Statistics
- Exchange Server 2013 Management Pack Guide
- Memory Management (Working Set Trimming)
- Nodes being removed from Failover Cluster membership on VMWare ESX?
- Blue Screen of Death
- Trauma for Exchange 2013 servers when Managed Availability goes bad
- Exchange Server 2013–0xEF BSOD

CPU considerations and Hyper-threading

- Introduction to Receive Side Scaling
- Information about the TCP Chimney Offload, Receive Side Scaling, and Network Direct Memory Access features in Windows Server 2008
- Network Settings, Network Teaming, Receive Side Scaling (RSS) & Unbalanced CPU Load
- Time to revisit recommendations around Windows networking enhancements usually called Microsoft Scalable Networking Pack
- A reminder on real life performance impact of Windows SNP features
- Processor 0 increased CPU utilization
- Hyper-threading
- Troubleshooting a virtual machine that has stopped responding: VMM and Guest CPU usage comparison
- Ask the Perf Guy: Sizing Exchange 2013 Deployments
- Plan it the right way - Exchange Server 2013 sizing scenarios
- CPU Contention and Exchange Virtual Machines
- Managed code
- .NET Framework
- Tuning .NET for the Exchange 2013 Information Store - come to Exchange Connections to learn more!
- Exchange 2013 and .NET 4.5 fixes KB2803754 & KB2803755
- Exchange 2016 system requirements
- Be Careful Installing .NET 4.5.2 Update on Exchange Servers

Exchange Virtualization

- Best Practices for Virtualizing and Managing Exchange 2013 (Microsoft)
- Exchange 2013 on VMware Best Practices Guide (VMware)
- Hyper-V Dynamic Memory Overview
- Understanding VMware Memory Resource Management
- CPU Overcommitment and Its Impact on SQL Server Performance on VMware
- Virtual Machine Performance – CPU Ready
- How to successfully Virtualize MS Exchange – Part 1 – CPU Sizing
- Hyper-V CPU Scheduling–Part 1
- CPU Contention and Exchange Virtual Machines
- Thin provisioning
- About Virtual Machine Snapshots
- Windows Server 2012 R2 Hyper-V: What's New

- Tiered storage
- Back pressure
- Exchange Database Is in a Dirty Shutdown State
- Be careful with VM snapshots of Exchange 2010 servers
- Exchange Storage for Insiders: It's ESE
- Plan it the right way - Exchange Server 2013 sizing scenarios

Exchange Diagnostic Logging

- Manage Diagnostic Logging Levels

Chapter 9: Troubleshooting Backup and Disaster Recovery

Andrew Higginbotham

Few would argue against the view that email is a business critical application for almost every company. For this reason, understanding how to recover from disaster, both minor and major, while restoring both data and functionality, is a critical skillset for Exchange Administrators and Support Engineers. Consultants are often asked to provide guidance on backup, restoration, and disaster recovery procedures too, making these extremely valuable skillsets for all who work with Exchange.

In this chapter, we first discuss common Exchange backup methodologies and the support considerations they imply. This will help us understand the scenarios we will work through and the tools needed to ensure that we can recover from disasters. Next we'll discuss Exchange database internals, logging, and how truncation occurs for transaction logs, including why this might or might not happen following a backup. Understanding transaction logging will also prove useful for later modules covering Recovery Databases and Lagged Database recovery. We'll then discuss database corruption and the tools/techniques used for recovery. Corruption can be caused for many reasons, including environmental factors such as storage or power failures, or possibly even misbehaving software such as Anti-Virus or backup software. In any case, knowing how to recover from these scenarios is useful for troubleshooting scenarios and for everyday operations of Exchange. Lastly, we'll cover how to install servers in disaster recovery scenarios and why you might need to take this action.

Backup Strategies

Exchange Native Data Protection

Exchange Native Data Protection is a major piece of Exchange core functionality and highlighted in Microsoft's Preferred Architecture. It is designed to protect Exchange data without relying upon traditional backups. Although the idea of not backing up Exchange was shocking for some and was initially met with cynicism, it's the way that things work in the largest Exchange deployment in the world (Exchange Online within Office 365). That being said, not every company is in the unique circumstances of having the knowledge, software, and hardware resources available to Microsoft to run Office 365.

Exchange Native Data Protection is not a singular product or feature. Rather, it is a framework of interconnecting Exchange features, most of which can be deployed without any dependency on one of the other features:

- Database Availability Group
- Single Item Recovery
- Lagged Database Copies
- Recovery Database
- In-Place Hold

Knowledge of how these features work is required if you are to design or support an environment which uses Exchange Native Data Protection. The DAG provides redundancy for databases, allowing up to 16 copies of

each database (although 3-5 is more common) on servers placed in one or more datacenters that can, in turn, be geographically dispersed. If one database copy or one datacenter fails, an up-to-date copy of that database can be mounted automatically on another server with little to no data loss.

Single Item Recovery (SIR) ensures that items which have been deleted/purged by end users can still be recovered up to the DeletedItemRetention property of a mailbox database. For instance, if the value of DeletedItemRetention is set to 365 days, no matter what action a user takes on a message, it can still be retrieved from the Recoverable Items folder for 365 days after deletion, even if the user selects and then purges the item using the Recover Deleted Items feature. SIR therefore avoids the resource-intense need to restore an entire database from backup media, just to recover one item. Single Item Recovery also allows preservation and recovery of mail in its original form, part of what provides Exchange with the ability to claim that its data is stored in an immutable format.

Normally, the replication that occurs within a DAG ensures that database copies are kept up-to-date as close as possible in terms of content to the active database copy. Lagged database copies represent a moving point-in-time of the data in the database. This is accomplished by establishing an artificial delay on when a database's transaction logs are replayed into the database's .edb file.The replay queue for a lagged database copy always contains the transaction logs that comprise the set of transactions that have taken place during the replay lag period. In effect, the lagged database copy is a snapshot of what the database looked like at the replay lag period (up to 14 days), thus providing the answer to the question of "how do we recover from logical corruption events?"

A DAG provides protection from physical database corruption and hardware failure events and is able to repair physical corruptions while active using techniques such as single page repair (Page Patching). Logical corruption falls outside the capability of these repair mechanisms. For instance, a virus corrupts data in mailboxes across the environment and the corruption is replicated across all database copies. In this case, the normal database copies are all corrupt and cannot be used to recover a good copy. However, a lagged database copy will not have replayed the transactions because, although waiting in the replay queue, the transaction logs containing the bad transaction lie outside its replay lag period. The database therefore remains good and is a viable candidate for recovery, including the ability to replay transaction logs up to the time when corruption occurred. The process for recovering via a lagged database will be covered later in this chapter.

Note: The analogy I like to use when discussing Exchange physical vs logical corruption is that of a damaged book. If a book's pages are torn out and it's binding is damaged, I compare that to physical corruption in a database. Now once the book's pages and binding have been repaired (with all pages now in the correct order), it could still have logical corruption if the words on the pages don't make sense to the reader, if the letters are smudged, the words are out of order, or it's now in the wrong language.

A Recovery Database is similar to a Recovery Storage Group used in older versions of Exchange. Traditionally, this database is used as a location for restoring Exchange database files from an Exchange-Aware Backup. The restored database could then be mounted and the data restored as needed. This method was commonly used when needing to recover mail items that had been deleted or during various disaster recovery scenarios. Making use of Recovery Databases will be covered later in this chapter.

In-Place Hold is an evolution of Litigation Hold. Although both types of hold allow data to be kept for an indefinite period, an in-place hold can apply a filter to make the hold more granular so that only important data is kept. Either hold type can replace the need to retain backup tapes for extended periods for compliance purposes.

Traditional Backup Methods

When discussing Exchange backups, we typically refer to the backup of Exchange databases (although it's also important to have a backup of the installed certificates and their private keys). An Exchange database is comprised of the following files:

- Database file (.edb)
- Transaction Logs (.log)
- Checkpoint File (.chk)
- System/Reservation files (.jrs)

In my opinion, the most important distinction to understand in regard to Exchange backups is the difference between Exchange-Aware backups (which are supported) and non-Exchange-Aware backups (which are not supported). To quote the TechNet article on this topic:

"Exchange 2013 supports only Exchange-aware, VSS-based backups. Exchange 2013 includes a plug-in for Windows Server Backup that enables you to make and restore VSS-based backups of Exchange data. To back up and restore Exchange 2013, you must use an Exchange-aware application that supports the VSS writer for Exchange 2013, such as Windows Server Backup (with the VSS plug-in), Microsoft System Center 2012 - Data Protection Manager, or a third-party Exchange-aware VSS-based application."

In simple terms, an Exchange-Aware backup program understands Exchange ESE databases and can back up the data in a consistent manner without data corruption. An Exchange-Aware backup also has the ability to inform Exchange that it may truncate unneeded log files after a backup successfully completes.

> **Note:** A Full or Incremental backup is required for the Exchange transaction logs to truncate.

These capabilities are not present in backups which are not Exchange-Aware, also commonly known as "Flat-File" backups. Flat-File backups are great when used with file servers or raw static data, as they simply make a copy of a file without any concern for what type of file it is or whether it's open and in active use. Unfortunately, such a backup is mostly useless (and unsupported) for backing up Exchange databases. This is because the backup has no understanding of Exchange database operations and will not have a means to freeze Exchange IO since there is no Exchange VSS Writer.

A common Support case is when a customer is in dire need of a good Exchange backup after a failure event, only to discover that the backup files are in a corrupt or Dirty Shutdown state, with no easy means of mounting the database. While Windows Server Backup has an Exchange VSS Writer and is therefore Exchange-Aware, many third-party backup technologies do not. I've worked several escalations where a customer used a non-Exchange-Aware program and resulted in production mailbox database corruption. You should avoid at all costs any program that attempts to lock Exchange database or pausing IO without understanding Exchange ESE storage.

> **Note:** Many third-party products have Exchange Backup Agents that require additional fees or subscriptions. Always consult with your backup vendor to ensure their backup method is supported by Microsoft.

Apart from asking the vendor a direct question, an easy means to determine whether a backup is Exchange-Aware is to monitor the Windows Application logs during and after the backup to verify events from sources MSExchangeIS and ESE. The presence of these events is an indication that the backup software is communicating with the Exchange Information Store to coordinate the backup and send the command for log truncation once the backup is complete. A set of articles is listed at the end of this chapter containing additional references about how to monitor Exchange-Aware backups and verify their completion.

Another simple command to inspect the last time a backup was performed against an Exchange database is to use this command:

```
[PS] C:\> Get-MailboxDatabase -Server <ServerName> -Status | Format-List Name,*FullBackup
```

I've encountered many customers (usually small businesses with untrained IT staff) who claim to have been successfully backing up Exchange for years, but by examining the last good backup time for the databases I can determine only a flat-file backup has been occurring. This means their backups contain databases in an inconsistent state which may or may not possess the ability to be mounted for recovery.

This scenario is typically discovered (if not during a recovery scenario) when the customer realizes Exchange transaction logs are consuming all of their disk space. Exchange Transaction Logs are 1 MB in size and can grow in the hundreds of thousands if good backups are not taken and the log set is not truncated. However, to fully understand Exchange transaction logs, we must first have an understanding of Exchange database internals.

Exchange Database Internals

This might seem to be a complicated topic but we're going to briefly discuss database internals from a very practical perspective.

Regardless of which Exchange client is used, after connecting through the various Exchange services and worker processes, a user ultimately accesses their mailbox via the Information Store process. The Exchange Information Store (formerly the Store.exe process, but with Exchange 2013 onward, Microsoft.Exchange.Store.Worker.exe) is where the Exchange Database Cache exists. A separate worker process runs for every database on a server. The cache holds all currently active Exchange database transactions in memory, such as new mail, deletions, modifications, etc. This cache can become quite large but this is by design as keeping transactions in memory is much faster than fetching and updating database pages from disk. When the cache does read/write to the database (.edb), it does so in 32 KB pages.

Note: One of the biggest contributors to the IOPS reductions between Exchange 2003 and Exchange 2010 was the increase in database page size. Page size was 4 KB in Exchange 2003, 8 KB in 2007, and 32 KB in 2010 onwards. A larger page size translates to fewer requests to disk, as more data can be read/written per IO, but requires more RAM.

It's important to understand that clients connect to the cache, not the actual .edb file. No client ever directly accesses the database (.edb) or any log files (.log). Instead, all connections occur in memory to the database cache. Of course, if the database (.edb) file becomes locked due to another process (such as anti-virus or backup programs), the Information Service will eventually be unable to communicate with it and the database will eventually dismount.

When discussing backups, transaction logs are extremely important, so it's vital to understand which role they play in database transactions. As transactions occur in the cache, they create a series of transaction records or log data, which fills a log buffer. Once a log buffer is full (1 MB), it is flushed to disk to create a 1 MB transaction log file. This process repeats as more transactions occur in cache and the log buffer fills and is written to disk. The currently active transaction log file is always E0n.log, where n represents the number of the log stream for that database.

Note: The first database on a server will be E00.log, the second will be E01.log and so on. Once the current log file is committed to disk, it is renamed to a value such as E0000000001.log,

E0000000002.log and so on. These log files are in Hexadecimal, so as an example, E0000000009.log would be followed by E000000000A.log.

As transaction log files are written to disk, the transactions in cache might yet still not be committed to the actual database (.edb) file. This process is referred to as "write-ahead logging" (In fact, technically the transactions are written to the logs before the user sees the change). This is because writes to the database are random IO while creating a transaction log is a sequential IO. Since the data being written to the database could be located anywhere in a very large .edb file, the operation on disk is usually random. On the other hand, creating a 1 MB transaction log takes a single new sequential write operation to disk. With rotational media, this distinction becomes important as Seek Time contributes to disk latency. Sequential IO is very low impact on a disk subsystem while random IO is more burdensome. Writing the transactions sequentially to logs instead of the database (.edb) file allows the transactions to be captured efficiently (in cache and on disk in the transaction log) while also reducing random IOPS. The focus on trading random IO for sequential IO by using memory has contributed to the gradual reduction in the product's IO requirement since Exchange 2007.

When are the transactions written to the database (.edb) file? After a predetermined amount of transaction logs are generated, the transactions in cache are flushed to the database (.edb) file. The predetermined amount is called the Checkpoint Depth Target and is tracked by the Checkpoint File (.chk). The Checkpoint File monitors the logs with the oldest outstanding uncommitted page. Databases which have no replicas/copies have a Checkpoint Depth Target of 20 transaction logs whereas databases with passive copies (DAG) have a target of 100 transaction logs. This fact will become relevant when I discuss log truncation, especially in a DAG.

Transaction Logging and Recovery

Of course, there is another reason that Exchange writes transactions to the log files before being committed to the database (.edb) file. By quickly committing transactions to disk (via transaction logs), it means that the transactions exist in two locations; memory and disk. If a failure occurs that causes the Information Store to terminate unexpectedly, the most recent transactions are still available and can be replayed into the database from the transaction logs after the database is mounted and to bring the database up-to-date.

For a long time, it was recommended to place your Exchange database file (.edb) onto different spindles than your transaction logs. This is still the recommendation when only one copy of a database exists (non-DAG protected). In fact, this is not for performance reasons but to assist recovery in the event of a storage failure. Say you lost your database drive, leaving you only with the transaction logs. Technically, if you still had every transaction log present since the database was first created, you could use ESEUTIL to generate a new database and commit every transaction from the logs into it. However, this is not usually the case. People usually resort to an Exchange-Aware backup, which leads us to why an Exchange-Aware backup truncates log files. When a Full Exchange-Aware backup is performed against a database, the .edb file is copied to the backup location, as well as any transaction logs present for the database. With these files, the database can be restored and the database can be brought into a Clean Shutdown state, meaning all transactions in the logs have been committed to the .edb file and the database can be mounted. As the backup completes, it sends a command to the ESE database engine stating that any transaction logs older than a certain point can be truncated (deleted). These logs are no longer required on the Exchange server because we now possess a copy of them in the backup set.

Note: Technically, once a transaction log has been written to disk, it is no longer needed. When the time comes to commit transactions to the database (.edb) file, this action occurs from cache to the .edb file, not from the transaction logs to the .edb file. However, you should not manually delete transaction logs as these are vital for recovery purposes.

Back to our scenario where the .edb file was lost due to a storage failure. We would simply restore the database and transaction logs from our Exchange-Aware backup, and replay both the restored and whatever logs are available into the .edb file. If a complete set of logs are available, the result is that Exchange can replay a continuous stream of transaction logs from the point where the backup was created to the time when the failure occurred and result in no data loss.

This brings me to the question: "Why can't I use Circular Logging to delete my transaction logs automatically so they don't fill up my drive?" First we need to understand that Circular Logging is a property on a database. When circular logging enabled, the Information Store deletes transaction log files as soon as the transactions contained within them have been committed from cache to the .edb file. A much smaller set of logs is maintained as the set never grows over time because logs are continuously deleted.

The command to enable circular logging is as follows:

```
[PS] C:\> Set-MailboxDatabase –Identity <DatabaseFileName> -CircularLoggingEnabled $True
```

The question then comes into play as to why not enable circular logging for every database? Having read the chapter up to this point, the answer is obvious. If your database subsequently becomes unusable due to a failure, no transaction logs will be available for replay into the database, as all but the most recent logs will have been automatically deleted by Exchange. Since all transaction logs must be replayed into the database in their proper sequence, your only choice would be to restore the most recent database backup, and lose all Exchange transactions that occurred between the time of the backup and the time of failure.

It's at this point that I should define the differences between traditional (JET) Circular Logging and Continuous Replication Circular Logging (CRCL). What I described above is JET Circular Logging and is used on standalone Exchange Servers, or databases on Exchange Servers which do not have replicated copies. CRCL is used when a database has replicated copies in a DAG and is enabled using the same command. When using traditional backup methods, neither should be enabled as the backup program's consistency check will fail due to missing logs.

Note: After executing this command, the database must be dismounted and remounted before the change takes effect. Once enabled, any transaction logs that have been committed to the database will be automatically removed. So if there were 50k transaction logs present, they would be automatically deleted.

In my opinion, JET Circular Logging should only ever be enabled in a lab or when you are in dire need of disk space during a recovery or during transaction-intense operations such as moving mailboxes. This is because if you encounter a failure and lose the database, you will lose all transactions that occurred between the last backup and the failure. CRCL should only be enabled in a DAG environment where Exchange Native Data Protection is being used, as traditional backups are not performed. In this configuration, since backups are not performed, no transaction logs will ever be truncated and database drives would ultimately reach capacity.

The logic used to determine when logs will be truncated is as follows (for more information, see this reference):

For truncation to occur on highly available (non-lagged) mailbox database copies, the answer must be "Yes" to the following questions:

- Has the log file been backed up, or is CRCL enabled?
- Is the log file below the checkpoint?
- Do the other non-lagged copies of the database agree with deletion?
- Has the log file been inspected by all lagged copies of the database?

For truncation to occur on lagged database copies, the answer must be "Yes" to the following questions:

- Is the log file below the checkpoint?
- Is the log file older than ReplayLagTime + TruncationLagTime?
- Is the log file deleted on the active copy of the database?

Simply put, the DAG ensures a transaction log has been inspected by all database copies in the DAG before it can be deleted. The log file must also be below the Checkpoint Depth Target, which is 100 transaction logs in DAG-protected databases. Knowing this target is important for understanding expected behavior when verifying transaction log truncation after a successful Exchange-Aware backup. On a standalone database there will always be ~20 transaction logs and on a DAG-protected database there will always be ~100 transaction logs, even after a successful Exchange-Aware backup. This is because only logs that have reached the Checkpoint Depth Target are eligible for truncation. On several occasions I've been asked why transaction logs within a DAG were not truncating after a successful backup because there always seemed to be transaction logs in the log directory. The short answer is there will always ~100 logs in these directories. If you want to verify successful log truncation following a backup in a DAG, I recommend the following article. Similarly, in lab environments truncation may not occur because the database is new and has yet to generate 100 transaction logs, so there's nothing to truncate.

Note: While CRCL is enabled on a DAG-protected database, additional copies cannot be added. CRCL must be disabled, and the database must be dismounted/remounted before additional copies can be added.

Understanding the inner workings of Exchange databases helps to determine why a backup might not be successful and the steps needed to remedy it. We'll now move on to troubleshooting the underlying VSS components which make Exchange-Aware backups possible.

Troubleshooting Volume Shadow Copy Service (VSS) with Exchange Backups

For an excellent guide on Exchange VSS Backups, I recommend this article by Exchange MVP Jaap Wesselius as well as his recorded session at MEC on Exchange Backup, Restore, and Disaster Recovery. He also has an excellent article on Exchange database technologies here. For even more information about Exchange database technology and DAG concepts, read the relevant chapters in Microsoft Exchange 2013 Inside Out: Mailbox and High Availability (Microsoft Press).

Streaming Backups are no longer used with Exchange 2010 and newer (although some of these same components are used when a DAG performs a database seed operation in a DAG). Instead, Exchange relies upon Exchange VSS Backups (another excellent blog series on Exchange backups). In summary, a VSS backup consists of the following components:

- VSS Service: Service within Windows
- Requestor: Backup Application
- Writer: An application's VSS component
- Provider: Hold backup data within Windows or Hardware

You can find some articles to help you understand backups and VSS at the end of this chapter.

It's important to understand the difference between a VSS Snapshot (which is a file system snapshot) and a Hypervisor Snapshot (which is used in virtualization; aka Checkpoint in Hyper-V). Hypervisor

Snapshots/Checkpoints are not supported with production Exchange servers and should not be relied on as a supported Exchange backup solution. When performing a backup against an Exchange virtual machine, the backup solution should have a method of contacting the VSS service inside the VM so a proper VSS backup can occur (and logs can be truncated).

Troubleshooting VSS extends beyond just Exchange troubleshooting, as many applications rely on VSS for backups. However, most VSS issues in my experience revolve around the following:

- VSS Writers
  - Use *vssadmin list writers* to query any writers in a failed state which may require restarting of their respective service
- VSS Providers
  - Use *vssadmin list providers* to view installed providers.
  - Most VSS-based backup applications leverage the built-in VSS provider, but some utilize their own which may present atypical behavior. Work with vendor if issues arise.
- VSS ShadowStorage
  - Use *vssadmin list shadowstorage* to view used/available space for snapshot data.
  - ShadowStorage serves as temporary storage for capturing copy-on-write blocks during a snapshot. Backups may fail if there is insufficient ShadowStorage space for a backup to complete.
  - It's important to keep Windows updated, as past issues have resulted in ShadowStorage data not being properly purged.

Note: In some environments, free disk space can become a problem if ShadowStorage is consuming too much space. While ShadowStorage is used for other functions, Exchange should only need ~1 GB maximum disk space to perform a VSS snapshot. Reference. Also, in some cases there can be inefficient use of ShadowStorage if a VSS provider vendor uses a large differential block size. Check with your backup VSS provider vendor if you experience excessive use of shadowstorage.

Database Corruption/Dirty Shutdown Scenarios

During normal Exchange database operations, it's certainly not unusual for a database process to terminate unexpectedly. A termination might be due to losing access to backend storage, file system corruption, or server-wide power loss. Many Exchange support problems involve corrupt databases which will not mount, and therefore prevent users from accessing their mailbox data. The following issues can prevent an Exchange database from mounting:

- The Microsoft Exchange Information Store Service is unable to start
- Missing Exchange database files
- Database is in "Dirty Shutdown" state
- Not enough free disk space on database or log file volume
- Loss of access to underlying storage
- While database is in Clean Shutdown state, significant logical corruption exists causing the database to dismount when an attempt is made to read the data

In situations where the Information Store service will not start, the most common issues involve communications between Exchange and Active Directory or some interaction involving a file used by Exchange and anti-virus software. I recommend using event viewer to verify Active Directory communications as well as

verifying there are no service dependencies not met (such as the Microsoft Exchange Active Directory Topology Service not being started). With anti-virus software, verify all exclusions have been properly configured. It is common to discover that an anti-virus product either locks the database files or blocks necessary RPC communications. Anti-virus is also a common reason why database files become corrupt as a result of multiple processes attempting to access the files simultaneously. It is also common to see that an anti-virus product quarantines database files or logs, a step that might break the sequence of the log stream, making recovery from backup challenging, and possibly breaking DAG replication.

Dirty Shutdown

The words "Dirty Shutdown" are fairly self-explanatory; a database is down and not in a healthy state. The words can still cause anxiety for Exchange Administrators who recall spending many hours while frantically attempting to get a database to mount and restore functionality. Over ten years of support, I have probably worked on well over a hundred Dirty Shutdown support cases. But to be honest, a decade worth of experience is not required to troubleshoot a Dirty Shutdown event accurately, especially since the proper course of actions can easily fit into a decision tree. Figure 9-1 provides a high-level flow chart for navigating common scenarios where a database will not mount:

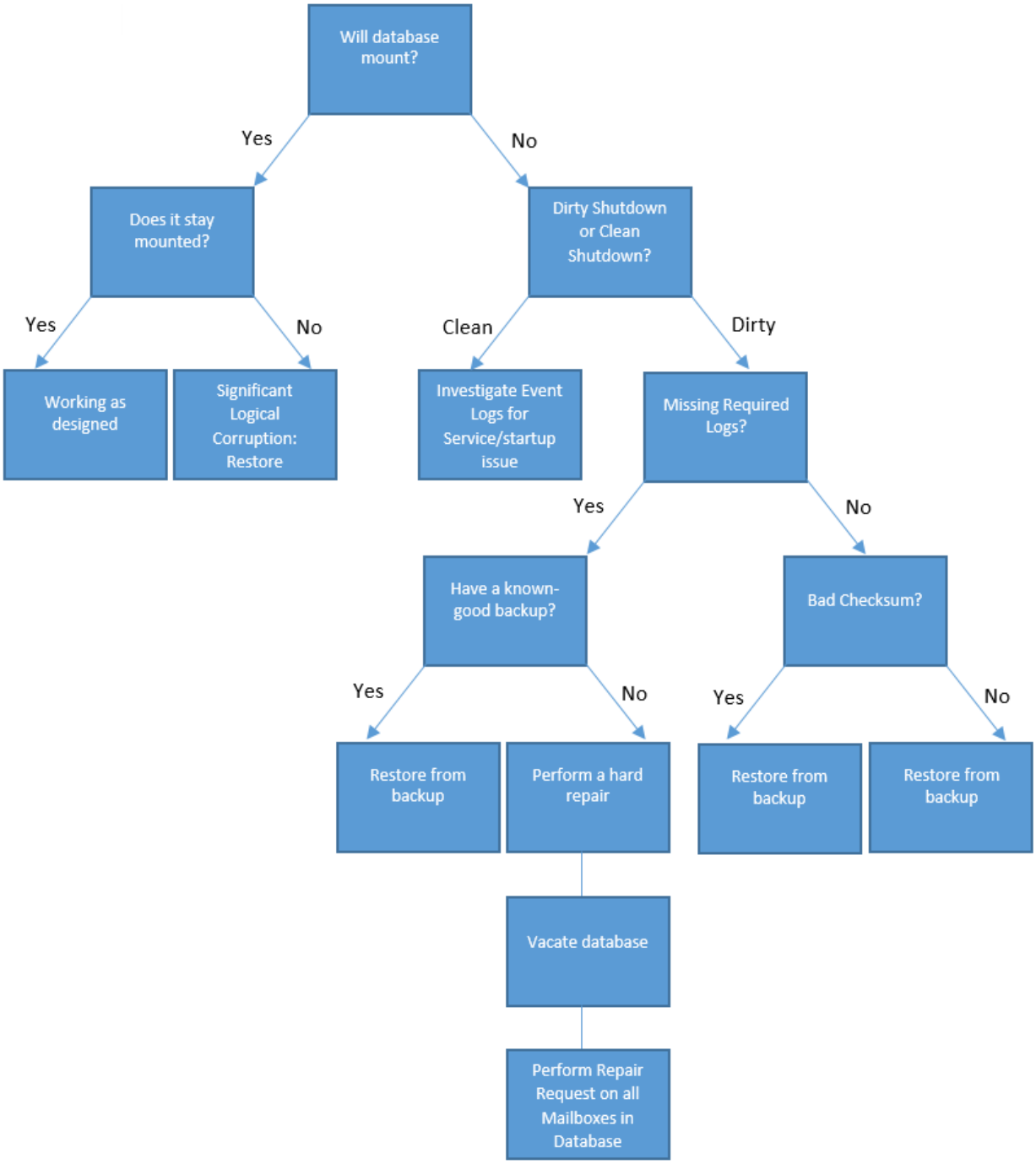


Figure 9-1: Provides a high-level decision tree for common database-down scenarios

First, let's define what happens in a Dirty Shutdown. When Exchange dismounts a database, the Information Store ensures that all transactions (dirty database pages) in cache are committed to the database (.edb) file. When this is allowed to happen, the database is said to be in "Clean Shutdown" and requires no transaction logs to mount because all transactions have already been replayed into the database. Figure 9-2 demonstrates how the ESEUTIL utility validates that a database is in Clean Shutdown state.

```
[PS] C:\db1>eseutil /mh db1.edb

Extensible Storage Engine Utilities for Microsoft(R) Exchange Server
Version 15.00
Copyright (C) Microsoft Corporation. All Rights Reserved.

Initiating FILE DUMP mode...
         Database: db1.edb


DATABASE HEADER:
Checksum Information:
Expected Checksum: 0xff5ae2ab
  Actual Checksum: 0xff5ae2ab

Fields:
        File Type: Database
         Checksum: 0xff5ae2ab
   Format ulMagic: 0x89abcdef
   Engine ulMagic: 0x89abcdef
 Format ulVersion: 0x620,20
 Engine ulVersion: 0x620,20
Created ulVersion: 0x620,20
     DB Signature: Create time:12/03/2015 14:54:57.970 Rand:625297155 Computer:
         cbDbPage: 32768
           dbtime: 158401 (0x26ac1)
            State: Clean Shutdown
    Log Required: 0-0 (0x0-0x0)
    Log Committed: 0-0 (0x0-0x0)
   Log Recovering: 0 (0x0)
```

Figure 9-2: Database DB1 in Clean Shutdown and requiring no logs to mount

However, if something happens to cause the database to terminate abruptly, it is probable to result in a Dirty Shutdown state. Whether a Dirty Shutdown happens and which transaction logs may be required to recover the database is dependent entirely upon which transactions were being processed at the time of the failure. Figure 9-3 displays a database in Dirty Shutdown, requiring log 3 (E0000000003.log) to mount.

```
[PS] C:\db1>eseutil /mh db1.edb

Extensible Storage Engine Utilities for Microsoft(R) Exchange Server
Version 15.00
Copyright (C) Microsoft Corporation. All Rights Reserved.

Initiating FILE DUMP mode...
         Database: db1.edb


DATABASE HEADER:
Checksum Information:
Expected Checksum: 0xf598e2e3
  Actual Checksum: 0xf598e2e3

Fields:
        File Type: Database
         Checksum: 0xf598e2e3
   Format ulMagic: 0x89abcdef
   Engine ulMagic: 0x89abcdef
 Format ulVersion: 0x620,20
 Engine ulVersion: 0x620,20
Created ulVersion: 0x620,20
     DB Signature: Create time:12/03/2015 14:54:57.970 Rand:625297155 Computer:
         cbDbPage: 32768
           dbtime: 158401 (0x26ac1)
            State: Dirty Shutdown
    Log Required: 3-3 (0x3-0x3)
    Log Committed: 0-4 (0x0-0x4)
   Log Recovering: 0 (0x0)
```

Figure 9-3: DB1 in Dirty Shutdown and requiring log E0000000003.log to mount

Note: An easy way to get a database into Dirty Shutdown (for testing purposes, only do this in a lab) is to use Task Manager to kill the Store.exe (Exchange 2000-2010) or the Microsoft.Exchange.Store.Service.exe (Exchange 2013-2016) process. Before doing this, I recommend setting the Microsoft Exchange Information Store Service to "Disabled" (but not stopping it). This will prevent the service from automatically restarting itself and attempting to mount the database while you inspect it.

The DB1 database shown in Figure 9-3 requires transaction log E0000000003.log because the transactions contained within it were either partially committed to the .edb file or are simply required to bring the database to a consistent state. It is hard to say exactly why certain logs are required without a deep programmatic understanding of JET Databases, but nonetheless the log is required and must be available before the database can be mounted.

ESEUTIL Commands and Recovery Procedure

As you've already seen from the preceding discussion, the ESEUTIL /mh command is run against a dismounted database to view database header information. The most useful data in the header is in the "State" and "Log Required" properties. State displays whether a database is in either "Dirty Shutdown" or "Clean Shutdown". As a bit of developer geek trivia, there are four other states which are seldom seen: JustCreated, Being Converted, ForceDetach, and Illegal. Log Required displays the transaction log or logs required to be present and uncorrupted for the database to be able to replay their contents into its tables. The first step of troubleshooting a Dirty Shutdown is to run ESEUTIL /mh against the database (.edb) file and determine which logs are required for that database.

This command can be used to verify the .edb file and log folder paths so you know where all the files are located for a given database:

```
[PS] C:\> Get-MailboxDatabase DB1 | Format-List Name,EdbFilePath,LogFolderPath
```

The next step is to verify the required log files are present in the database's log directory. This can be done visually via Windows Explorer, but the preferred method is to use the below command:

```
C:\> ESEUTIL /ml E00
```

In the above command, E00 is the log sequence for this database. The first database on a server will have a sequence of E00, the second database E01, the third E02, and so on in Hexadecimal. This means the 11th database on a server will have a sequence of E0A. This command will parse through every transaction log in the directory from oldest to newest (E0n is always the current log file so is therefore the newest) and verify that every log in the stream is present and not corrupted. Figure 9-4 displays this action on the same DB1 database which I intentionally placed into a Dirty Shutdown.

```
[PS] C:\db1>eseutil /ml e00

Extensible Storage Engine Utilities for Microsoft(R) Exchange Server
Version 15.00
Copyright (C) Microsoft Corporation. All Rights Reserved.

Initiating FILE DUMP mode...

Verifying log files...
     Base name: e00

      Log file: C:\db1\E0000000001.log - OK
      Log file: C:\db1\E0000000002.log - OK
      Log file: C:\db1\E0000000003.log - OK
      Log file: C:\db1\E00.log - OK

No damaged log files were found.

Operation completed successfully in 0.203 seconds.
```

Figure 9-4: ESEUTIL /ml run against DB1, which is newly created and therefore only has 4 log files

In the case of DB1, all log files are healthy and accounted for. This means that if I attempt to mount this database, following a successful mount the database will automatically recover by replaying the transactions in the available log set. However, for this example I removed log 3 from the directory and repeated the

command (Figure 9-5). Remember from Figure 9-3 that log 3 is required for this database to successfully mount. The -528 error indicates that a required log file is missing.

```
[PS] C:\db1>eseutil /ml e00

Extensible Storage Engine Utilities for Microsoft(R) Exchange Server
Version 15.00
Copyright (C) Microsoft Corporation. All Rights Reserved.

Initiating FILE DUMP mode...

Verifying log files...
     Base name: e00

      Log file: C:\db1\E0000000001.log - OK
      Log file: C:\db1\E0000000002.log - OK
      Log file: C:\db1\E00.log
                ERROR: Missing log file(s). Log file is generation 4 (0x4), but expected generation is 3 (0x3).

Operation terminated with error -528 (JET_errMissingLogFile, Current log file missing) after 0.156 seconds.
```

Figure 9-5: Result of ESEUTIL /ml with a missing file in the log stream

Because we are missing a required log file, the database will not mount, as seen in Figure 9-6.

```
[PS] C:\db1>Mount-Database db1
Failed to mount database "db1". Error: An Active Manager operation failed. Error: The database action failed. Error:
Operation failed with message: MapiExceptionDatabaseError: Unable to mount database. (hr=0x80004005, ec=1108)
Diagnostic context:
    Lid: 65256
    Lid: 10722   StoreEc: 0x454
    Lid: 1494    ---- Remote Context Beg ----
    Lid: 45120   dwParam: 0x161B410C
    Lid: 57728   dwParam: 0x161B41D7
    Lid: 46144   dwParam: 0x161B42C2
    Lid: 34880   dwParam: 0x161B42C2
    Lid: 34760   StoreEc: 0xFFFFFDF0
    Lid: 41344   Guid: 404e7bc0-c482-4ce8-9c21-7bc8270b42c3
    Lid: 35200   dwParam: 0x219E0
    Lid: 46144   dwParam: 0x161B588C
    Lid: 34880   dwParam: 0x161B588C
    Lid: 54472   StoreEc: 0x1388
    Lid: 42184   StoreEc: 0x454
    Lid: 1750    ---- Remote Context End ----
    Lid: 1047    StoreEc: 0x454      [Database: DB1, Server: ASH-EX1.ASH.NET]
    + CategoryInfo          : InvalidOperation: (DB1:ADObjectId) [Mount-Database], InvalidOperationException
    + FullyQualifiedErrorId : [Server=ASH-EX1,RequestId=07108678-9708-43a9-960d-db4d5098c377,TimeStamp=12/6/2015 7:15:
   40 AM] [FailureCategory=Cmdlet-InvalidOperationException] 761B127D,Microsoft.Exchange.Management.SystemConfigurati
  onTasks.MountDatabase
    + PSComputerName        : ash-ex1.ash.net
```

Figure 9-6: Result of attempting to mount a database with a missing Required Log

Now if I place log 3 back into the directory but instead remove log 2, what do you think will happen? Remember, only log 3 was required because a transaction contained within it was deemed necessary to mount the database. So while running *ESEUTIL /ml* will still list log 2 as missing, the database would mount because the Checkpoint File (.chk) tells the database to only be concerned with transaction log file 3 or newer.

What would happen if I removed the checkpoint file while log 2 was still missing (remember, the database only requires log 3)? In this case, the database will fail to mount because without a checkpoint file, all of the logs present in the directory must be inspected in ascending order and any missing logs will result in a failure to mount. Now what's really interesting is that if I also remove log file 1, the database will mount successfully. This is because when the log stream is reviewed, there are no missing log files. The Information Store identifies log file 3 as the oldest log, and begins processing onward starting at that log. Whereas before, it detected logs 1 and 3 but knew 2 was missing, so the process failed. Remember, we don't require every transaction log since the beginning of the database to be present in the directory (this would be unreasonable, as old log files get deleted once a successful backup has occurred), but the logs which are there must be contiguous from oldest to newest.

Understanding what log files might be required is important. Although this example database has only a few log files, production databases can have tens of thousands, depending on the last successful backup. Therefore, since many backup programs perform a log stream integrity check (similar to an *ESEUTIL /ml*) for the log directory, missing or damaged log files within the directory can cause this check to fail.

Let's now discuss how a soft recovery using ESEUTIL /r is performed. With Exchange 2010 and later versions, it is rarely necessary to manually run a soft recovery as a means to get a database to mount after a failure. This is because Exchange does this process automatically (and does a pretty good job), and if the necessary log files are present in the directory, you simply need to mount the database for recovery to happen. More commonly

however, it's used when restoring a database from backup or recovering using a Lagged Copy. For our DB1 example, I will replace all transaction log files and run:

```
C:> ESEUTIL /r e00
```

Figure 9-7 displays a soft recovery being performed against DB1 which places it into a Clean Shutdown and ready to mount state.

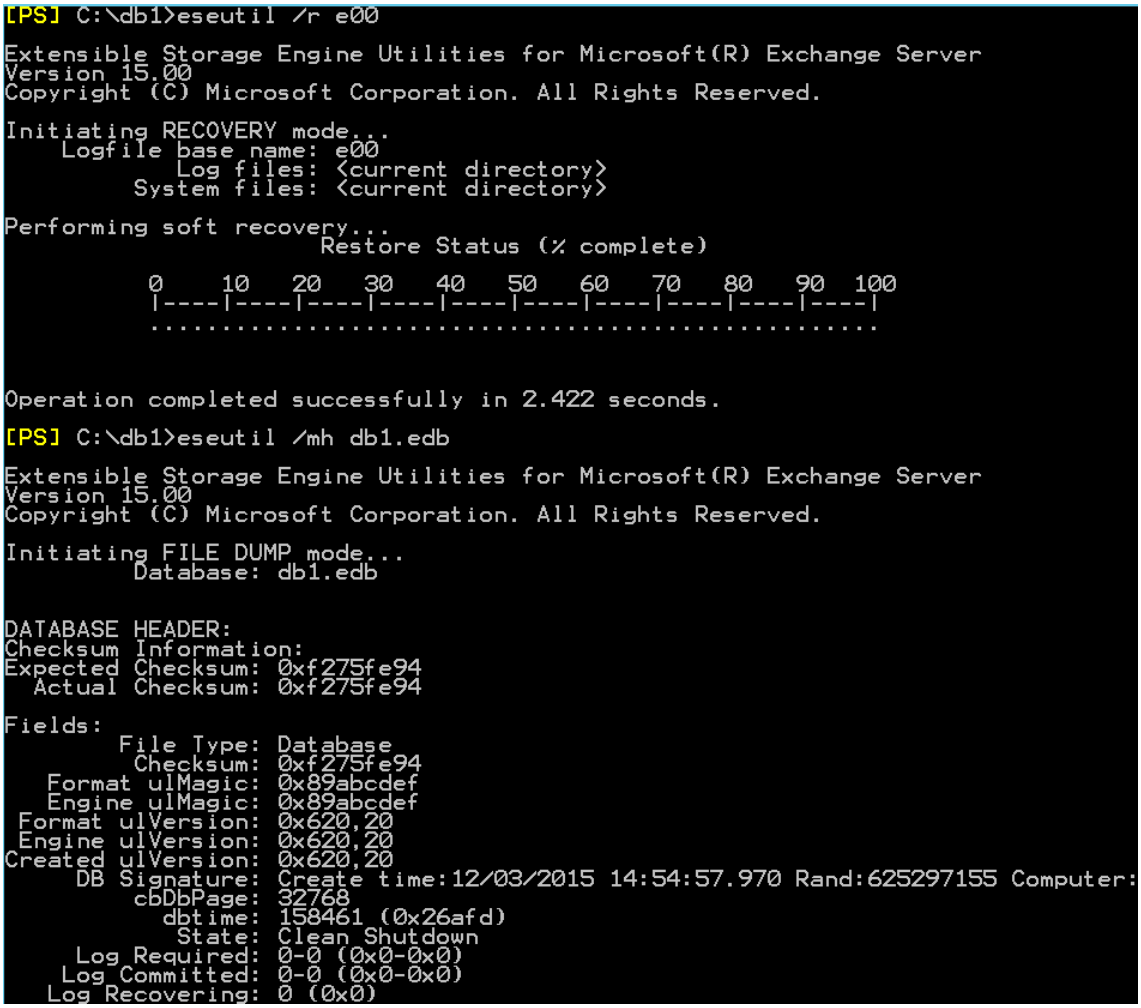


Figure 9-7: Performing a soft recovery against DB1 to place it into a Clean Shutdown state

Another useful ESEUTIL command performs a checksum validation against the .edb file, using the following command syntax:

```
C:> ESEUTIL /k <DatabaseFileName>
```

Think of this as a scan for physical corruption in the database. If this command returns some bad checksums, your likelihood of a soft recovery is unlikely and a restore from backup could be required. Figure 9-8 displays ESEUTIL /k being performed against DB1. As you can see, no bad checksums were encountered.

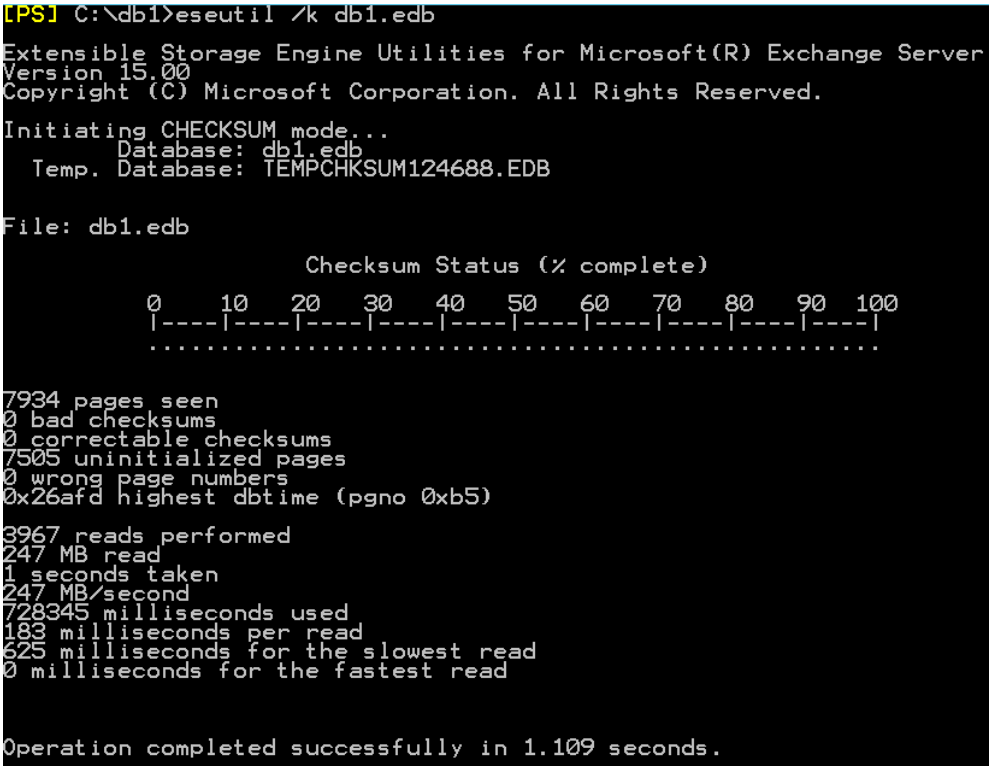


Figure 9-8: ESEUTIL /k being performed against DB1

The last command we need to cover is the dreaded *ESEUTIL /p* command (also known as a Hard Recovery/Repair). Unfortunately, the internet is full of bad advice regarding *ESEUTIL /p*. The usual scenario is that someone starts a thread pleading for help with a database that will not mount, and the first response is usually, "just run an *ESEUTIL /p <DatabaseFileName>* and the database should mount". This advice is rarely accompanied with a warning about data loss or the fact that a /p should **NEVER** be your first troubleshooting step when troubleshooting databases which will not mount. In fact, a /p must always be your path of last resort, as it invariably means that data loss will result.

So what is a /p and why is it so bad? Simply put, a /p does anything it can to remove problems that might cause a database to fail to mount, even if that means deleting data and rolling back transactions in an effort to get into a Clean Shutdown state. In the event of bad checksums or other types of corruption, you could say a /p removes anything it doesn't recognize as valid, all in an effort to purge corruption. This can potentially have catastrophic effects to the data in the database as no one can predict what pages will be removed or what data is held in those pages as it all depends on what was happening with the database at the time of failure or which part of the database structure was affected. Sometimes it is a single data page, but sometimes the removal of a single page can remove an entire table from the database.

Based on my experience, ESEUTIL /p will be successful in 9/10 instances and the database will mount with minimal data loss. However, I have also seen how ESEUTIL purged so many pages that a 200 GB database was reduced to 60 GB with no indication of what was lost in the 140 GB that was removed. Losing data is the biggest danger of running ESEUTIL /p, mounting the database, and then carrying on business as usual. You'll probably be happy that the database comes back online and not realize that data loss has occurred until users point out large holes in their mailboxes later, at which point it's likely too late to recover it easily. So if an Exchange-Aware backup is available, it should always be used instead of running the quicker/easier *ESEUTIL /p*.

However, even if hard recovery is successful and involves minimal data loss, there are still repercussions that flow from performing a hard repair on a database. Microsoft's New Support Policy for Repaired Exchange

Databases explicitly states a database is in an unsupported state once a hard repair has been performed on it. This is because Microsoft Support has collected data indicating these databases are unstable and often exhibit a number of unpredictable issues, even with client connectivity. The effect of the support policy means that if you ever have to perform a hard repair on a database, you should immediately move all mailboxes from the database to other databases as soon as the database can be mounted. Once all the mailboxes are moved, you can remove the database. Moving the mailboxes will remove any lurking corruption and make sure that they will function correctly in the future. Also, I recommend a New-MailboxRepairRequest be performed against all migrated mailboxes to remove any logical corruption.

Assuming all other recovery methods have failed and no backup exists, you need to know how to run a hard repair. The syntax for this command is:

```
C:> ESEUTIL /p <DatabaseFileName>
```

Before this command is allowed to be run, you'll receive the warning shown in Figure 9-9.

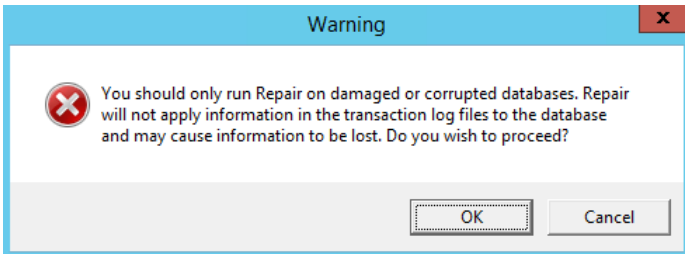


Figure 9-9: Warning regarding data loss when running an *ESEUTIL /p*

A hard repair can take anywhere from a few seconds to over 24 hours to run, depending upon database size, available computer resources, and the amount of corruption detected. This is another reason to keep your databases under 200 GB for standalone servers, as recovery operations (including restoring a backup) take considerably longer for larger databases. It is reasonable to expect that ESEUTIL will process data at the rate of 10-15 GB an hour (storage performance is also a factor), but without knowing the nature of the corruption, it is difficult to predict how long the complete repair will take. Figure 9-10 displays the execution and completion of an ESEUTIL /p operation.

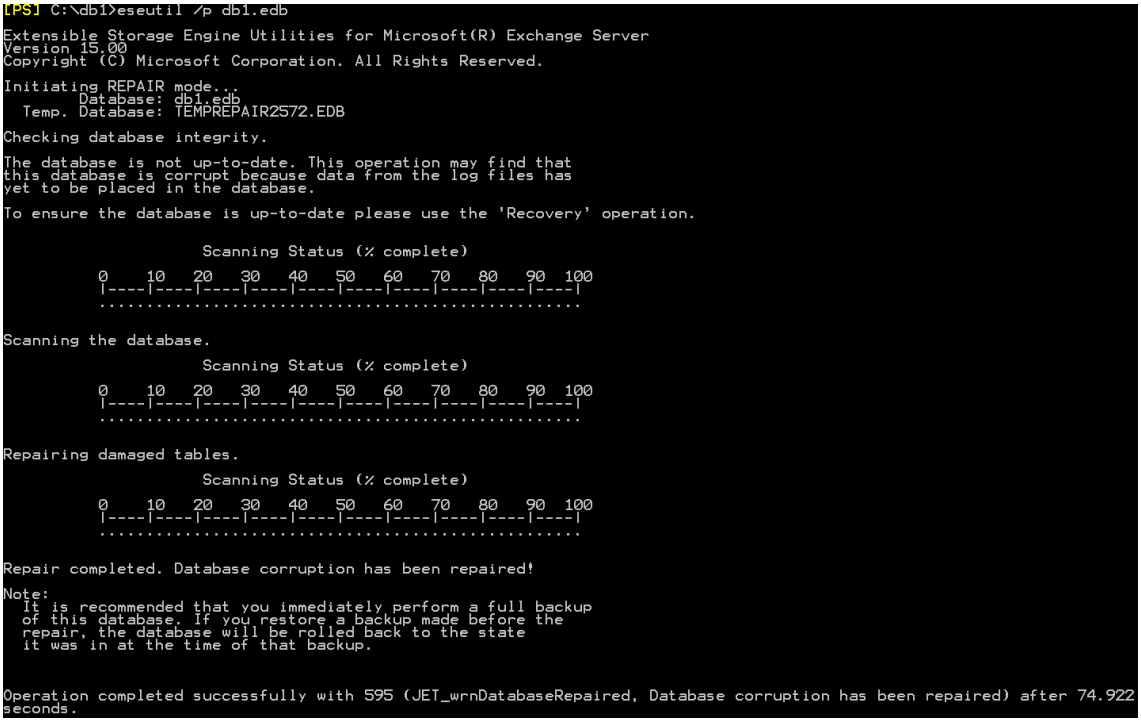


Figure 9-10: The execution and completion of *ESEUTIL /p* against DB1

Once the hard repair is complete, all database files should be removed from the folder except for the .edb file. In fact, you can purge any previous transaction logs for this database, as they'll now be useless as they cannot be replayed into the database because the newly repaired database has a different signature to the corrupted file. Also, it's possible you'll need to enable this database to be overwritten using this command:

```
[PS] C:\> Set-MailboxDatabase –Identity DB1 –AllowFileRestore $True
```

After the database has been mounted, as previously mentioned, all mailboxes should be moved from it, and a New-MailboxRepairRequest should be performed against them.

Database Defragmentation

There was a time when performing an Offline Defragmentation on an Exchange database was considered a maintenance operation that should be scheduled regularly. However, beginning with Exchange 2007, there is little reason to perform an Offline Defragmentation. Not only does database replication complicate the process, but also because there is little to be gained. Historically, when mailbox data was purged, it left whitespace in the database. In other words, after deleting 40 GB of data from a 100 GB database file (.edb) would still remain a 100 GB file. Eventually Exchange would reclaim this space as data is added during normal user operations, but if available disk space was running low, administrators might perform an offline defrag against the database to reduce it down to ~60 GB. This was accomplished using the following command on a dismounted database:

```
C:> ESEUTIL /d <DatabaseFileName>
```

An offline defrag was also required after an *ESEUTIL /p* was performed against a database (Microsoft now requires the database instead be vacated). The downside of the offline defrag was it required the database to be offline and could take many hours (5-10 GB/hr depending on the hardware of the time), but it was thought a better alternative than moving all mailboxes out of the database, as that too was an offline operation (for

Exchange 2000-2007). Also, because a new database is created during the defrag operation, enough free disk space equal to 110% of the current .edb file is required to perform an offline defrag.

Starting with Exchange 2010, as all mailbox moves use an online process, offline defragmentation should be avoided. Database whitespace should either be allowed to be consumed by new database pages over time, or mailboxes should be moved to a new database (allowing you to delete the old database file). Since the move is now an online process, it's a much better alternative than an Offline Defrag.

Command Examples

General ESEUTIL Log File Replay Rules (Reference):

- You cannot replay log files from one database against a different one. The operations held inside a log file are low-level that collectively form a stream of transactions to be applied against a database. You will not see anything inside a log file such as "Deliver Message A to Mailbox B." A better example of a log file operation is "Write this stream of 123 bytes to byte offset 456 on database page 7890." Imagine that you gave someone instructions for editing a document, and your instructions are "On page five, paragraph four, in the third sentence, insert the phrase 'to be or not to be' after the second word." If these instructions were applied to a document other than the one intended, the result would be random corruption of the document. Likewise, if the wrong log files were played against an Exchange database, a similar result would occur. Exchange therefore has multiple safeguards to prevent such corruption.
- If you defragment or hard repair (ESEUTIL /p) an Exchange database, transaction logs that previously were associated with this database can no longer be replayed into it. If you try to replay log files after a defragmentation or hard repair, Exchange skips the inappropriate transaction logs. Again, consider the analogy of editing the document. If a paragraph has been moved, edited, or deleted since the instructions were created, applying the out-of-date instructions would be as destructive as applying them to an entirely different document.
- You cannot replay log files unless all uncommitted log files from the time the database was last running are available. You must have all log files starting from the checkpoint at the time the database was backed up. You can then replay log files from this point as long as they follow an unbroken sequence. If there is a single log file missing in the middle or from the beginning of the sequence, replay stops there.
- You cannot replay log files if the checkpoint file points to the wrong log. Exchange treats a checkpoint log as if it were the first log available and ignores all older log files. If you restore an older file copy of the database, the checkpoint will be too far ahead, and Exchange tries to start log file replay from a log file that is too new. You can solve this problem by removing the checkpoint file; thus forcing Exchange to scan all available logs. (If you restore an online backup, recovery ignores the checkpoint file.)

Check the header of a database:

```
C:> ESEUTIL /mh <DatabaseFileName>
C:> ESEUTIL /mh DB1.edb
```

Verify log stream of transaction logs in a folder:

```
C:> ESEUTIL /ml <E0n>
C:> ESEUTIL /ml E00
```

Verify the Checksum of a database file:

```
C:> ESEUTIL /k <DatabaseFileName>
C:> ESEUTIL /k DB1.edb
```

Perform a soft recovery against a database:

```
ESEUTIL /r <E0n>
ESEUTIL /r E00
```

Note: /a can also be used if the last log in the stream is missing. The /a is not always successful.
Perform a hard recovery against a database:

```
C:> ESEUTIL /p <DatabaseFileName>
C:> ESEUTIL /p DB1.edb
```

Perform an offline defragmentation against a database:

```
C:> ESEUTIL /d <DatabaseFileName>
C:> ESEUTIL /d DB1.edb
```

Perform a Mailbox Repair Request against a mailbox while checking against all corruption types:

```
[PS} C:\> New-MailboxRepairRequest -Mailbox “John Smith” -CorruptionType
SearchFolder,FolderView,AggregateCounts,ProvisionedFolder,ReplState,MessagePTAGCn,MessageID,Rule
MessageClass,RestrictionFolder,FolderACL,UniqueMidIndex,CorruptJunkRule,MissingSpecialFolders,Dr
opAllLazyIndexes,ImapID,ScheduledCheck,Extension1,Extension2,Extension3,Extension4,Extension5
```

Monitoring a Mailbox Repair Request:

```
[PS] C:\> Get-MailboxRepairRequest –Mailbox “John Smith” | Format-List
```

Note: See the Migration chapter and the Mailbox Corruption module for additional details on Repair Requests and their function.

Recovery Databases

A Recovery Database is not a new concept to Exchange. It is merely the evolution of Recovery Storage Groups which were used in Exchange 2003-2007. After Microsoft removed Storage Groups from the product in Exchange 2010, the naming was changed from Recovery Storage Group to Recovery Database. A Recovery Database (RDB) is a special kind of mailbox database that allows you to mount a restored mailbox database taken from a backup and extract data from the restored database as part of a recovery operation. Database files can be restored from backup, placed into a Clean Shutdown state, and mounted in an RDB so its contents can be extracted and placed into a mailbox of the administrator’s choosing. The following points differentiates an RDB from a regular mailbox database (see reference):

- An RDB can only be created using the Exchange Management Shell.

- Mail can't be sent to or from an RDB. All client protocol access to an RDB (including SMTP, POP3, and IMAP4) is blocked. This design prevents using an RDB to insert mail into or remove mail from the messaging system.
- Client MAPI access using Microsoft Office Outlook or Outlook Web App is blocked. MAPI access is supported for an RDB, but only by recovery tools and applications. Both the mailbox GUID and the database GUID must be specified when using MAPI to log into a mailbox in an RDB.
- Mailboxes in an RDB can't be connected to user accounts. To allow a user to access the data in a mailbox in an RDB, the mailbox must be merged into an existing mailbox, or exported to a folder.
- System and mailbox management policies aren't applied. This design prevents items in an RDB from being deleted by the system during the recovery process.
- Online maintenance isn't performed for RDBs.
- Circular logging can't be enabled for RDBs.
- Only one RDB can be mounted at any time on a Mailbox server. The use of an RDB doesn't count against the database limit per Mailbox server.
- You can't create mailbox database copies of an RDB.
- An RDB can be used as a target for restore operations, but not backup operations.
- A recovered database mounted as an RDB isn't tied to the original mailbox in any way.

Let's walk through an example of how a Recovery Database might be used:

- A database named DB15 contains 100 mailboxes, of which one is named "John Smith"
- John Smith accidentally deletes a very important item from his mailbox
- The deletion goes unnoticed for a month, allowing the default Deleted Item Retention window of 14 days on the mailbox database to expire and the email to be purged

In this example, an Exchange-Aware backup of DB15 from before the deletion can be restored into a directory. Using ESEUTIL commands, an administrator can verify the database is in a clean shutdown. If not, they can perform an *ESEUTIL /r* to soft recover the restored log files into the database, bringing it into a Clean Shutdown state.

Once the database is in a Clean Shutdown, it can be mounted as a Recovery Database. No users will be able to access this database, and even though the mailboxes within it also exist on the production database (DB15), it is allowed to mount in this special state. Once the database is mounted, the mailboxes it contains, along with their item count, can be viewed using the below command:

```
[PS] C:\> Get-MailboxStatistics –Database <RecoveryDBName> | Format-Table –Auto
```

This mailbox data can now be exported into a production mailbox using the New-MailboxRestoreRequest Cmdlet. This restore can be performed to the same folder the item originally existed in, or sent to a different folder

```
[PS] C:\> New-MailboxRestoreRequest -SourceDatabase <RecoveryDBName> -SourceStoreMailbox
"John Smith" -TargetMailbox "John Smith" -TargetRootFolder "Restored Items"
```

Once the RDB is no longer needed, it can be dismounted, removed from Active Directory, and the restored database files removed.

The preceding example is a fairly common one, though many third-party backup solutions make this process easier by providing streamlined item-level recovery. For this reason, the RDB has become a bit of a lost art to Exchange Administrators. Of course, there are several other use cases where Recovery Databases are used:

- Specific item recovery (as described above) - You can restore from backup data that has been deleted or purged from a mailbox.
- Mailbox recovery - You can recover an individual mailbox from backup when the deleted mailbox retention period has elapsed. You then extract data from the restored mailbox and copy it to a target folder or merge it with another mailbox.
- Same server dial tone recovery - You can perform a recovery from an RDB after the original database has been restored from backup, as part of a dial tone recovery operation.
- Alternate server dial tone recovery - You can use an alternate server to host the dial tone database, and then later recover data from an RDB after the original database has been restored from backup.

Dial Tone Recovery

The capability to implement a Dial Tone Database is also not a new concept in Exchange. The general function of a traditional Dial Tone Database is as follows:

- Database files have been lost or corrupted
- A restore from backup will result in an extended outage as users are unable to access the contents of their mailbox and cannot send/receive emails while the restore processes
- A desire exists to provide users the ability to send/receive new emails while their mailbox data is being recovered
- All remaining database files are removed from the database folders and the command is given to mount the database
- Exchange will warn that the database will be mounted with empty mailboxes, as no database files exist
- All users will be able to access their (now empty) mailboxes and send/receive new mails while access to any preexisting mail, contact, and calendar data is temporarily lost
- The original database files are restored from a backup to a Recovery Database and brought to a Clean Shutdown
- Two Options
    - The mailbox data from the RDB can be merged into the Dial Tone Database to make it the new production database.
    - Both the Dial Tone and RDB can be dismounted and their database files switched. The RDB (now being the smaller of the two) can now be merged into the production database. This option is preferred as it takes less time to merge the smaller database into the larger.
- The Recovery Database is removed and a full Exchange-Aware backup is taken ASAP

This was a common scenario played out in when slow backup solutions and slower hardware meant that it took hours to recover even small databases, leading to in extended periods of unavailability as recovery took place. It is still for this reason that Microsoft recommends that non-DAG protected databases be no greater than 200 GB. As the size of files needing to be restored grows, so does the potential for downtime.

Exchange 2010 introduced Database Portability, a feature permitting database files to be mounted on any server in the Exchange organization. Also, a user's mailbox location can be configured simply by running this command:

```
[PS] C:\> Set-Mailbox –Identity “John Smith” –Database <NewDatabaseName>
```

Of course no data is migrated with this command, instead “John Smith” will find a new empty mailbox when he opens Outlook. But this capability is extremely useful when combined with the ability to move database files and mount them anywhere in the Exchange organization. It's this capability that makes Dial Tone

Portability possible, a much simpler way to use a dial tone database in an Exchange Disaster Recovery scenario. The general outline of a Dial Tone Portability scenario is as follows (Reference):

- Database files have been lost or corrupted (for our example, FailedDB)
- A restore from backup will result in an extended outage as users are unable to access the contents of their mailbox and cannot send/receive emails
- A desire exists to provide users the ability to send/receive new emails while their mailbox data is being recovered
- A new database is created (in this example "DialToneDB")
- Mailboxes on the failed database are rehomed to DialToneDB using *Get-Mailbox -Database <FailedDB> | Set-Mailbox -Database <DialToneDB>*
- DialToneDB is mounted so users can send/receive emails during the restore operation
- A Recovery Database is created and the FailedDB files are restored from backup to its location
- After the FailedDB data is copied to the Recovery Database from backup, but before mounting the restored database, copy any log files from the current FailedDB to the Recovery Database log folder so they can be played against the restored database. This will bring the database as close to current as possible.
- Mount the Recovery Database so the logs can be replayed and then dismount it
- Dismount DialToneDB (involves outage to users) and swap the database files for DialToneDB and the Recovery Database. This results in the restored copy of FailedDB being mounted in production, but missing the past few days of email (which exist in DialToneDB). DialToneDB' s data is now mounted in the Recovery Database.
- Use the *New-MailboxRestoreRequest* Cmdlet to move/merge the contents of the Recovery Database into FailedDB
- At this point recovery is complete and the Recovery Database can be removed

Lagged Copy Recovery

Lastly, a Recovery Database is also used when using a Lagged Mailbox Database Copy to perform a point-in-time recovery. For a detailed outline on his this is performed, see this excellent article from co-author Paul Cunningham, Exchange Server 2013 Lagged Database Copies In Action.

Lagged copies in a DAG hold the replaying of their transaction logs for up to 14 days, allowing the copy to represent a past point in time for that database. In Exchange Native Data Protection, this feature enables the backup-less solution to restore data from a past date chosen by the administrator. The following provides an overview of this process:

- The lagged database copy is suspended
- A copy is made of the lagged database files to preserve the lagged copy itself
- The lagged copy is resumed
- A Recovery Database is created
- The copy of the lagged database files are placed into the Recovery Database folder structure
- All transaction logs just before the point where accidental deletion occurred and up to the current logs can be removed
- ESEUTIL is used to place the database files into a Clean Shutdown
- The files in the Recovery Database can be mounted
- New-MailboxRestoreRequest can be used to extract the required mailbox data

Single Item Recovery makes this process seldom required, as you can easily recover individual items without going through the trouble of mounting a Recovery Database.

Another scenario, and really the primary purpose of lagged copies, is if widespread logical corruption left a full database restoration as the only option for recovery. This scenario requires the lagged copy be activated into production, but not before all transaction logs involved in the corruption are removed. The following provides an overview of this process:

- The lagged database copy is suspended
- A copy is made of the lagged database files to preserve the lagged copy itself
- All transaction logs just before the point where the logical corruption occurred and up to the current logs are removed
- ESEUTIL is used to place the database files into a Clean Shutdown
- The lagged database copy is activated
- At this point, Safety Net will activate and re-inject all mail stored in the transport database for the SafetyNewHoldTime.

Note: When using Lagged Database Copies, the SafetyNetHoldTime value should be equal to the ReplayLagTime of the lagged copies. This ensures that should the Lagged Copies be activated, the messages contained within Safety Net will include the entire time window of the Replay Lag Time. Also, outside of Lagged Copy Recovery scenarios, the cmdlet Add-ResubmitRequest can be used to request data contained within SafetyNet to be replayed into mailboxes. Remember, the Information Store performs duplicate checking, so users should not experience duplicated messages.

Disaster Recovery Installation

It is natural that most Exchange recovery operations center on databases. However, what should the recovery method be when the entire Exchange installation on Windows becomes unrecoverable? Or when issues exist which require the Operating System to be reinstalled? For most other applications the solution would be a full image-based backup of the system, reverting it to a point in time which would include the OS, the registry, and any installed applications with their data. Yet Exchange would have us abandon these sometimes complex and pointless full recovery methods. Why pointless? Often, the reason for reinstallation is a problem within the OS itself, be it corruption, misconfiguration, or instability. It's highly likely these problems will still exist in the restored system.

For these reasons Exchange provides the Disaster Recovery Installation option for Exchange. This installation option following the below steps (Reference):

- An Exchange server installation is lost or beyond repair
- The system is reinstalled with the same Operating System and brought back to the same patch level as the previous server (DO NOT attempt to gracefully uninstall Exchange or attempt to disjoin the server from the domain; I recommend a format of the OS partition)
- The Computer Account for the Exchange Server should be reset
- Configure the Computer Name on the new installation the same as the previous server
- Rejoin the computer to the domain
- Install Exchange prerequisites including the same CU version of Exchange that was previously installed (this process will not work if a different installation version is used than was previously installed)
- Open an elevated Command Prompt on the server to be recovered and run the below command:
    - *Setup /m:RecoverServer /IAcceptExchangeServerLicenseTerms*
    - **Note:** If Exchange is installed in a location other than the default location, you must use the /TargetDir switch to specify the location of the Exchange binary files. If you don't use the /TargetDir switch, the Exchange files are installed in the default location (%programfiles%\Microsoft\Exchange Server\V15).

- This causes the Exchange installation to look to Active Directory and pull the Exchange Server configuration that was previously present onto this server
- Once completed and the server is restarted, any databases will be dismounted
- At this time, restore databases from backup
- Import and assign any previously installed certificates

Hopefully this chapter provides and insight into the various tools you have at your disposal, as well being a useful reference when encountering backup, restore, corruption, and disaster recovery scenarios out in the wild. This chapter discussed database corruption and how to overcome it. For information on how to recover from mailbox corruption, see the Mailbox Corruption module in the Migration chapter later in this book. Though first, we'll discuss troubleshooting Exchange Hybrid environments. This is a topic with many moving parts and complex technologies coming together to provide a hybrid On-Prem and Office 365 experience.

Additional reading

Exchange Native Data Protection

- Exchange Native Data Protection
- Exchange Preferred Architecture
- Database Availability Group
- Single Item Recovery
- Lagged Database Copies
- Recovery Database
- In-Place Hold
- Database availability groups (DAGs)
- Recover deleted messages in a user's mailbox
- Activate a lagged mailbox database copy
- Restore an Exchange Server 2013 Database to a Recovery Database
- In-Place Hold and Litigation Hold
- Single Item Recovery Walkthrough
- DeletedItemRetention Parameter
- Email Immutability
- Database Copy Types
- ReplayLagTime Parameter
- Recovery Storage Group
- Exchange-Aware Backup
- Working with Recovery Storage Groups
- Litigation Hold

Traditional Backup Methods

- Exchange Database Files
- Supported Backup Technologies
- Exchange ESE
- Types of Backup (Copy VS Full VS Incremental VS Differential etc.)
- Exchange VSS Writer
- Windows Server Backup
- Exchange Backup Event Logs

- Exchange and Windows Server Backup
- Exchange 2007 / 2010: Windows Server Backup and Performance Issues
- Backups fail due to consistency check failure…
- Verifying log file truncation…
- Exchange 2010: Log Truncation and Checkpoint At Log Creation in a Database Availability Group
- Exchange and VSS -- My Exchange writer is in a failed retryable state…
- Transaction logs and checkpoint files for backup and restore in Exchange 2013
- Exchange 2013: Store, FAST, and ESE Cache Demystified…Hopefully!
- Exchange Database Cache
- Hexadecimal
- Random VS Sequential IO
- Seek Time
- Checkpoint File

Transaction Logging and Recovery

- Best Practice for Supported Storage Configurations
- ESEUTIL
- Getting an Exchange Database into a Clean Shutdown State using Eseutil
- Circular Logging
- CircularLoggingEnabled Parameter
- Dismount-Database Cmdlet
- Mount-Database Cmdlet
- Differences between traditional (JET) Circular Logging and Continuous Replication Circular Logging (CRCL)
- Exchange Circular Logging and VSS Backups
- Circular Logging and Mailbox Database Copies
- Verifying transaction log truncation
- Exchange 2010: Log Truncation and Checkpoint At Log Creation in a Database Availability Group

Troubleshooting Volume Shadow Copy Service (VSS) with Exchange Backups

- VSS BACKUPS IN EXCHANGE
- Session at MEC on Exchange Backup, Restore, and Disaster Recovery
- Exchange Database Technologies
- Microsoft Exchange 2013 Inside Out: Mailbox and High Availability
- Streaming Backups
- Changes to Backup and Restore in Exchange 2010
- Everything You Need to Know About Exchange Backups
- The Basics of the Volume Shadow Copy Service (VSS)
- Troubleshooting the Volume Shadow Copy Service
- Troubleshooting VSS and Backup
- Vssadmin TechNet Article
- Everything You Need to Know About Exchange Backups
- LUN design and Hardware VSS
- The Problem with Hardware VSS Providers and Cluster Technologies like CSV and DAG
- How to use Eseutil.exe to perform actions while databases are online
- Exchange writer in Exchange 2013

- Snapshot (computer storage)
- Checkpoints
- Exchange Server 2013 Virtualization Best Practices
- Why you should keep Windows updated to maintain healthy VSS service
- ShadowStorage Size
- Exchange VSS and Differential Block Size

Database Corruption/Dirty Shutdown Scenarios

- Quick method to diagnose Exchange Active Directory Access & Service Startup Issues
- Anti-Virus Exclusions for Exchange
- Recovering using a Lagged Copy

ESEUTIL Commands and Recovery Procedure

- New Support Policy for Repaired Exchange Databases
- ActiveSync Synching Folders but not Mail (caused by running ESEUTIL /p)
- New-MailboxRepairRequest

Database Defragmentation

- Offline Defragmentation
- Little reason to perform an Offline Defragmentation
- Offline Defrag And DAG Databases, Oh My!
- How To Check Database White Space In Exchange
- How to Defrag an Exchange 2010 Mailbox Database
- Microsoft now requires Hard Repaired (/p) databases be vacated
- Moving mailboxes, the Exchange 2010 way (Online Moves)

Additional References and Command Examples

- Eseutil
- Exchange: Recovering a Database from Dirty Shutdown (the Easy Way)
- Why Exchange Databases Might Remain Dirty After ESEUTIL /R Recovery
- Difference between Exchange Databases Physical and Logical Corruption
- Validate backup integrity by using the Eseutil tool in Exchange 2013
- Transaction Log File Replay: Soft Recovery and Hard Recovery in Exchange Server 2003
- Mailbox Repair Request
- Monitoring a Mailbox Repair Request

Recovery Databases

- Recovery Database
- Recovery Storage Groups
- Storage Groups
- New-MailboxRestoreRequest
- Restoring a Mailbox from an Exchange Server 2013 Recovery Database
- Restoring Exchange Server 2016 Mailboxes and Items Using a Recovery Database

Dial Tone Recovery

- Dial Tone Database

- Database Portability
- Database Parameter of Set-Mailbox Cmdlet
- Move a Mailbox Database Using Database Portability
- Dial tone portability
- Perform a dial tone recovery

Lagged Copy Recovery

- Lagged Mailbox Database Copy
- Exchange Server 2013 Lagged Database Copies In Action
- Exchange Native Data Protection
- New-MailboxRestoreRequest
- Single Item Recovery
- Activate a lagged mailbox database copy
- SafetyNewHoldTime
- ReplayLagTime
- Add-ResubmitRequest

Disaster Recovery Installation

- Disaster Recovery Installation
- Reset a Computer Account
- Exchange prerequisites
- Recovering a Failed Exchange Server 2016 Server
- Introduction to Exchange Server 2016 Backup and Recovery

Chapter 10: Troubleshooting Hybrid Exchange

Andrew Higginbotham

Individual products are usually fairly easy to troubleshoot and support. You can attend product training, install it in a lab, and eventually end up solving enough support tickets to become a subject matter expert. On the other hand, solutions are complex configurations that bring together several different products to solve a defined technical and business need. An Exchange Hybrid Configuration is a complex solution that relies on several different Exchange features and utilizes multiple Microsoft products and services to connect Exchange on-premises organizations to Office 365 tenants. The work necessary to configure a hybrid deployment has evolved since Exchange 2010 SP1 and now Microsoft provides a Hybrid Configuration Wizard to automate the many tedious manual configuration steps which were often poorly implemented. A Hybrid Configuration can involve the following Exchange Server components:

- Cross-Forest Mailbox Moves
- Free/Busy Sharing
- MailTips
- Online Archiving
- Secure Mail via shared SMTP namespace
- Shared Address Books (Unified GAL)
- Remote Domains
- eDiscovery Search and Compliance
- Integrated Administration (RBAC, Exchange Management Shell, EAC)

A Hybrid Configuration might also involve the following Microsoft products, features, and services:

- Office 365
- Azure Active Directory
- Remote PowerShell
- Azure AD Connect (aka. Azure AD Sync/DirSync)
- Active Directory Federation Services (ADFS)
- On-Premises Active Directory Domain Services
- Microsoft Office Suite
- Information Rights Management
- Exchange Online Protection

Needless to say, becoming an effective troubleshooter in all of these areas is an ambitious task. Even for Microsoft, it is often difficult to troubleshoot and resolve issues that occur in hybrid configurations.

The breadth of technologies included in Office 365 can make the manager of any IT support team wonder how it is possible to provide satisfactory support for everything, especially when some of the components operate in the cloud and some on-premises. I call this out because when troubleshooting a product which involves both on-premises and cloud-hosted components, you must understand and be able to identify when an issue relates to an on-premises component, cloud component, or in-between. Support is provided by

Microsoft for the cloud-hosted components and knowing when to take advantage of that support can mean a faster time to resolution. When troubleshooting Exchange Hybrid, issues are likely to arise in these areas:

- On-Premises Environment
- Office 365 Environment
- Network connecting On-Premises and Office 365
- Local clients (operating on the company WAN)
- External clients connecting to their mailbox

The first step towards resolution is to identify in which environment the problem belongs. Once this is done, you can use the appropriate tools and techniques to resolve the issue or gather the required logs for Microsoft to analyze. It would be impossible to cover every possible troubleshooting scenario for every Microsoft technology involved in an Exchange hybrid configuration, so the purpose of this chapter is to highlight the tools, logs, and methodologies used to resolve the most common hybrid break-fix scenarios. For a deeper dive into the various Hybrid components as well as designing, deploying, and managing a Hybrid Configuration, I recommend this eBook: Office 365 Complete Guide to Managing Hybrid Exchange Deployments

For information on Hybrid configurations, as well as how to operate Office 365 from the perspective of an Exchange professional, I recommend this eBook: Office 365 for Exchange Professionals

Additional References:

- Getting Started with Hybrid Exchange Deployment
- Configuring an Exchange 2013 Hybrid Deployment and Migrating to Office 365 (Exchange Online)
- Exchange Server Deployment Assistant

Note: For a great beginning-to-end overview of a Hybrid configuration, I highly recommend the Microsoft Virtual Academy course on Exchange Hybrid Deployments..

The Hybrid Configuration Wizard

When Microsoft released the first iteration of the Exchange Hybrid Configuration Wizard (HCW) in Exchange 2010 Service Pack 2, it replaced over 50 manual configuration steps previously required to build a Hybrid configuration. Today, Microsoft does not support a hybrid connection that is manually configured, primarily due to the multiple problems that can occur due to misconfiguration that lead to many support calls as well as a bad Exchange Hybrid experience.

While the first versions of the Hybrid Configuration Wizard had some issues, over the course of its four-year history, it has evolved into an extremely robust tool. The largest evolutionary step was the repackaging and rebranding as the Office 365 Hybrid Configuration Wizard in September 2015. The wizard is now maintained and downloaded separately instead of being packaged with each Exchange Cumulative Update. This means any issues with the code can be detected, remedied, tested, and published in a matter of days and means that the wizard always represents the latest state of knowledge about hybrid connectivity. Here's a summary of the major improvements in the new wizard:

- Up-To-Date Hybrid Experience - the latest version of the code is checked and downloaded every time the wizard is run. As changes are made to, or issues are fixed in the HCW, customers will see the benefits soon thereafter.

- Piloting Changes – because the Microsoft Hybrid team can roll out changes as quickly or slowly as they deem appropriate, the Hybrid experience should improve over time. Members of the Exchange Technology Adoption Program (TAP) or First Release customers can test new features early and provide feedback.
- Logging Sent to Microsoft – the new wizard records and shares data about hybrid connections with Microsoft. This data is used to correct any issues with the code as well as to guide the addition of improvements intended to result in fewer problems in the field.
- Improvements to Error Handling – Microsoft has focused on improving the error text and codes used by the wizard. Each page of the new HCW also gives the option to provide feedback to the Hybrid team at Microsoft.

Note: If you notice anything out of the ordinary when running the HCW, you are strongly encouraged to provide feedback using the built-in "Give Feedback" option. This feedback is closely monitored by the Hybrid Support, Development, and Engineering teams. On more than one occasion, the feedback received allowed Microsoft to quickly identify and correct customer issues encountered with the HCW.

If you want to opt out of uploading the Hybrid logs you can do that by using the registry key below on the machine were you are running the HCW from:

1. Navigate to the following location in the registry, create the path if needed:

- Exchange 2016: *HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\ExchangeServer\v16\Update-HybridConfiguration*
- Exchange 2013: *HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\ExchangeServer\v15\Update-HybridConfiguration*

2. Create REG_DWORD "DisableUploadLogs" with value 1

Walking through the Office 365 Hybrid Configuration Wizard

Let's walk through the steps involved in downloading and running the new HCW in a green field single-server Exchange 2016 environment. The process to create a hybrid configuration begins with navigating to the Hybrid feature pane within Exchange Admin Center (EAC) for your on-premises environment or from the Exchange Online Admin Center (Figure 10-1). After clicking Enable, you are prompted to authenticate to your Office 365 tenant and download the new Hybrid Configuration Wizard.

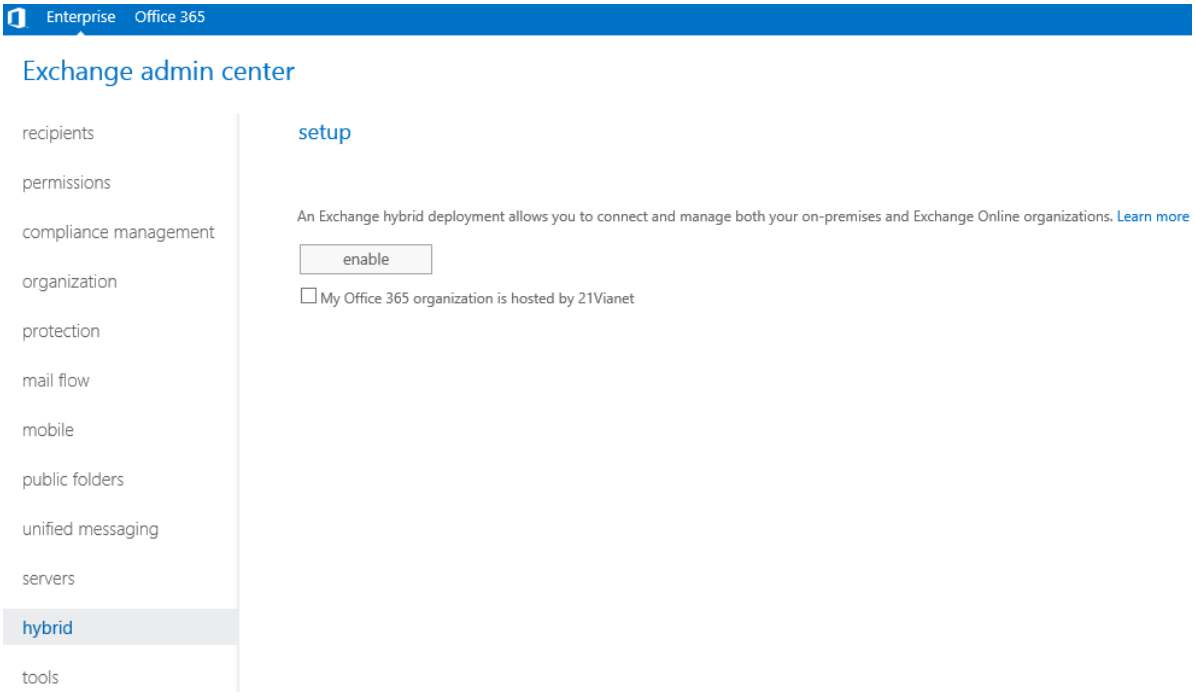


Figure 10-1: Enabling Hybrid in EAC

Note: While you might wish to keep a local copy of the HCW, it's possible (and likely) it could be outdated within a week's time. Therefore, it's highly recommended to always download the latest version when you need to run the HCW, whether for the first time or to make modifications to a Hybrid Configuration.

When you run the wizard (Figure 10-2), you have the option to learn more about what the wizard does and which settings it will modify depending on your input.

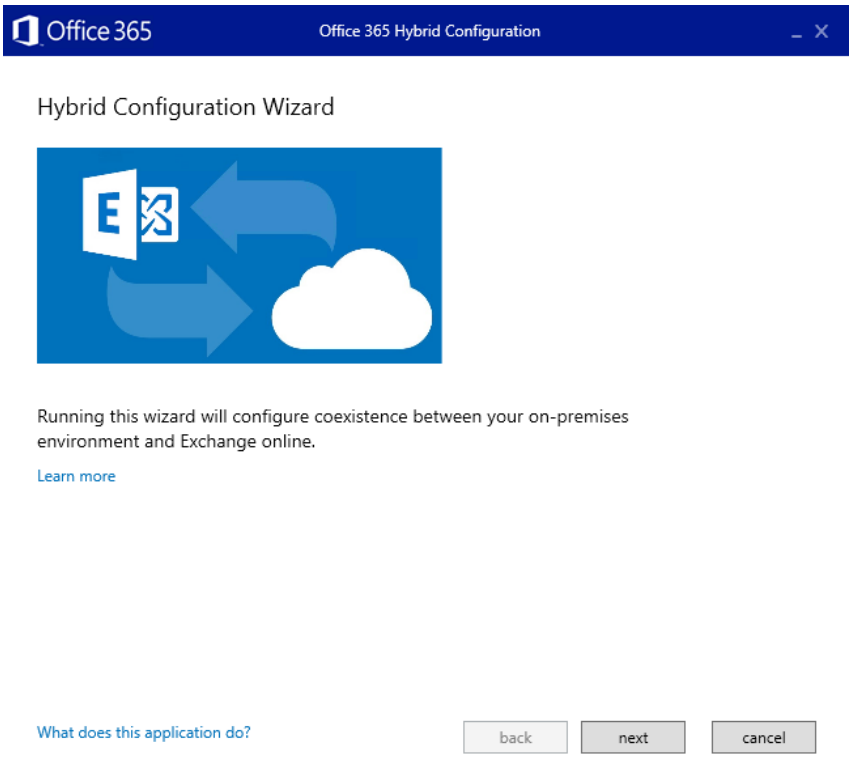


Figure 10-2: The first screen of the new Hybrid Configuration Wizard

The next screen of the HCW (Figure 10-3) allows you to specify which Exchange 2010, 2013, or 2016 server the HCW will connect to for management functionality.

Note: At the time of this writing, Exchange 2010 SP3, Exchange 2013 (CU9 or above), or Exchange 2016 can be used with the new HCW.

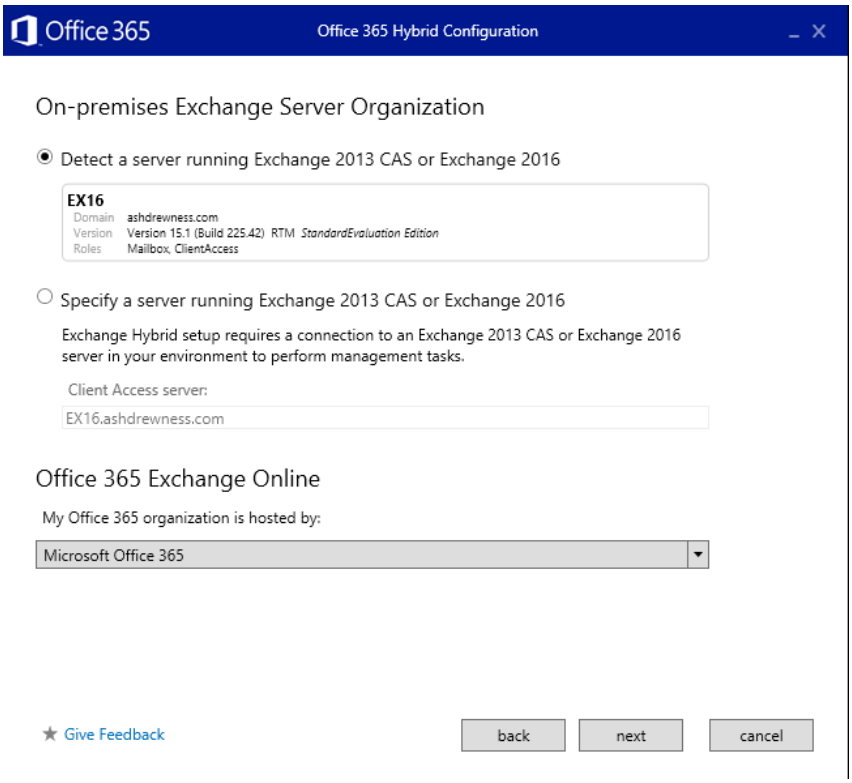


Figure 10-3: Selecting the on-premises server

The next screen (Figure 10-4) looks for credentials for an on-premises account which is a member of the Organization Management role group (if the currently logged-on user will not suffice) as well as administrative credentials for your Office 365 Tenant. The HCW then connects to both the on-premises Exchange organization and the Office 365 tenant to gather information and validate access rights. If network or internet proxy issues exist, then you may require additional network configuration to allow the HCW to access the Office 365 endpoints. It's highly recommended to view the Office 365 URLs and IP address ranges article to determine which endpoints are used. Understand that these IP's and URL's may change so I recommend subscribing to the RSS feed to receive notifications of changes.

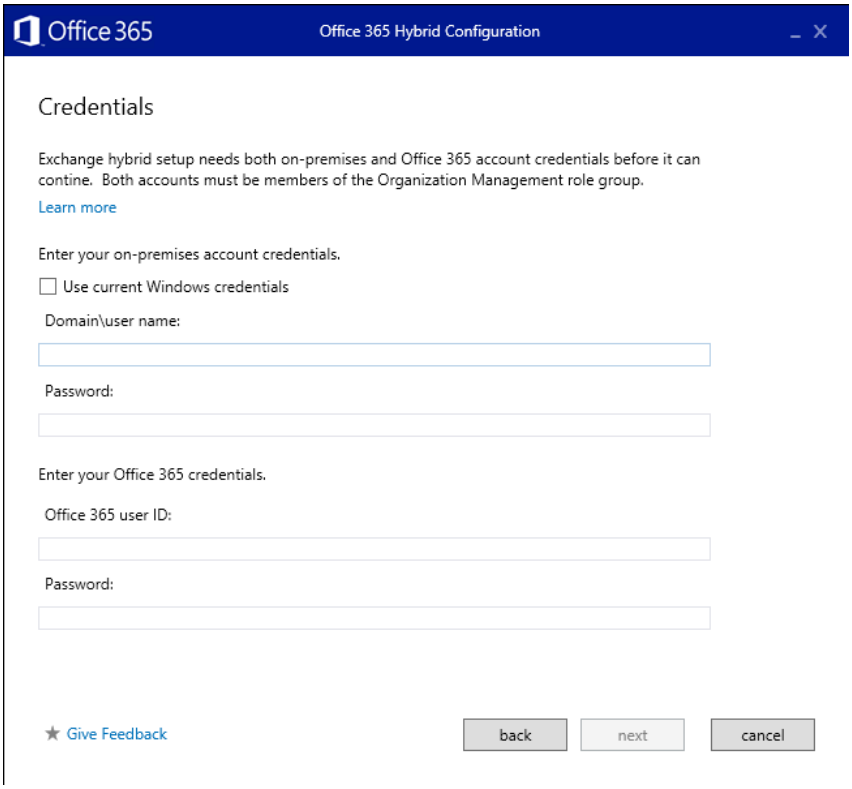


Figure 10-4: Providing administrative credentials

The Federation Trust section of the HCW asks whether Federation should be enabled. Federation is used for sharing calendar free/busy information with other Exchange organizations, amongst other things. From here the Hybrid Domains page will list the Accepted Domains found when the HCW queries both the Exchange Online tenant and the on-premises Exchange organization. My environment only has one domain, ashdrewness.com. However, if I also had ashdrewness.net and ashdrewness.org listed as Accepted Domains in my tenant and on-premises environment, all three domains would be displayed. I would be given the option to designate one as an Autodiscover Domain. Obviously, if a domain that you'd like to use as a shared namespace in the hybrid configuration is missing, you should verify that domain exists as an Accepted Domain in both the tenant and on-premises. This requires the tenant domain to have fully completed verification via TXT record. You can choose to stop the wizard and add the missing domains, or re-run the HCW at a later time. Going forward, when new domains are to be added to the hybrid configuration, the HCW can be re-run as needed.

Note: If you are not presented with the Hybrid Domains page then the HCW has only detected one domain which is present in both the tenant and on-premises as Accepted Domains. Therefore, there is no need to choose domains from the wizard's perspective. If this is not desired and multiple domains are to be added to the hybrid configuration, then verify all domains have been added as Accepted Domains in the tenant and on-premises.

Once Federation is enabled and domains are selected, the Domain Ownership screen is displayed (Figure 10-5). The domains to be used in this Hybrid Configuration must first pass validation that they are indeed owned by the on-premises organization. This is done by requiring a TXT record be created in the domain's zone with a unique token string. Creating this record validates the individual running the HCW actually owns (or has appropriate access to) this domain. It's recommended to use the "Copy to clipboard" option to ensure only the required text is copied. Then add a TXT record with the pasted token as its contents. Once this is done, it may take several minutes (or even hours) for the record to be propagated, at which point the "verify domain ownership" button can be clicked.

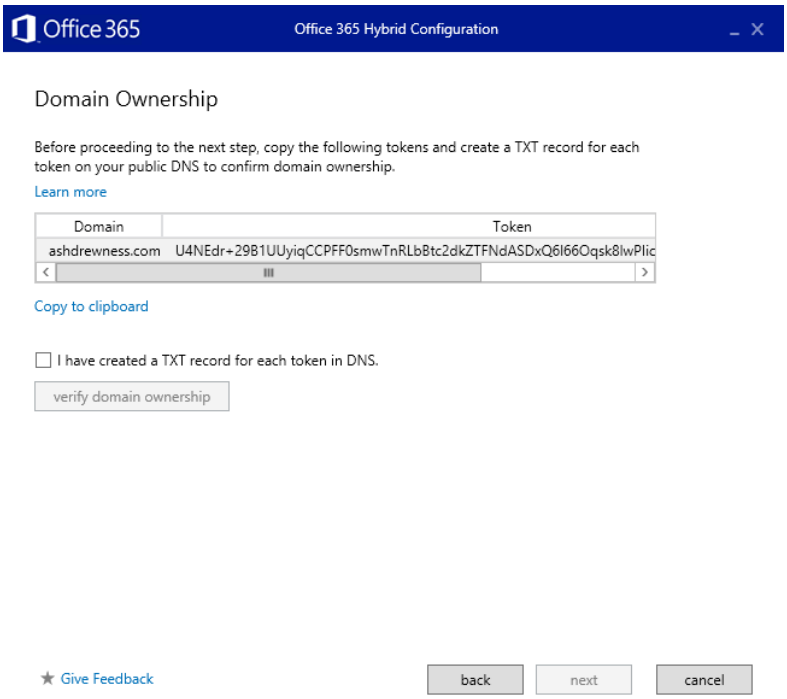


Figure 10-5: The HCW requesting I validate ownership of the domains used in the Hybrid Configuration

The HCW now asks if I utilize an Edge Transport Server for secure transport (Figure 10-6). By clicking "Advanced…" the option to enable Centralized Transport is presented. Centralized Transport ensures all outbound mail from both on-premises as well as the Office 365 Tenant are routed through the on-premises Exchange servers. By default, outbound mail from mailboxes in the tenant leave via Exchange Online Protection (EOP) to the intended recipient. Enabling Centralized Transport changes this behavior so that all outbound emails from Exchange Online are routed back on-premises. This is usually done for compliance reasons.

The two following screens (Receive Connector Configuration and Send Connector Configuration) allow you to select which Exchange servers are to be used for hosting Receive and Send Connectors to facilitate SMTP communications between the on-premises organization and the Office 365 tenant. For obvious reasons, these servers must be able to communicate to and from Office 365 over port 25. The next (Transport Certificate) screen asks which certificate will be used for SMTP communications to the Office 365 tenant. This certificate and private key must be installed on every server that will participate in hybrid mailflow, as well as be assigned to SMTP services on Exchange. The final transport screen (Organization FQDN) of the HCW asks which FQDN should be used by EOP when delivering mail to your on-premises organization.

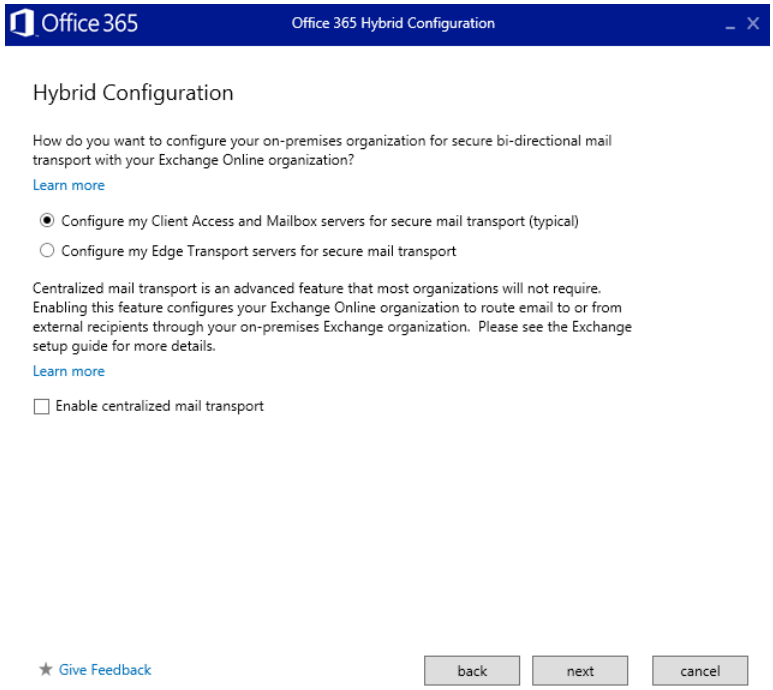


Figure 10-6: Choosing the secure transport option

If you have not configured an External URL for the Exchange Web Services (EWS) virtual directory, you may see the screen illustrated in Figure 10-7.

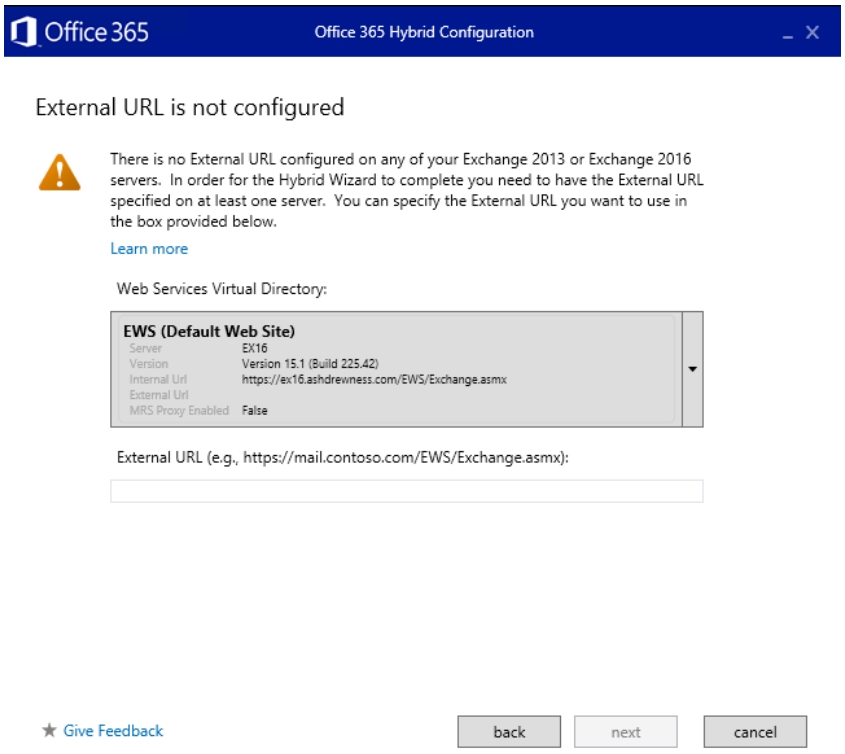


Figure 10-7: The absence of an EWS External URL causes the HCW to display a warning

Seeing this screen should be a red flag that the current environment hasn't been properly configured. Any issues should be remedied before proceeding to complete the hybrid configuration. However, in this case, we will manually provide an EWS URL and proceed forward see how the HCW reacts.

The “Ready for Update” screen (Figure 10-8) is now ready for us to click “Update.” It’s important to note that up to this point in the HCW there have been no modification made to the on-premises or Office 365 tenant environment. When we click “Update”, the HCW will begin making configuration changes based on the previous input. This configuration will be stored in the on-premises Active Directory and the HCW will read from AD to make the necessary modifications during the wizard’s execution.

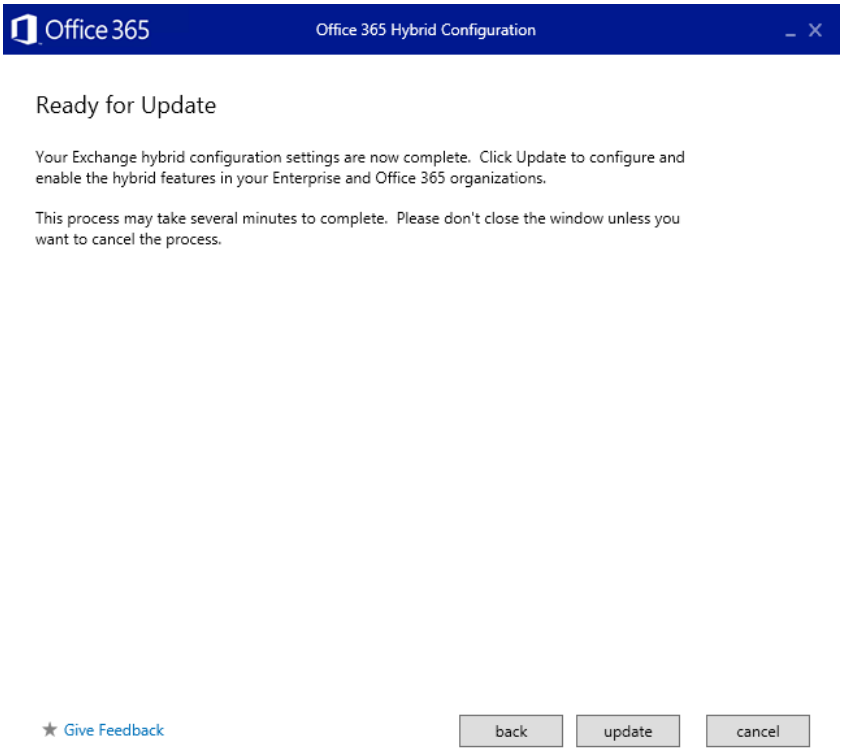


Figure 10-8: Pushing the changes to the on-premises organization and the Office 365 tenant

The HCW will now take several minutes to complete the requested configuration changes. The first time it runs it will enable the MRSProxy on all Exchange Servers in the organization and can take a very long time to complete in a globally distributed environment. Enabling it beforehand may lower the time to completion for the wizard. In either case, please be patient. In my case, the HCW encountered a failure (Figure 10-9) because even though it warned me before, I still have not configured an External URL for my EWS Virtual Directory.

I’ve never been known to be fond of error messages or failure in general, but there are two things about this notification that I like. First, I’m provided with hyperlinks at the bottom-left of the wizard to open Exchange PowerShell either for my on-premises organization or my Exchange Online environment in my Office 365 tenant. This can allow me to remedy any mistakes which may have caused the HCW to fail. In my case, I used the on-premises link to configure an External URL for my EWS Virtual Directory with the intent of resolving this issue. The other thing I like about this screen is that once I’ve made the necessary changes to correct my environment, I’m able to simply click “Back” to go to the “Ready for Update” screen once again and click “Update” once more. No need to restart the HCW, just two simple mouse clicks.

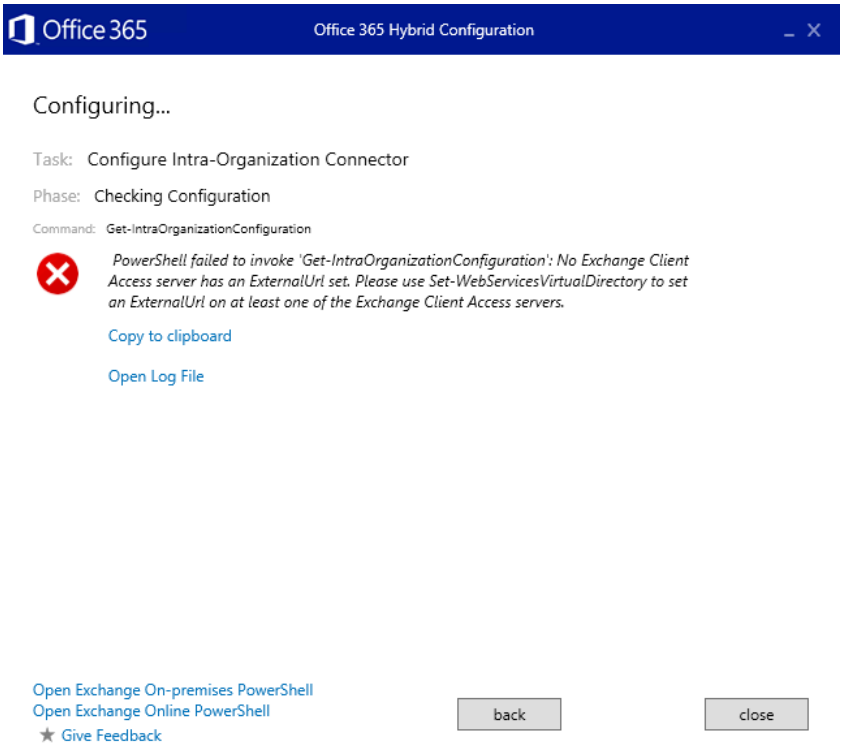


Figure 10-9: Failure encountered due to the lack of an External URL for the EWS Virtual Directory

This time, the HCW completed successfully and displayed the wizard completion screen (Figure 10-10).

Figure 10-10: Completion page where Feedback can be provided to the HCW team

Again, I highly suggest you take the time to provide feedback to the Hybrid Configuration Wizard team. Once the HCW has been run, the Hybrid Configuration may be updated by navigating to the Hybrid feature pane in

EAC and clicking "Update". This will prompt you to download the latest version of the HCW for execution again. The below scenarios would require running the Hybrid Configuration Wizard again:

- Changing a certificate
- Adding a new Exchange transport server
- Adding a domain to be used in the Hybrid Configuration

Office 365 Hybrid Configuration Wizard Logging

Since the HCW is a standalone application, the Exchange installation path logging location is not used. Logs for the HCW are now found in the location: *%appdata%\Microsoft\Exchange Hybrid Configuration*

The logs provide a record of each action taken during the wizard's execution. The following data is contained within the logs:

- User account the HCW was run as
- Time of the wizard's execution
- Build number of the HCW
- Supported releases of Exchange for use with the HCW
- Connectivity checks for the wizard's ability to access on-premises Exchange as well as the Office 365 Tenant
- Current Hybrid Configuration (if one currently exists)
- Information about SSL certificate used for Hybrid functions
- On-premises and Office 365 tenant Accepted Domains
- References to PowerShell commands executed to enable/update Hybrid Configuration

If you encounter any issue during the execution of the HCW and the error message displayed in the wizard GUI not be sufficient, I recommend reviewing the content of the logs to assist in identifying the root cause. Common issues encountered during the wizard's execution are:

- Incorrect permissions of the account used for the wizard's execution
- A proxy server or internet filter blocking communications between the HCW and the Office 365 tenant
- Corrupt objects in Active Directory
- Missing certificates
- Incorrectly configured Send/Receive connectors

While work is being continuously performed to make issues encountered during the wizard's execution easier to diagnose via easy to understand messages, the HCW logs may still be required to determine exactly what went wrong. For example, I removed my third-party certificate from the Exchange server's local certificate store and re-ran the HCW. I was greeted with this warning message:

"*No valid certificate could be found to use for securing hybrid mail transport. The certificate must be installed on all servers where Send or Receive connectors are hosted. After installing a valid certificate on all respective servers return to this step and push `search again'*"

If I then check the HCW logs, I can see the last action the wizard performed, which was to execute the *Get-ExchangeCertificate* cmdlet and gather all installed certificates on the selected Exchange servers. While the default self-signed certificates installed with Exchange as well as the Federation certificate are detected in the logs, a valid third-party certificate with the FQDN names used on the transport connectors was not found. I could then use this data to correlate the findings on my Exchange server using *Get-ExchangeCertificate*. In my scenario, after re-importing the third-party certificate, enabling it for SMTP, and clicking "Search Again" in the HCW, the certificate was detected and the wizard was allowed to continue.

So even in a scenario where the HCW gave a descriptive error message, the logs could still be used to determine exactly which command was executed and further aid in troubleshooting the issue.

Hybrid Configuration Components

Let's now take a look at some of the changes the HCW makes to the on-premises Exchange organization as well as the Office 365 tenant. Taking note of these settings can be useful for your records and to understand what's going on during later troubleshooting. I've provided screenshots to help serve as a simple reference, but your environment's settings might be different depending on the settings chosen in the HCW. Viewing the changes made to on-premises is as simple as opening Exchange Management Shell, but to connect to your Office 365 tenant you'll need to connect to Exchange Online with remote PowerShell. As PowerShell is often used to manage Exchange Online, the best idea is to include a function to connect in your PowerShell profile.

Once connected, let's look at the Hybrid Configuration object created on-premises using the Get-HybridConfiguration cmdlet (Figure 10-11).

```
[PS] C:\Windows\system32>Get-HybridConfiguration


RunspaceId                    : f3bf15eb-9d40-4590-9d36-33f8c15196b3
ClientAccessServers           : {}
EdgeTransportServers          : {}
ReceivingTransportServers     : {EX16}
SendingTransportServers       : {EX16}
OnPremisesSmartHost           : mail.ashdrewness.com
Domains                       : {ashdrewness.com}
Features                      : {FreeBusy, MoveMailbox, Mailtips, MessageTracking, OwaRedirection, OnlineArchive,
                                SecureMail, Photos}
ExternalIPAddresses           : {}
TlsCertificateName            : <I>CN=DigiCert SHA2 Secure Server CA, O=DigiCert Inc, C=US<S>CN=*.ashdrewness.com, OU=IT,
                                O=Andrew Higginbotham, L=Austin, S=Texas, C=US
ServiceInstance               : 0
AdminDisplayName              :
ExchangeVersion               : 0.20 (15.0.0.0)
Name                          : Hybrid Configuration
DistinguishedName             : CN=Hybrid Configuration,CN=Hybrid Configuration,CN=ASHExch,CN=Microsoft
                                Exchange,CN=Services,CN=Configuration,DC=ashdrewness,DC=com
Identity                      : Hybrid Configuration
Guid                          : bd398896-f36a-4e78-8bf2-f44bd7d73c48
ObjectCategory                : ashdrewness.com/Configuration/Schema/ms-Exch-Coexistence-Relationship
ObjectClass                   : {top, msExchCoexistenceRelationship}
WhenChanged                   : 12/23/2015 1:56:58 AM
WhenCreated                   : 12/23/2015 1:56:58 AM
WhenChangedUTC                : 12/23/2015 7:56:58 AM
WhenCreatedUTC                : 12/23/2015 7:56:58 AM
OrganizationId                :
Id                            : Hybrid Configuration
OriginatingServer             : EX16.ashdrewness.com
IsValid                       : True
ObjectState                   : Unchanged
```

Figure 10-11: Hybrid Configuration object in on-premises Active Directory

This object is a reference to the Hybrid Configuration itself and contains the servers, domains, transport services, and features used in the Hybrid Configuration. This object will only exist in the on-premises environment. While this object can be removed using the Remove-HybridConfiguration cmdlet, this will not actually remove hybrid features such as Organization Relationships, connectors, Remote Domains etc.

Note: Removing a HybridConfiguration object should typically only be performed in circumstances where the hybrid deployment state is corrupt and under the direction of Microsoft Customer Service and Support. After removing the HybridConfiguration object, your existing hybrid deployment configuration settings aren't disabled or removed. However, when the Hybrid Configuration wizard is run again after removing the HybridConfiguration object, the wizard won't have a hybrid configuration reference point for your existing feature settings. As a result, it will automatically create a HybridConfiguration object and record the new hybrid deployment configuration feature values defined in the wizard. The feature settings associated with the hybrid deployment, such as organization relationship or Send and Receive connector parameters, which were configured with the HybridConfiguration object that's removed, aren't removed or modified until the Hybrid Configuration wizard is run again.
Additional Reference:
Decommissioning your Exchange 2010 servers in a Hybrid Deployment

We'll next look at the Organization Relationships using the Get-OrganizationRelationship cmdlet (Figure 10-12).

```
[PS] C:\Windows\system32>Get-OrganizationRelationship | fl


RunspaceId             : f3bf15eb-9d40-4590-9d36-33f8c15196b3
DomainNames            : {Ashdrewness.mail.onmicrosoft.com}
FreeBusyAccessEnabled  : True
FreeBusyAccessLevel    : LimitedDetails
FreeBusyAccessScope    :
MailboxMoveEnabled     : True
DeliveryReportEnabled  : True
MailTipsAccessEnabled  : True
MailTipsAccessLevel    : All
MailTipsAccessScope    :
PhotosEnabled          : True
TargetApplicationUri   : outlook.com
TargetSharingEpr       :
TargetOwaURL           : http://outlook.com/owa/ashdrewness.com
TargetAutodiscoverEpr  : https://autodiscover-s.outlook.com/autodiscover/autodiscover.svc/WSSecurity
OrganizationContact    :
Enabled                : True
ArchiveAccessEnabled   : True
AdminDisplayName       :
ExchangeVersion        : 0.10 (14.0.100.0)
Name                   : On-premises to O365 - f08a73fc-59e7-4144-94d8-7341ee0f2075
DistinguishedName      : CN=On-premises to O365 -
                         f08a73fc-59e7-4144-94d8-7341ee0f2075,CN=Federation,CN=ASHExch,CN=Microsoft
                         Exchange,CN=Services,CN=Configuration,DC=ashdrewness,DC=com
Identity               : On-premises to O365 - f08a73fc-59e7-4144-94d8-7341ee0f2075
Guid                   : 68e740bd-1b78-4b5a-9c9c-0dd10f83cee3
ObjectCategory         : ashdrewness.com/Configuration/Schema/ms-Exch-Fed-Sharing-Relationship
ObjectClass            : {top, msExchFedSharingRelationship}
WhenChanged            : 12/23/2015 1:59:01 AM
WhenCreated            : 12/23/2015 1:58:58 AM
WhenChangedUTC         : 12/23/2015 7:59:01 AM
WhenCreatedUTC         : 12/23/2015 7:58:58 AM
OrganizationId         :
Id                     : On-premises to O365 - f08a73fc-59e7-4144-94d8-7341ee0f2075
OriginatingServer      : EX16.ashdrewness.com
IsValid                : True
ObjectState            : Unchanged
```

Figure 10-12: Organization Relationship object in on-premises Active Directory

This cmdlet displays the properties of the Organization Relationship the on-premises Exchange organization has with the Office 365 tenant Exchange organization. This information is used for sharing calendar free/busy information between each organization. Remember, the end goal of a Hybrid Configuration is to have an on-premises Exchange organization and an Office 365 tenant appear to the users as one entity. Allowing cloud mailboxes to query free/busy information of on-premises mailboxes (and vice versa) is one feature which makes that illusion possible. To view the Organization Relationship in the Office 365 tenant, run the same command in the remote PowerShell session to your tenant (Figure 10-13).

```
PS C:\> Get-OrganizationRelationship | fl

RunspaceId             : 120f8a5e-8c6d-44b5-90ff-90ff5e6ad8a3
DomainNames            : {ashdrewness.com}
FreeBusyAccessEnabled  : True
FreeBusyAccessLevel    : LimitedDetails
FreeBusyAccessScope    :
MailboxMoveEnabled     : False
DeliveryReportEnabled  : True
MailTipsAccessEnabled  : True
MailTipsAccessLevel    : All
MailTipsAccessScope    :
PhotosEnabled          : True
TargetApplicationUri   : FYDIBOHF25SPDLT.ashdrewness.com
TargetSharingEpr       :
TargetOwaURL           :
TargetAutodiscoverEpr  : https://autodiscover.ashdrewness.com/autodiscover/autodiscover.svc/WSSecurity
OrganizationContact    :
Enabled                : True
ArchiveAccessEnabled   : False
AdminDisplayName       :
ExchangeVersion        : 0.10 (14.0.100.0)
Name                   : O365 to On-premises - f08a73fc-59e7-4144-94d8-7341ee0f2075
DistinguishedName      : CN=O365 to On-premises - f08a73fc-59e7-4144-94d8-7341ee0f2075,CN=Federation,CN=Configuration,CN
                         =Ashdrewness.onmicrosoft.com,CN=ConfigurationUnits,DC=NAMPR04A001,DC=prod,DC=outlook,DC=com
Identity               : O365 to On-premises - f08a73fc-59e7-4144-94d8-7341ee0f2075
Guid                   : 5035b1bf-e0fc-45b2-8d5d-2b95cda18a41
ObjectCategory         : NAMPR04A001.prod.outlook.com/Configuration/Schema/ms-Exch-Fed-Sharing-Relationship
ObjectClass            : {top, msExchFedSharingRelationship}
WhenChanged            : 12/23/2015 1:59:13 AM
WhenCreated            : 12/23/2015 1:58:59 AM
WhenChangedUTC         : 12/23/2015 7:59:13 AM
WhenCreatedUTC         : 12/23/2015 7:58:59 AM
OrganizationId         : NAMPR04A001.prod.outlook.com/Microsoft Exchange Hosted
                         Organizations/Ashdrewness.onmicrosoft.com -
                         NAMPR04A001.prod.outlook.com/ConfigurationUnits/Ashdrewness.onmicrosoft.com/Configuration
Id                     : O365 to On-premises - f08a73fc-59e7-4144-94d8-7341ee0f2075
OriginatingServer      : BN1PR04A001DC05.NAMPR04A001.prod.outlook.com
IsValid                : True
ObjectState            : Unchanged
```

Figure 10-13: Organization Relationship object in Azure Active Directory for your Office 365 tenant

Next we'll look at the custom Remote Domains configured by the HCW for the ".onmicrosoft.com" and "mail.onmicrosoft.com" domains using the Get-RemoteDomain cmdlet (Figure 10-14).

```
[PS] C:\>Get-RemoteDomain

Name                                DomainName                                AllowedOOFType
----                                ----------                                --------------
Default                             *                                         External
Hybrid Domain - Ashdrewness...      Ashdrewness.mail.onmicrosoft.com          External
Hybrid Domain - Ashdrewness...      Ashdrewness.onmicrosoft.com               External
```

Figure 10-14: Custom Remote Domains configured on-premises

Remote Domains are used for controlling message formatting, automatic reply settings, and NDR information. Since Exchange uses these onmicrosoft.com SMTP domains for routing mail to the Office 365 tenant, these Remote Domains allow the mail to be treated (and formatted) as if it were being sent internally. Further keeping the appearance of one singular Exchange organization.

OAUTH and Federation

The HCW configures an IntraOrganizationConnector in both on-premises and the Office 365 tenant to support features such as eDiscovery in a Hybrid environment. These features rely on OAUTH, which the HCW can also configure. When your environment is comprised entirely of Exchange 2013 SP1 and later servers, the HCW will automatically configure and enable OAUTH. If the environment contains legacy Exchange servers, the HCW notify you that it cannot configure OAUTH. However, you can manually configure OAUTH for these servers,.

OAUTH is an open-source secure server-to-server authorization framework which is used by Exchange, SharePoint, Skype for Business, and other applications. For Exchange 2013 mailboxes, OAUTH will not only be used for eDiscovery queries but also for cross-organization free/busy requests which bypass Organization Relationships and the Microsoft Federation Gateway (the legacy method). Instead, an IntraOrganization Connector will be used (IOC). Figure 10-15 displays the IntraOrganization Connector that exists on-premises.

```
[PS] C:\>Get-IntraOrganizationConnector | fl

RunspaceId            : f3bf15eb-9d40-4590-9d36-33f8c15196b3
TargetAddressDomains  : {Ashdrewness.mail.onmicrosoft.com}
DiscoveryEndpoint     : https://autodiscover-s.outlook.com/autodiscover/autodiscover.svc
Enabled               : True
AdminDisplayName      :
ExchangeVersion       : 0.20 (15.0.0.0)
Name                  : HybridIOC - f08a73fc-59e7-4144-94d8-7341ee0f2075
DistinguishedName     : CN=HybridIOC - f08a73fc-59e7-4144-94d8-7341ee0f2075,CN=Intra Organization
                        Connectors,CN=ASHExch,CN=Microsoft Exchange,CN=Services,CN=Configuration,DC=ashdrewness,DC=com
Identity              : HybridIOC - f08a73fc-59e7-4144-94d8-7341ee0f2075
Guid                  : cbc1281e-908f-41a3-8b47-a793749febf8
ObjectCategory        : ashdrewness.com/Configuration/Schema/ms-Exch-Intra-Organization-Connector
ObjectClass           : {top, msExchIntraOrganizationConnector}
WhenChanged           : 12/23/2015 2:04:21 AM
WhenCreated           : 12/23/2015 2:04:21 AM
WhenChangedUTC        : 12/23/2015 8:04:21 AM
WhenCreatedUTC        : 12/23/2015 8:04:21 AM
OrganizationId        :
Id                    : HybridIOC - f08a73fc-59e7-4144-94d8-7341ee0f2075
OriginatingServer     : EX16.ashdrewness.com
IsValid               : True
ObjectState           : Unchanged
```

Figure 10-15: IntraOrganization connector that exists on-premises

This connector is used whenever queries are issued for the ashdrewness.mail.onmicrosoft.com domain, which is the TargetAddress for Exchange Online mailboxes that have a mail-enabled user account on-premises (this requires directory synchronization to be enabled; more on this later in the chapter).

Figure 10-16 displays the IntraOrganization Connector which exists in the Office 365 tenant. This connector would be used whenever queries are issued for the ashdrewness.com domain, which is the TargetAddress for on-premises mailboxes that have a mail-enabled user account in the Office 365 tenant.

```
PS C:\> Get-IntraOrganizationConnector | fl


RunspaceId            : 495338fd-47b5-4362-af5c-5c0294b5d594
TargetAddressDomains  : {ashdrewness.com}
DiscoveryEndpoint     : https://mail.ashdrewness.com/autodiscover/autodiscover.svc
Enabled               : True
AdminDisplayName      :
ExchangeVersion       : 0.20 (15.0.0.0)
Name                  : HybridIOC - f08a73fc-59e7-4144-94d8-7341ee0f2075
DistinguishedName     : CN=HybridIOC - f08a73fc-59e7-4144-94d8-7341ee0f2075,CN=Intra Organization Connectors,CN=Configur
                        ation,CN=Ashdrewness.onmicrosoft.com,CN=ConfigurationUnits,DC=NAMPR04A001,DC=prod,DC=outlook,DC=
                        com
Identity              : HybridIOC - f08a73fc-59e7-4144-94d8-7341ee0f2075
Guid                  : 41ed13ca-7146-40db-8abd-6452ad70b041
ObjectCategory        : NAMPR04A001.prod.outlook.com/Configuration/Schema/ms-Exch-Intra-Organization-Connector
ObjectClass           : {top, msExchIntraOrganizationConnector}
WhenChanged           : 12/23/2015 2:04:27 AM
WhenCreated           : 12/23/2015 2:04:22 AM
WhenChangedUTC        : 12/23/2015 8:04:27 AM
WhenCreatedUTC        : 12/23/2015 8:04:22 AM
OrganizationId        : NAMPR04A001.prod.outlook.com/Microsoft Exchange Hosted
                        Organizations/Ashdrewness.onmicrosoft.com -
                        NAMPR04A001.prod.outlook.com/ConfigurationUnits/Ashdrewness.onmicrosoft.com/Configuration
Id                    : HybridIOC - f08a73fc-59e7-4144-94d8-7341ee0f2075
OriginatingServer     : BN1PR04A001DC05.NAMPR04A001.prod.outlook.com
IsValid               : True
ObjectState           : Unchanged
```

Figure 10-16: IntraOrganization connector that exists in the Office 365 tenant

The IntraOrganization Connectors might appear to serve the same purpose as Organization Relationships. It seems that way because it is true. The difference being that Exchange 2013/2016 mailboxes will utilize the IntraOrganization Connectors and OAUTH for authentication while Exchange 2010 mailboxes will utilize Organization Relationships and the Microsoft Federation Gateway for authentication. Of course, if OAUTH is not configured or an IntraOrganization Connector for the target domain is missing, Exchange 2013/2016 mailboxes will use the legacy Organization Relationship method. If an Organization Relationship is missing, Exchange 2010/2013/2016 will all fall back to use an Availability Address Space. Of course, in a functional Hybrid Configuration, an Organization Relationship should not be missing. An Availability Address Space is a legacy method for cross-forest free/busy using a service account or trust to mitigate permissions across forests. The HCW creates an Availability Address Space on-premises for the mail.onmicrosoft.com address space (Figure 10-17). This is actually a special "InternalProxy" access method, which resolves to a local 2013 SP1 or newer Exchange server to proxy the availability request to Office 365. This is used for legacy Exchange servers to proxy their queries through the newer versions of Exchange.

```
[PS] C:\>Get-AvailabilityAddressSpace | fl

RunspaceId             : f3bf15eb-9d40-4590-9d36-33f8c15196b3
ForestName             : Ashdrewness.mail.onmicrosoft.com
UserName               :
UseServiceAccount      : True
AccessMethod           : InternalProxy
ProxyUrl               : https://ex16.ashdrewness.com/EWS/Exchange.asmx
TargetAutodiscoverEpr  :
ParentPathId           : CN=Availability Configuration
AdminDisplayName       :
ExchangeVersion        : 0.1 (8.0.535.0)
Name                   : Ashdrewness.mail.onmicrosoft.com
DistinguishedName      : CN=Ashdrewness.mail.onmicrosoft.com,CN=Availability Configuration,CN=ASHExch,CN=Microsoft
                         Exchange,CN=Services,CN=Configuration,DC=ashdrewness,DC=com
Identity               : Ashdrewness.mail.onmicrosoft.com
Guid                   : 3f9e8b58-a865-449f-9cba-aefa514c3111
ObjectCategory         : ashdrewness.com/Configuration/Schema/ms-Exch-Availability-Address-Space
ObjectClass            : {top, msExchAvailabilityAddressSpace}
WhenChanged            : 12/23/2015 1:59:04 AM
WhenCreated            : 12/23/2015 1:59:04 AM
WhenChangedUTC         : 12/23/2015 7:59:04 AM
WhenCreatedUTC         : 12/23/2015 7:59:04 AM
OrganizationId         :
Id                     : Ashdrewness.mail.onmicrosoft.com
OriginatingServer      : EX16.ashdrewness.com
IsValid                : True
ObjectState            : Unchanged
```

Figure 10-17: Availability Address Space on-premises

It should be noted however, that once Exchange 2013/2016 detects the presence of an IOC and OAUTH, should an issue subsequently arise with either, it will never failover to using an Organization Relationship and the Microsoft Federation Gateway. Similarly, if Exchange 2010 detects the presence of an Organization Relationship, if a failure is encountered, it will not failover to using an Availability Address Space.

Let's walk through a few example scenarios for how free/busy requests work between on-premises and Office 365 tenants.

Scenario #1: In the Ashdrewness Company, Exchange 2013 connects to Office 365 in a Hybrid Configuration. An on-premises Exchange 2013 mailbox (UserA) wishes to query the Free/Busy information of a mailbox in Office 365 (UserB) which is also in Ashdrewness Company. OAUTH is configured and an IntraOrganization Connector exists (created by the HCW). The process would be as follows:

- UserA@ashdrewness.com uses Outlook to query the free/busy status of UserB@ashdrewness.com via the Scheduling Assistant
- The Outlook client contacts the on-premises EWS endpoint via either the Internal or External EWS URL value configured and distributed to Outlook via Autodiscover
- On-premises Exchange determines UserB@ashdrewness.com is a mail-enabled user in the local Active Directory with a TargetAddress of UserB@ashdrewness.mail.onmicrosoft.com
- Because UserA is on Exchange 2013 SP1 or later, it searches for the configuration of OAUTH as well as the presence of an IntraOrganization connector for ashdrewness.mail.onmicrosoft.com
- On-premises Exchange uses the Discovery Endpoint in the IntraOrganization connector to query the remote organization (our Office 365 tenant in this scenario) for an EWS endpoint. Authentication for this request uses OAUTH
- On-premises Exchange retrieves the free/busy information for UserB and returns the data to UserA's Outlook client

Scenario #2: In the Contoso Company, Exchange 2013 and Exchange 2010 connect to Office 365 in a Hybrid configuration. An on-premises Exchange 2010 mailbox (UserA) wishes to query the Free/Busy information of a mailbox in Office 365 (UserB) which is also in Contoso Company. OAUTH is configured and an IntraOrganization Connector exists (manually created by the administrator, as the HCW will only enable OAUTH in a pure Exchange 2013 CU5 or later environment). The process would be as follows:

- UserA@contoso.com uses Outlook to query the free/busy status of UserB@contoso.com via the Scheduling Assistant

- The Outlook client contacts the on-premises EWS endpoint via either the Internal or External EWS URL value configured and distributed to Outlook via Autodiscover
- On-premises Exchange determines UserB@contoso.com is a mail-enabled user in the local Active Directory with a TargetAddress of UserB@contoso.mail.onmicrosoft.com
- Because UserA is on Exchange 2010, it searches for the presence of an Organization Relationship for contoso.mail.onmicrosoft.com
- On-premises Exchange uses the TargetAutodiscoverEpr in the Organization Relationship to query the remote organization (our Office 365 tenant in this scenario) for an EWS endpoint. Authentication for this request uses the Microsoft Federation Gateway
- On-premises Exchange retrieves the free/busy information for UserB and returns the data to UserA's Outlook client

Scenario#3: In the Ashdrewness Company, Exchange 2013 connects to Office 365 in a Hybrid Configuration. An on-premises Exchange 2013 mailbox (UserA) wishes to query the Free/Busy information of a mailbox in Office 365 (UserB) which is also in Ashdrewness Company. OAUTH is configured and an IntraOrganization Connector exists (created by the HCW). The process would be as follows:

- UserA@ashdrewness.com uses Outlook to query the free/busy status of UserB@ashdrewness.com via the Scheduling Assistant
- The Outlook client contacts the on-premises EWS endpoint via either the Internal or External EWS URL value configured and distributed to Outlook via Autodiscover
- On-premises Exchange determines UserB@ashdrewness.com is a mail-enabled user in the local Active Directory with a TargetAddress of UserB@ashdrewness.mail.onmicrosoft.com
- Because UserA is on Exchange 2013 SP1 or later, it searches for the configuration of OAUTH as well as the presence of an IntraOrganization connector for ashdrewness.mail.onmicrosoft.com
- On-premises Exchange uses the Discovery Endpoint in the IntraOrganization connector to query the remote organization (our Office 365 tenant in this scenario) for an EWS endpoint. Authentication for this request attempts to use OAUTH but the certificate was recently changed or expired, therefore OAUTH fails
- Even though an Organization Relationship for ashdrewness.mail.onmicrosoft.com exists on-premises, it will not be used and the Free/Busy request will fail. If an IOC is present, Exchange 2013/2016 will not attempt to use an Organization Relationship, even if the IOC is misconfigured

Note: The process is similar when an Office 365 mailbox attempts to query free/busy information for an on-premises mailbox. Because office 365 mailboxes are on Exchange 2016 (or newer), they will always use IOC/OAUTH if it has been configured.

As previously mentioned, when OAUTH is not used, Exchange relies on Organization Relationships and the Microsoft Federation Gateway (MFG). A Federation Trust is established by the on-premises organization with the MFG (the HCW performs this action). Since each Office 365 tenant already trusts the MFG, the on-premises organization and the Office 365 tenant now have a mutual trust. The on-premises organization will federate the company's domain names while the Office 365 tenant will federate the <domain>.mail.onmicrosoft.com domains. The Federation Information for these domains contains their Autodiscover endpoints. With these in place, authentication can now occur between these two federated domains via the issuance of tokens. Each of these federated domains can be seen below in Figure 10-18.

```
[PS] C:\>Get-FederationInformation

cmdlet Get-FederationInformation at command pipeline position 1
Supply values for the following parameters:
DomainName: ashdrewness.com


RunspaceId             : f3bf15eb-9d40-4590-9d36-33f8c15196b3
TargetApplicationUri   : PYDIBOHF25SPDLT.ashdrewness.com
DomainNames            : {ashdrewness.com}
TargetAutodiscoverEpr  : https://ashdrewness.com/autodiscover/autodiscover.svc/WSSecurity
TokenIssuerUris        : {urn:federation:MicrosoftOnline}
Identity               :
IsValid                : True
ObjectState            : Unchanged


[PS] C:\>Get-FederationInformation

cmdlet Get-FederationInformation at command pipeline position 1
Supply values for the following parameters:
DomainName: ashdrewness.mail.onmicrosoft.com


RunspaceId             : f3bf15eb-9d40-4590-9d36-33f8c15196b3
TargetApplicationUri   : outlook.com
DomainNames            : {Ashdrewness.onmicrosoft.com, Ashdrewness.mail.onmicrosoft.com}
TargetAutodiscoverEpr  : https://autodiscover-s.outlook.com/autodiscover/autodiscover.svc/WSSecurity
TokenIssuerUris        : {urn:federation:MicrosoftOnline}
Identity               :
IsValid                : True
ObjectState            : Unchanged
```

Figure 10-18: Federated domains involved in this Hybrid Configuration

> **Note:** The Federation is established using a self-signed certificate with a Subject Name of CN=Federation which is enabled for the Federation service. These actions are performed by the Hybrid Configuration Wizard. If this certificate expires or is removed, the Federation trust will be broken.

As you can see, understanding how free/busy requests are processed in Hybrid is vital to troubleshooting free/busy issues from on-premises to Office 365 and vice versa. The following are common issues with free/busy in a Hybrid configuration and tips for troubleshooting them:

Expired or invalid certificate used for Federation (when not using OAUTH):

- Run Get-FederationTrust to view the certificate used for Federation
- If the current certificate assigned to the trust is missing or expired, use Set-FederationTrust to assign a new certificate to the trust, similar to the processed used for refreshing the Federation Trust. For example:
    - *Set-FederationTrust -Identity "Microsoft Federation Gateway" -Thumbprint <Thumbprint of new certificate>*
    - *Set-FederationTrust -Identity "Microsoft Federation Gateway" –PublishFederationCertificate*
    - These commands will define the new certificate and enable its use.
- Alternatively, re-running the HCW may resolve this without manually specifying the new certificate
- Use Test-FederationTrust to verify the trust
    - *Test-FederationTrust -UserIdentity <On-PremisesMailbox>*

Expired or invalid certificate used for Autodiscover/EWS endpoints:

- Use Get-ExchangeCertificate to verify the following:
    - **IsSelfSigned parameter -** This parameter value should be *False*
    - **RootCAType parameter** - This parameter value should be *Third Party*
    - **Services parameter** - This parameter value should be *IIS, SMTP*
    - **NotAfter parameter** - This parameter value is the certificate expiration date. The date listed here should not be expired

The troubleshooting process should be the same as an on-premises solution, verify the certificate is valid, assigned to IIS, and the EWS External URL value is listed on the certificate. See the Client Access Services chapter for reference in troubleshooting endpoints.

Free/Busy failures using OAUTH:

- If OAUTH was manually configured, verify the spelling and formatting of all URL's and that the certificate uploaded to Azure Active Directory Access Control Services is not expired.
- The following command verifies the Service Principal is still present
    - *Get-MsolServicePrincipal -ServicePrincipalName 00000002-0000-0ff1-ce00-000000000000 | fl*
- The following command can verify the credentials created during manual OAUTH configuration is still valid
    - *Get-MsolServicePrincipalCredential -AppPrincipalId 00000002-0000-0ff1-ce00-000000000000*
- Use Test-OauthConnectivity to validate connectivity and authentication for OAUTH
- Of course, re-running the HCW should correct most issues with OAUTH configuration, assuming OAUTH was not manually configured due to legacy Exchange Servers being present.

Remote or local AutoDiscover/EWS endpoints not reachable:

- Use the Microsoft Remote Connectivity Analyzer to validate AutoDiscover and EWS connectivity for mailboxes on-premises and in the Office 365 tenant
- For EWS Issues, run the *Synchronization, Notification, Availability, and Automatic Replies (OOF)* test in the *Microsoft Exchange Web Services Connectivity Tests* section, and verify that there aren't any errors. If errors occur, correct the items that the test identified
- For AutoDiscover issues, run the *Outlook Test E-mail AutoConfiguration* test in the *Microsoft Office Outlook Connectivity Tests* section, and verify that there aren't any errors. If errors occur, correct the items that the test identified
- From the Exchange server itself, verify you can reach the target AutoDiscover and EWS endpoints using Internet Explorer and not receive a certificate warning
- When verifying access to Office 365 EWS endpoints, it's recommended to use the Microsoft Remote Connectivity Analyzer to first query AutoDiscover. It will then provide a specific application server to use for EWS queries
- If queries from Office 365 to on-premises fail, verify a valid ExternalURL is configured for the Web Services Virtual Directory. Also verify this URL is reachable from outside your environment
- Please refer to the Client Access Services chapter for further tips on troubleshooting Exchange endpoints.

Cannot query free/busy information for newly created mailbox:

- Verify an Azure AD Connect (Formerly Azure AD Sync/DirSync) synchronization has been performed and the mailbox exists in the remote Exchange organization
- Often times, users are created on-premises, mailbox-enabled, a directory synchronization is performed, and their mailbox is moved to the Office 365 tenant. If a process other than this was performed (such as the mailbox being created directly in Office 365) then a free/busy request may fail

Additional References:

- Office 365 Exchange Hybrid Deployment
- Configure OAuth authentication with SharePoint 2013 and Lync 2013
- Troubleshoot a hybrid deployment
- Automated Hybrid Troubleshooting Experience

A quick note on Hybrid Mail flow and Exchange Online Protection

Depending on your selections in the Hybrid Configuration Wizard, transport connectors will be created both on-premises and in the Office 365 tenant to enable secure mail flow between the two organizations. Much of the troubleshooting tools, tips, and techniques discussed in the Transport Services chapter still apply to Hybrid configurations (tracking mail, port connectivity, etc.). However, the following tools, articles, and blogs will provide a good additional reference for troubleshooting Hybrid mail flow.

- Mail Flow Guided Walkthrough for Office 365
- Mail flow in EOP
- Configure mail flow using connectors in Office 365
- Mail flow best practices for Exchange Online and Office 365
- Troubleshoot Exchange Hybrid Mail Flow with Office 365
- Get-ReceiveConnector (On-Premises)
- Get-SendConnector (On-Premises)
- Get-InboundConnector (Office 365)
- Get-OutboundConnector (Office 365)

Note: Hybrid Mailbox Move troubleshooting, another feature of Hybrid Configurations, will be covered in the Migration chapter.

Directory Synchronization

The ability to synchronize directory objects between organizations has existed for some time and has been provided by various Microsoft and third-party tools. The available Microsoft toolsets have seen extensive change and rebranding since 2010:

- Forefront Identity Manager (FIM) 2010 R2
- Azure Active Directory Synchronization Tool (DirSync)
- Azure Active Directory Sync (AADSync)
- Azure Active Directory Connect (Azure AD Connect)
- Microsoft Identity Manager (MIM) 2016

The rapid change in product naming and confusion over which tool is used under which circumstances has led to many Exchange professionals to fear directory synchronization. In addition, few administrators may feel confident in troubleshooting directory synchronization when problems arise. With Directory Synchronization, most issues arise during deployment, which emphasizes the need to achieve a proper configuration to ensure a healthy installation. To be honest, once synchronization has been deployed it usually simply works. With the release of Azure AD Connect, much of the complexity has been removed during deployment and has also made the more complex migration scenarios (such as migrating from a legacy tool such as DirSync) much easier.

Microsoft likes to market Azure AD Connect as enabling customers to connect their on-premises Active Directory to Microsoft's cloud "with only 4 clicks." They aren't exaggerating either, as a simple Exchange environment can be integrated with Azure Active Directory/Office 365 in 4 clicks and usually less than an hour.

The green field Exchange 2016 environment I've used to demonstrate a Hybrid Configuration in this chapter has still not had a directory synchronization performed against it. That's right, you can run the Hybrid Configuration Wizard successfully without any form or directory synchronization be performed against it. In theory, a functional Hybrid Configuration can exist by manually creating/updating directory objects between

on-premises Active Directory and the Office 365 Tenant. However, this is certainly not recommended, but only serves to show the purpose of Directory Synchronization; to easily enable the synchronization of directory objects numbering in the hundreds of thousands without requiring manual intervention.

Note: At the time of this writing, one limitation of Azure AD Connect is its inability to synchronize Mail-Enabled Public Folders from on-premises to Office 365. This blog post from Exchange MVP Michael Van Horenbeeck has more details.

Preparation and configuration of Azure AD Connect

Let's walk through an installation of Azure AD Connect to demonstrate how quickly we can have Directory Synchronization operational. At the time of this writing, a new Directory Synchronization enablement experience is being rolled out to tenants. It's a new wizard-driven guided walkthrough of the following:

- Help determine whether directory synchronization is right for your environment (Figure 10-19)
- Perform a check to verify on-premises domains to be synchronized have been added to the tenant and verified (see Note)
- You're given the option to download and run the idFix tool against your on-premises Active Directory
- Download and install Azure AD Connect as well as help verify synchronization has occurred

Note: "Let's check your Directory" performs the following actions:

- Scans on-premises Active Directory for user count, object count, Exchange mailbox count, Exchange mail contact count, Exchange distribution group count, Accepted Domains and UPN suffixes information.
- Verifies if current login user is domain joined user
- Verifies Accepted Domain and UPN suffixes are publicly routable and current tenant domain verification status. Also, automatically adds non-verified routable domain to tenant's domain list, so that tenant can verify those domains.

The purpose of the idFix tool is to identify and remedy issues with on-premises directory objects which would prevent a successful synchronization; such as duplicate objects or invalid characters.

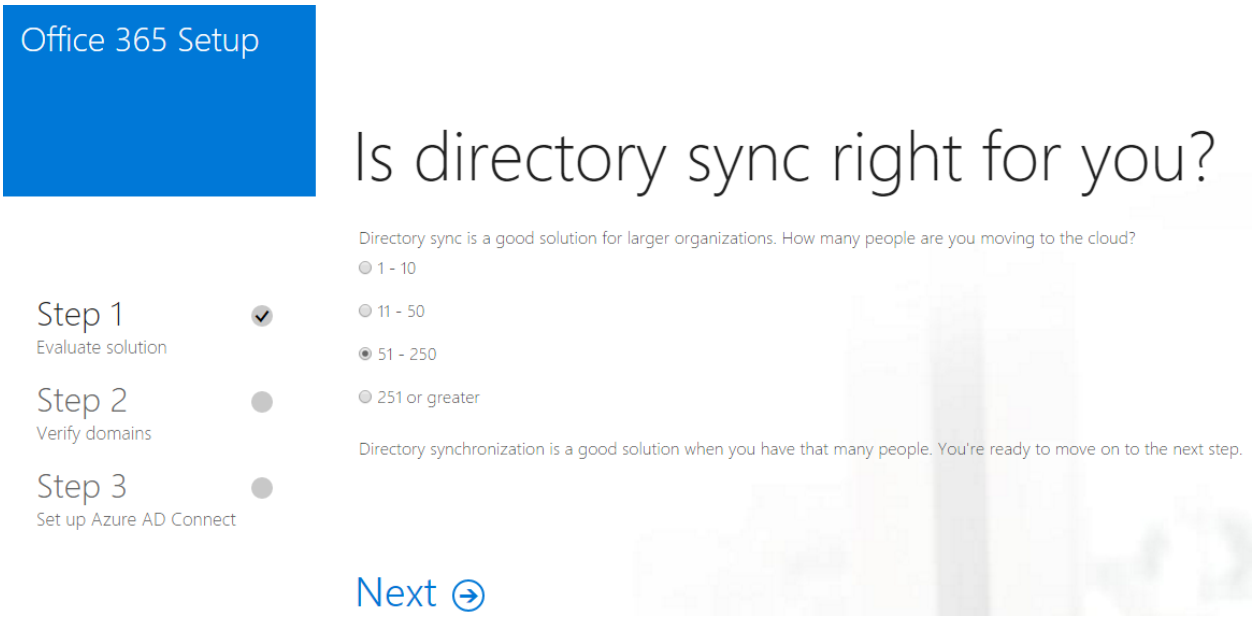


Figure 10-19: Directory Synchronization management page in Office 365 portal

Running idFix is a fairly simple task. You simply run the executable and click the "Query" button to query all objects in the local directory. The "Settings" sprocket icon can be used to customize a query or use a different set of credentials. In Figure 10-20 you can see that my friend Paul's mailbox somehow has an SMTP address with a space in it. Newer versions of Exchange include checks to prevent the creation of such an address using the Exchange Management Tools. However third-party tools (such as user provisioning software) or scripts which directly modify the AD object might not include similar checks. In my case, I've simulated this scenario by using ADSIEDIT to add this invalid SMTP address to Paul's AD user object.

Using idFix, an Administrator can simply edit the object by accepting the suggested changes provided by the tool, or manually supply an alternative supported attribute value. I highly suggest reading the idFix user's guide included in the download to fully understand both the capabilities of the tool as well as the cautions that need to be taken in running it.

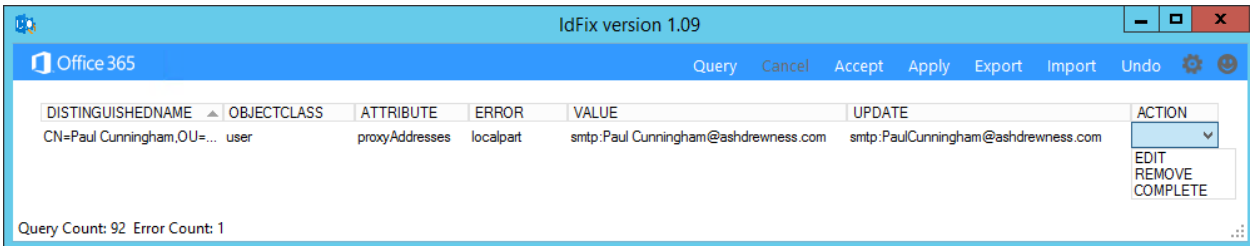


Figure 10-20: Invalid character (a space) detected in a user's *ProxyAddresses* Active Directory attribute

Once all preparations are complete, Azure AD Connect can be downloaded and configured. In this example, we will use the guide and the Express settings to configure Azure AD Connect (which will enable Password Sync and use a SQL local DB Instance) as well as selecting the option for an Exchange Hybrid deployment. The Express Settings are perfectly acceptable for a lab environment and the install should take ~10 seconds as it will not include any of the management tools SQL Express uses. As expected, four clicks and about ten minutes later, I have directory synchronization installed and am ready to begin synchronizing my local directory with Azure Active Directory. If you encounter any issues during installation, Azure AD Connect installation logs can be found in the directory: *C:\Users\<Current User>\AppData\Local\AADConnect*

Note: If the Express settings option was chosen, AADConnect will be configured to automatically upgrade itself.

The following folders are created in their default locations (to change these locations, select Custom Installation instead of Express):

- C:\Program Files\Microsoft Azure Active Directory Connect
- C:\Program Files\Microsoft Azure AD Connect Health Sync Agent
- C:\Program Files\Microsoft Azure AD Sync
- C:\Program Files\Microsoft SQL Server
- C:\Program Files\Windows Azure Active Directory

Any troubleshooting logs should be contained in the above paths. The logs are fairly straightforward to read yourself but if you need to open a support case with Microsoft they may ask you to gather the logs for analysis.

The wizard also creates several service accounts and security groups:

- AAD_GUID (User)
- AADSyncSched_GUID (User)
- MSOL_GUID (User)
- ADSyncAdmins (Group)

- ADSyncBrowse (Group)
- ADSyncOperations (Group)
- ADSYncPasswordSet (Group)

Ensure these user accounts are not deleted or their group membership modified as otherwise directory synchronization functionality will be hindered. Since Azure AD Connect with the Express installation options is a fairly basic install, uninstalling the program and reinstalling it is not a cumbersome task. However, for more advanced custom installations it may be preferred to attempt to reset permissions on these accounts based on a lab environment or Microsoft guidance. Also, any configuration customizations should be backed up beforehand.

Manual Synchronization

By default n AAD Connect 1.1.x, a synchronization should occur every 30 minutes via a scheduled workflow. In previous versions, it was performed via a Scheduled Task in Windows (Figure 10-21). It can be manually synchronized but cannot be configured for an interval less than 30 minutes.

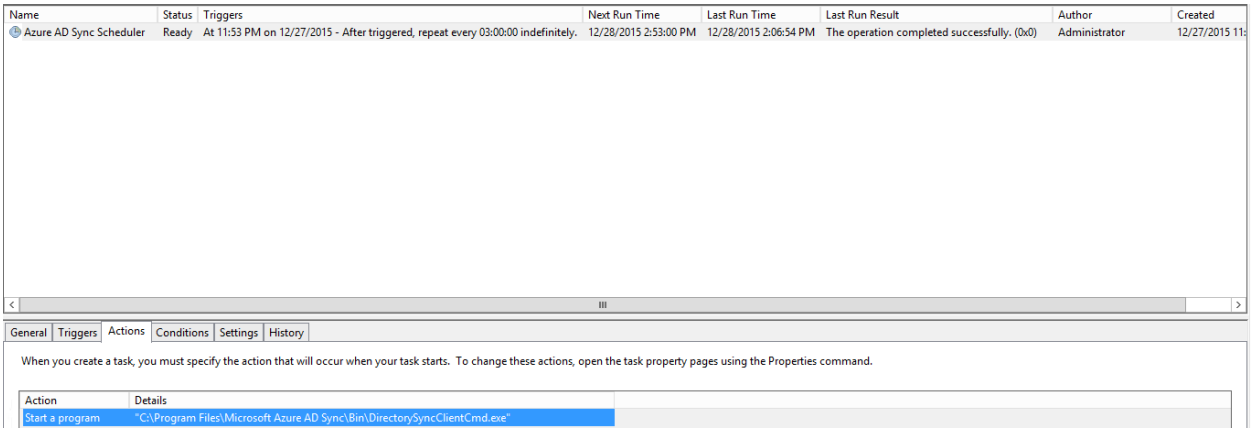


Figure 10-21: Scheduled Task for an early version of Azure AD Sync created by the Azure AD Connect installer

When Password Hash Synchronization is enabled, whenever a user changes their password on-premises, synchronization should be triggered within a two minutes. However, in early versions if you needed to manually invoke synchronization for troubleshooting purposes, you could right-click the task and select "Run".

Alternatively, you can run this executable via the command prompt: *C:\Program Files\Microsoft Azure AD Sync\Bin>DirectorySyncClientCmd.exe*

In current versions of the tool, the Scheduled Task was replaced by a custom synchronization engine workflow exposed via PowerShell. The below Cmdlets can be used to manipulate the synchronization engine:

- Get-ADSyncScheduler
- Set-ADSyncScheduler
- Start-ADSyncSyncCycle

For more information, see Microsoft MVP Jeff Guillet's post on How to Schedule and Force Sync Updates with AAD Connect 1.1.x

Monitoring Synchronization and Azure AD Connect Health Status

To view the results of the latest synchronization jobs, navigate to the synchronization GUI. By default, this program is located in the directory path:

C:\Program Files\Microsoft Azure AD Sync\UIShell\miisclient.exe

The GUI (Figure 10-22) has a fairly intuitive interface, which displays each synchronization job as a process of importing data from local Active Directory and exporting it to the remote Office 365 tenant directory.

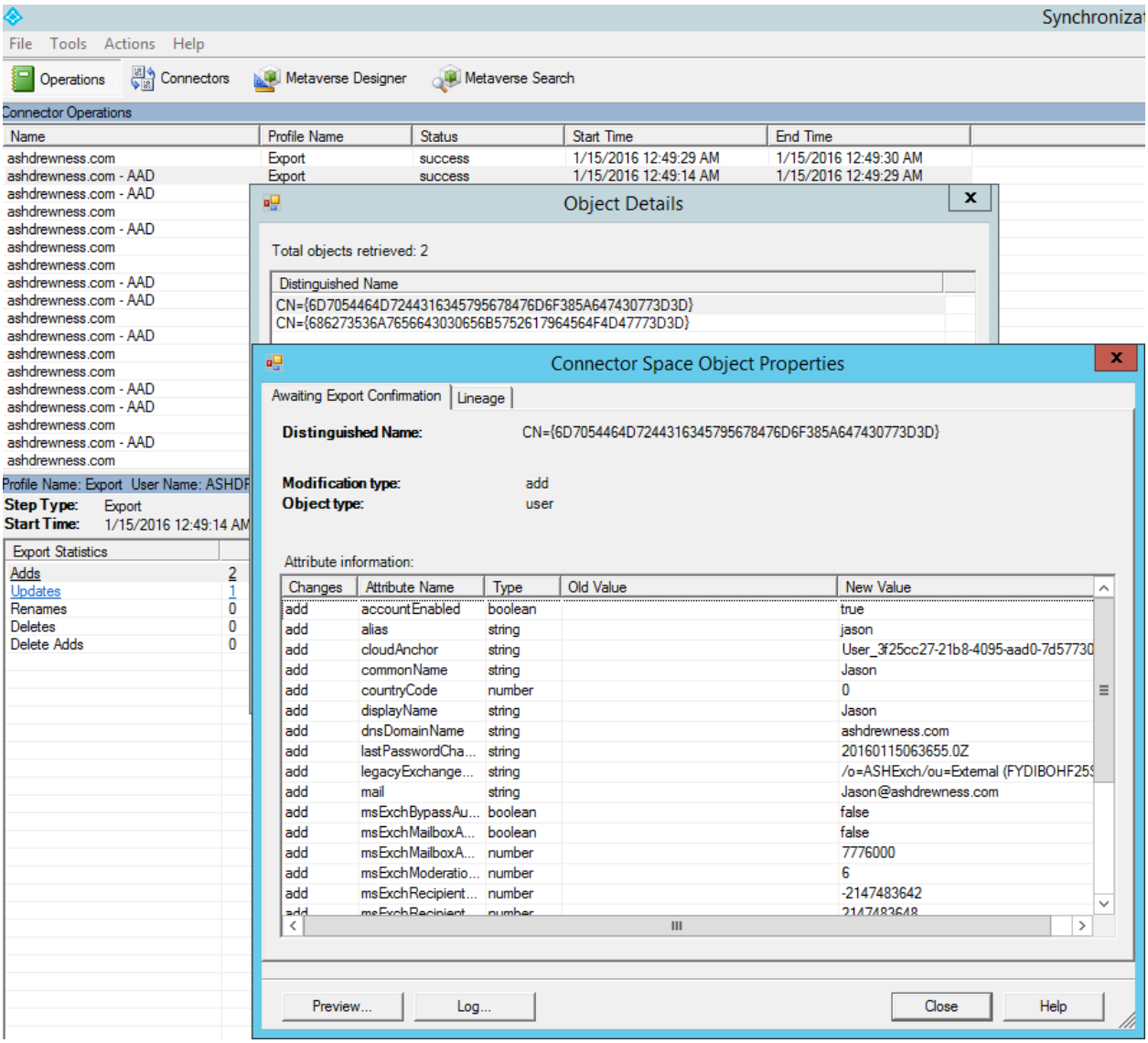


Figure 10-22: Viewing the properties of an object synchronized to Office 365

Synchronization events can be monitored using the program or using a very helpful third-party script written by Exchange MVPs Mike Crowley and Michael Van Horenbeeck.

If you want to Monitor Azure AD Connect from the Azure Portal, the Azure AD Connect Health option helps you monitor and gain insight into your on-premises identity infrastructure and the synchronization services available through Azure AD Connect.

A common issue is newly created on-premises objects not appearing in the Office 365 tenant. The above tools should allow you to view why an objects weren't synchronized. Of course the answer may simply be that synchronization has not been performed since the creation of the object. You'd be surprised how many support incidents have been created simply due to impatience. With Hybrid configurations, since the source of authority is the on-premises Active Directory via directory synchronization, objects such as mailboxes must be created on-premises and allowed to synchronize to the tenant. The two common ways to create a mailbox which will eventually live in Office 365 are:

- Create a mailbox on-premises, allow directory synchronization to complete (thus creating the corresponding mail-enabled user in the tenant), and move the mailbox to the tenant. At this point, the move process will convert the on-premises object to a mail-enabled user and the object in the tenant will change from being mail-enabled to mailbox-enabled.
- From on-premises, create a Remote Mailbox and allow directory synchronization to occur. This creates a mail-enabled user account on-premises and a mailbox in the tenant (after directory synchronization is run).

To create a remote mailbox, you can either run New-RemoteMailbox which will create a new mail-enabled user account in local Active Directory with a TargetAddress of User@Domain.Mail.Onmicrosoft.com. Once directory synchronization has occurred, the user account will show up in the tenant, and (in my experience) after a few minutes will be mailbox-enabled. Once an Exchange Online license is assigned, it'll be a fully functioning mailbox existing in Office 365. Alternatively, if you already have a user account which exists in on-premises Active Directory, you can instead use the Enable-RemoteMailbox cmdlet.

In each of these scenarios, a successful directory synchronization must occur before a mailbox can exist in the Office 365 tenant.

Matching existing objects in Azure Active Directory

Although this is more of a deployment/configuration/migration topic, the matching of on-premises objects to already created Office 365 objects is a common struggle, so the issue is worth mentioning here. Simply put, if user objects already exist in the Office 365 tenant when you first attempt a directory synchronization, DirSync/AADSync attempts to automatically match these objects. For a soft-match to occur, objects in on-premises AD and the Office 365 tenant must share the same SMTP address (Primary SMTP Address if mailbox-enabled or "mail" attribute if not mailbox-enabled). If soft-matching is not possible then a hard match using an ImmutableID might be required. For information about these scenarios, see these articles.

- How to do Hard match in Dirsync?
- PowerShell to generate ImmutableID
- How to use SMTP matching to match on-premises user accounts to Office 365 user accounts for directory synchronization

Additional References**:**

- Extending On-Premises Directories to the Cloud Made Easy with Azure Active Directory Connect
- Monitor your on-premises identity infrastructure and synchronization services in the cloud
- Deploy Office 365 Directory Synchronization (DirSync) in Microsoft Azure
- Azure AD Connect FAQ
- Azure AD Connect PowerShell cmdlets
- How to troubleshoot password synchronization when using an Azure AD Sync appliance

Note: For a deeper dive into the above topics, as well as Office 365 from the Exchange perspective, I recommend the eBook Office 365 for Exchange Professionals.

Active Directory Federation Services

Active Directory Federation Services (ADFS) was unveiled with Windows Server 2003 R2 as ADFS 1.0. Since then, ADFS has evolved to the current version (ADFS 3.0 with Windows Server 2012 R2). Although ADFS showed real promise when first released, it never became a popular solution due to its complexity and status as a relatively new technology. At one point, the Microsoft Certified Master: Active Directory program didn't

even bother including ADFS in its curriculum because so few customers used the software. The big difference came when Windows Azure and Office 365 appeared. The need for customers to synchronize user identities to Office 365 as well as a requirement to enable people to use Single Sign-On (SSO) authentication meant that ADFS became more popular. Now it is the solution of choice for SSO when accessing Microsoft cloud-hosted resources. Although the installation, configuration, and operation of ADFS is outside the goals of this chapter, it's appropriate to provide a quick overview of ADFS troubleshooting tools and techniques.

Office 365 Identity Options

When connecting your Exchange environment to Office 365, three choices exist for how your users will authenticate to cloud-hosted resources:

- Cloud identity
- Synchronized identity (with or without password synchronization)
- Synchronized identity with federation

> **Note:** Third party identity systems can be used to integrate with Office 365 in a variety of ways, whether there is also an on-premises Active Directory or not. In this chapter we'll focus on the native options provided by Microsoft.

In the Cloud Identity model (Figure 10-23), credentials in Office 365 are completely separate from on-premises Active Directory credentials. A user logs into their local domain-joined machine with their local Active Directory credentials but when connecting to their Office 365-hosted mailbox via OWA or Outlook they must enter a separate set of credentials. These credentials may have the same values but only because the user has managed them identically, not because automated synchronization is in place.

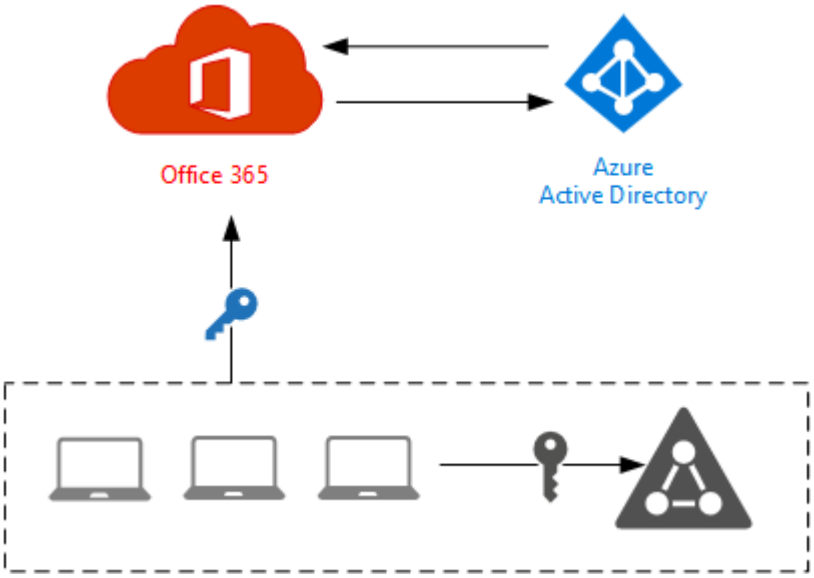


Figure 10-23: Cloud identity model

In the Synchronized identity model (Figure 10-24),with the "PasswordSync" option, Azure AD Connect instructs the Azure AD Sync components to synchronize a hash of the locally stored password hash in the on-premises Active Directory to matching objects in Azure Active Directory. This "salted hash" is non-reversible and therefore secure. This arrangement ensures the user can use their same password configured on-premises to access cloud-hosted resources. A user logs into their local domain-joined machine with their local Active Directory credentials but when connecting to their Office 365-hosted mailbox via OWA or Outlook they will

enter the same set of credentials again. This is considered Same Sign-On (SSO), not to be confused with Single Sign-On (SSO) which shares the same acronym.

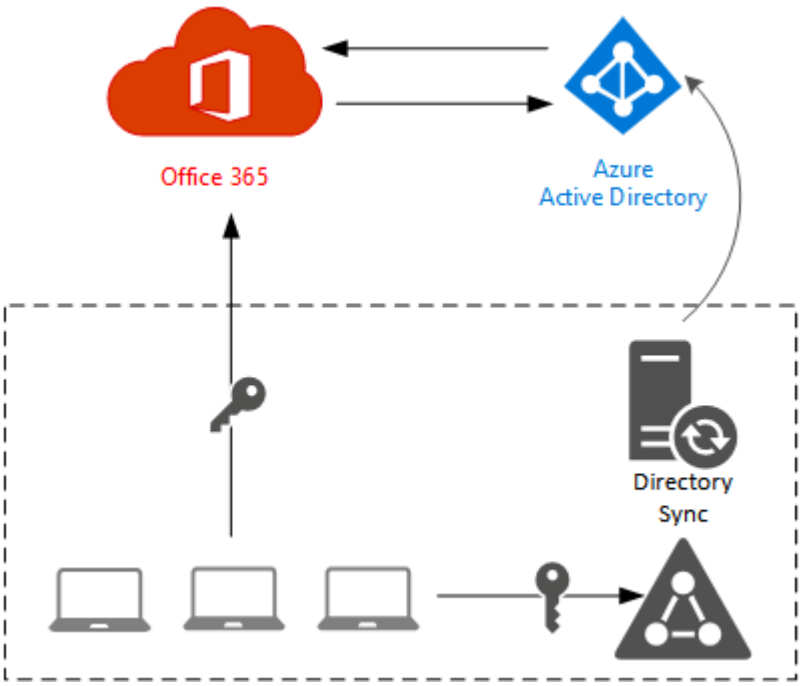


Figure 10-24: Synchronized identity model

In the Federated identity model (Figure 10-25), federation is enabled on a per-domain basis for each of the domains configured in the Office 365 tenant. After ADFS has been configured, the domains used for a Hybrid Configuration can be converted from "Managed" to "Federated." This change causes authentication requests to Office 365 to be redirected to the on-premises ADFS farm where users actually authenticate which then grants an authentication token, granting access to the requested resources. A user logs into their local domain-joined machine with their local Active Directory credentials but when connecting to their Office 365-hosted mailbox via OWA or Outlook they should connect without having to enter their credentials again. This is considered Single Sign-On (SSO). External or non-domain joined computers may still need to authenticate to ADFS.

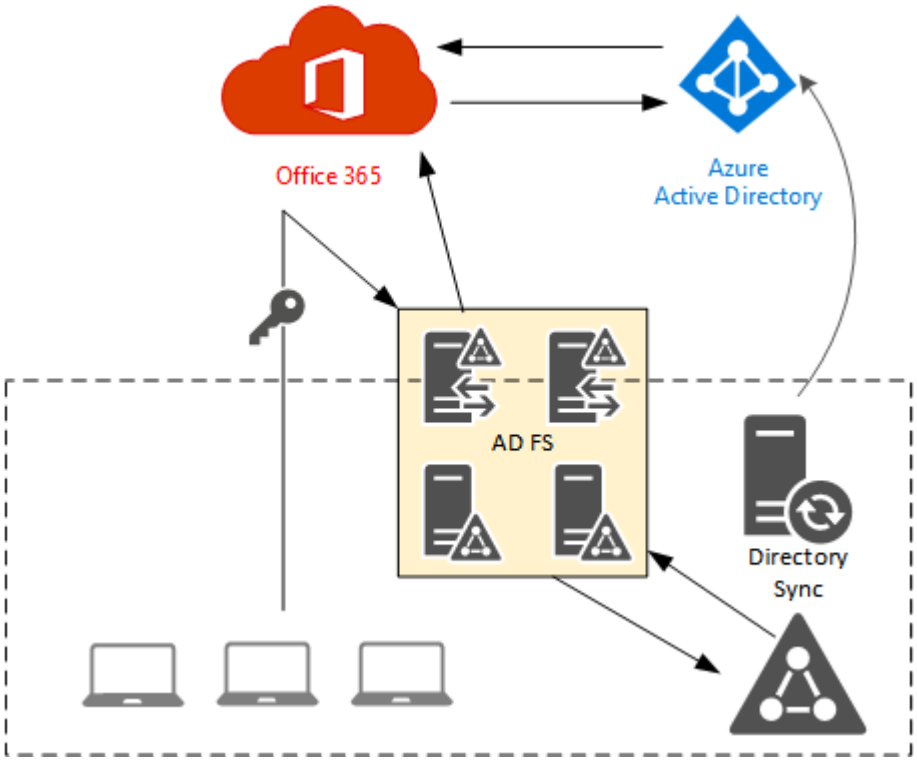


Figure 10-25: Federated identity model

Note: Depending on the client, the ADFS Federated Identity process may not necessarily be SSO. For example, Outlook still requires the password be presented. However, selecting the option to remember the password often effectively leads to a "SSO" experience. The long-term solution for Outlook will be Modern Authentication.

Troubleshooting Authentication Issues

Knowing the identity model being used in an environment is the first step to troubleshooting authentication issues. It's also important to understand that when using Federated Identities if your ADFS infrastructure is unavailable, there will be no means for your users to access Office 365 resources. This is why it's critical that redundancy and high availability are top priorities for any ADFS deployment. In addition, when troubleshooting ADFS, any action that brings down the ADFS servers, farm, namespace, or connectivity will adversely affect all Office 365 users. Therefore, outage windows should be planned accordingly. Also, if you are troubleshooting a minor issue with ADFS, take care not to escalate its severity. ADFS farms containing multiple servers should be used so changes or testing can be performed using individual servers and not the entire farm.

To view whether a domain is Managed or Federated, you use the Get-MsolDomain cmdlet. The value for "Authentication" will either be "Managed" or "Federated." As previously stated, the risk of federating a domain and using ADFS is that it becomes a single point of failure. Therefore, it is recommended to also enable Password Sync when configuring Azure AD Connect to provide a backup authentication method. Unfortunately, this is not an automatic process and you must manually switch from single sign-on to password sync. I recommend practicing this process in a lab environment or during an outage window to be prepared in the event an extended ADFS outage occurs. Of course, simply correcting the ADFS issues would be ideal as this process can potentially take up to two hours to complete.

In my experience, once ADFS is configured, there's typically little that goes wrong with the application itself. Issues typically arise as a result of problems with IIS, the service account, Group Policy, or the network infrastructure. However, here are some common troubleshooting scenarios for ADFS and ADFS Proxy Services:

- Expired SSL certificate that's assigned to the AD FS proxy server - Suggested Resolution Steps
    - Jeff Guillet's blog post for How to Update Certificates for AD FS 3.0
- Incorrect configuration of IIS authentication endpoints - Suggested Resolution Steps
- Broken trust between the AD FS proxy server and the AD FS Federation Service - Suggested Resolution Steps
- Incorrect ADFS URL – Suggested Resolution Steps
- Request Signing Certificate failing revocation – Suggested Resolution Steps
- Problem with individual ADFS/WAP Server - Suggested Resolution Steps

Additional References:

- How To Install ADFS 2012 R2 For Office 365
- Publishing Exchange to the Internet using Web Application Proxy and AD FS 3.0
- Web Application Proxy Troubleshooting
- How to troubleshoot Password Management
- DirSync: How To Switch From Single Sign-On To Password Sync
- ADFS Deep-Dive: Primer
- ADFS Deep-Dive: Troubleshooting
- How to troubleshoot AD FS endpoint connection issues when users sign in to Office 365, Intune, or Azure
- Configuring Exchange 2013 SP1 to Accept AD FS Claims
- Troubleshooting Federation, ADFS, and More

Miscellaneous Troubleshooting Tools

Microsoft Remote Connectivity Analyzer – Verify mail flow, AutoDiscover, Exchange Web Services, ActiveSync, Outlook Anywhere, and Single Sign On connectivity and functionality.

Microsoft Office 365 Deployment Readiness Tool – Determine readiness of environment prior to deploying Office 365 services

Exchange Deployment Assistant – Provides Microsoft official guidance on configuration and migration based on environmental inputs provided

The Hybrid Free Busy Troubleshooter – Troubleshoot free/busy issues between on-premises Exchange and Office 365

Automated Hybrid Troubleshooting Experience – Gathers Hybrid connectivity logs, performs analysis, and feedback for how best to resolve the issue with the Hybrid Configuration.

Mail Flow Guided Walkthrough for Office 365 – Guided walkthrough for troubleshooting mail flow issues in Office 365 and Hybrid environments.

Office 365 Support and Recovery Assistant – Identifies and fixes Outlook and Office 365 connectivity issues

Office 365 Client Performance Analyzer - Identify issues that affect network performance between your company's client PCs and Office 365

Note: When planning your data recovery strategy for a Hybrid environment, it's highly recommended to be familiar with the below Exchange Team Blog post:
Common mailbox recovery scenarios for hybrid environments

The post covers the following scenarios:

- Recover a mailbox that was deleted due to Directory Synchronization filtering changes resulting in filtering out the on-premises Active Directory user account associated with the cloud mailbox
- Recover a mailbox when the on-premises Active Directory user account associated with the cloud mailbox was accidentally or purposely deleted
- Recover a mailbox when the on-premises Active Directory user account associated with the cloud mailbox was accidentally or purposely deleted and the mailbox is on litigation hold

Additional reading

Introduction

- Exchange Hybrid Configuration
- Hybrid in Exchange 2013
- Hybrid Configuration Wizard
- Cross-Forest Mailbox Moves
- Free/Busy Sharing
- MailTips
- Online Archiving
- Secure Mail via shared SMTP namespace
- Shared Address Books (Unified GAL)
- Remote Domains
- eDiscovery Search and Compliance
- Integrated Administration (RBAC, Exchange Management Shell, EAC)
- Office 365
- Azure Active Directory
- Remote PowerShell
- Azure AD Connect (aka. Azure AD Sync/DirSync)
- Active Directory Federation Services (ADFS)
- On-Premises Active Directory Domain Services
- Microsoft Office Suite
- Information Rights Management
- Exchange Online Protection
- Office 365 Complete Guide to Managing Hybrid Exchange Deployments
- Office 365 for Exchange Professionals
- Getting Started with Hybrid Exchange Deployment
- Configuring an Exchange 2013 Hybrid Deployment and Migrating to Office 365 (Exchange Online)
- Exchange Server Deployment Assistant

The Hybrid Configuration Wizard

- Hybrid Configuration Wizard's initial release in 2011
- Office 365 Hybrid Configuration Wizard in 2015
- Office 365 release options

- Office 365 URLs and IP address ranges
- RSS Feed for Office 365 URL and IP address ranges
- AutoDiscover Domains
- Centralized Transport

Hybrid Configuration Components

- Connect to Exchange Online using remote PowerShell
- Creating a PowerShell function to connect to Exchange Online.
- Get-HybridConfiguration
- Remove-HybridConfiguration
- Organization Relationships
- Remote Domains
- Decommissioning your Exchange 2010 servers in a Hybrid Deployment
- Get-OrganizationRelationship
- Get-RemoteDomain
- New-IntraOrganizationConnector
- Using OAuth authentication to support eDiscovery in an Exchange hybrid deployment
- Configure OAuth authentication between Exchange and Exchange Online organizations
- OAUTH
- Configure the Availability service for cross-forest topologies
- Create a Federation Trust
- Get-FederationInformation
- Office 365 Exchange Hybrid Deployment (Channel 9-TechEd)
- Enabling an Exchange Certificate for Federation
- Get-FederationTrust
- Set-FederationTrust
- Keep your Federation Trust up-to-date
- Test-FederationTrust
- Get-ExchangeCertificate
- Test-OauthConnectivity
- Microsoft Remote Connectivity Analyzer
- Set-WebServicesVirtualDirectory
- Office 365 Exchange Hybrid Deployment
- Configure OAuth authentication with SharePoint 2013 and Lync 2013
- Troubleshoot a hybrid deployment
- Automated Hybrid Troubleshooting Experience
- Mail Flow Guided Walkthrough for Office 365
- Mail flow in EOP
- Configure mail flow using connectors in Office 365
- Mail flow best practices for Exchange Online and Office 365
- Troubleshoot Exchange Hybrid Mail Flow with Office 365
- Get-ReceiveConnector (On-Premises)
- Get-SendConnector (On-Premises)
- Get-InboundConnector (Office 365)
- Get-OutboundConnector (Office 365)

Directory Synchronization

- Forefront Identity Manager (FIM) 2010 R2

- Azure Active Directory Synchronization Tool (DirSync)
- Azure Active Directory Sync (AADSync)
- Azure Active Directory Connect (Azure AD Connect)
- Microsoft Identity Manager (MIM) 2016
- Integrating your on-premises identities with Azure Active Directory
- Connecting AD and Azure AD: Only 4 clicks with Azure AD Connect
- Mail-Enabled Public Folders & Directory-Based Edge Blocking
- Enabling Directory Synchronization in Office 365 Portal (Legacy Method)
- Set up directory synchronization in Office 365 (New Wizard-driven experience)
- Install and run the Office 365 IdFix tool
- Prepare directory attributes for synchronization with Office 365 by using the IdFix tool
- IDFix Download
- Azure AD Connect Download
- Install the Azure Active Directory Sync Service
- Azure Active Directory Synchronization Services: How to Install, Backup & Restore with full SQL
- How to Schedule and Force Sync Updates with AAD Connect 1.1.x
- Azure AD Sync Tool HTML Report
- Monitor your on-premises identity infrastructure and synchronization services in the cloud
- New-RemoteMailbox
- Enable-RemoteMailbox
- How to do Hard match in Dirsync?
- PowerShell to generate ImmutableID
- How to use SMTP matching to match on-premises user accounts to Office 365 user accounts for directory synchronization
- Extending On-Premises Directories to the Cloud Made Easy with Azure Active Directory Connect
- Deploy Office 365 Directory Synchronization (DirSync) in Microsoft Azure
- Azure AD Connect FAQ
- Azure AD Connect PowerShell cmdlets
- How to troubleshoot password synchronization when using an Azure AD Sync appliance

Active Directory Federation Services

- Active Directory Federation Services
- Modern Authentication
- Get-MsolDomain
- DirSync: How To Switch From Single Sign-On To Password Sync
- Group Policy Issues affecting ADFS
- How To Install ADFS 2012 R2 For Office 365
- Publishing Exchange to the Internet using Web Application Proxy and AD FS 3.0
- Web Application Proxy Troubleshooting
- How to troubleshoot Password Management
- DirSync: How To Switch From Single Sign-On To Password Sync
- ADFS Deep-Dive: Primer
- ADFS Deep-Dive: Troubleshooting
- How to troubleshoot AD FS endpoint connection issues when users sign in to Office 365, Intune, or Azure
- Configuring Exchange 2013 SP1 to Accept AD FS Claims
- Troubleshooting Federation, ADFS, and More

Chapter 11: Troubleshooting Migration

Andrew Higginbotham

Migration, by its very definition, involves change. In the field of IT, change can be the catalyst for issues to emerge, and why proper change management is so vital to an environment's health. Exchange migrations are often performed by experienced third-party Consultants, not because Exchange Administrators are necessarily incapable, but for two reasons. Planning and performing an Exchange migration can be a very time consuming endeavor, meaning Administrators my not have the necessary time while also performing their regular duties. However, primarily because there are many "gotchas" or hurdles that only an experienced specialist in migrating Exchange data will be able to efficiently navigate; or mistakes they'll know to avoid based on past experience.

This chapter covers the more common issues experienced during an Exchange migration. The topics covered will include:

- Moving Mailboxes
- Public Folder Migration and Coexistence
- Client Access Coexistence between Exchange versions
- Mail flow considerations
- Managing Exchange during coexistence

However, let us first list the various types of migrations as well as tips to ensure a successful migration experience.

Internal migrations

This is the most common type of migration, even if it's simply moving mailboxes from one database to another. However, migrating from one version of Exchange to another is what we're really referring to when discussing internal migrations. Whether the scenario is migrating from Exchange 2007 to 2013, or 2010 to 2016, or a double-hop migration from 2007 to 2010/2013 to 2016 etc., the hurdles are usually a result of getting one version of Exchange code to properly interact with another version.

Historically, the struggle for Microsoft has been engineering a newer code base which can interoperate with a legacy version. The constraints are typically that due to major architecture changes in the product, a new feature may be impossible or very time-consuming to code into a legacy version of the product. For example, when migrating from Exchange 2003 to Exchange 2010, there was no ability to proxy OWA traffic from Exchange 2010 to 2003 OWA. Instead a redirect to the client was necessary, which resulted in a sub-optimal client experience.

In another example, when Exchange 2013 was introduced into an Exchange 2007/2010 environment, access to Public Folders or Shared Mailboxes on the legacy servers did not work. Negotiate Authentication for Outlook Anywhere had to be disabled, as the legacy servers did not support it. Changing the code base of Exchange 2007/2010 to support Negotiate authentication would have been too burdensome of a task for Microsoft, therefore a change/workaround during the migration was required to allow proper functionality.

In each of these examples, a sacrifice had to be made for simplicity sake and the individual performing the migration must be aware of these "gotchas" to ensure a smooth project. In this chapter we'll further discuss such constraints which would otherwise result in failed connectivity and a possible call to Microsoft Support.

Note: One form of migration troubleshooting not covered in this chapter is migrating from a third-party (non-Exchange) email system. These often involve the use of third-party utilities/services and troubleshooting would involve contacting their Support organization.

Cross-Forest

Mergers, acquisitions, divestitures, and temporary business partnerships often involve a cross-forest migration or at least a period of coexistence across Exchange organizations. Many Consultants specialize in these scenarios as they often involve many moving parts outside the normal expertise of an Exchange Administrator. Technologies often involved in such migrations include but are not limited to:

- Certificates trusted by both organizations to ensure secure mail flow as well as trusted HTTPS endpoints for accessing resources such as calendaring Free/Busy information
- Directory Synchronization tool to provide synchronization of objects between organizations enabling querying of Calendaring Free/Busy information, cross-forest AutoDiscover redirection, Remote Mailbox Moves, Linked Mailboxes, and cross-forest mail flow
- Connectors for secure cross-forest mail flow, which are dependent upon mutually trusted certificates for proper TLS communications
- (Optionally) Third-party tools used to simplify the migration experience, provide additional features, and potentially increase the speed of the migration

Issues surrounding cross-forest migrations typically involve one of the following areas:

- Moving mailboxes across forest
- Accessing Calendaring Free/Busy information across organizations
- Shared Mailbox and Delegate Access across forest

These scenarios, and common issues encountered, will be covered later in this chapter.

Note: Cross-Forest mailbox moves typically require the Mailbox Replication Service Proxy (MRSProxy) to be enabled. The MRSProxy facilitates cross-forest mailbox moves and remote move migrations between multiple on-premises Exchange organizations or between an on-premises Exchange organization and Exchange Online. The MRSProxy is a subcomponent of the Exchange Web Services Virtual Directory and can be enabled via EAC or EMS. For Hybrid migrations, the Hybrid Configuration Wizard enables the MRSProxy during its execution.

Hybrid

An Exchange Hybrid Configuration, is essentially a specialized cross-forest coexistence state. An on-premises Exchange organization is made to coexist with Exchange Online in an Office 365 tenant. Secure mail flow, cross-forest free/busy, and cross-forest mailbox moves are all features shared with a non-hybrid cross-forest migration. Key differences are:

- Hybrid has a dedicated Hybrid Configuration Wizard which automates this complex configuration with Office 365 and Exchange Online
- Hybrid relies on Federation and OAuth for authorization whereas cross-forest scenarios often involve a forest trust

- Hybrid configurations offer full Microsoft support for features such as Online Archives and Hybrid eDiscovery

See the Hybrid chapter for details regarding troubleshooting Hybrid configurations.

Note: When choosing to use Shared Mailboxes in a Hybrid Configuration, do not convert a regular mailbox which has been synched to Office 365 into a Shared Mailbox. Instead, make the mailbox Shared while it exists on-premises and then synchronize it to Office 365.

Note: Just as permission inheritance being disabled in on-premises Exchange can cause various issues, it can also cause Hybrid Mailbox Moves to fail:
A user can't access a mailbox by using Outlook after a remote mailbox move from an on-premises Exchange Server environment to Office 365

Other Migration Options

- Cutover Migration
    - Migrating environments with less than 2,000 mailboxes to Exchange Online in one singular action
    - Utilizes Outlook Anywhere on-premises to pull Exchange data to Office 365
    - Requires Outlook profiles to be recreated
    - Cannot use Directory Synchronization during migration
    - Migrations to Exchange Online Only. Cannot migrate back on-premises.
- Staged Migration
    - Migrating a subset of mailboxes for an environment to Exchange Online
    - Utilizes Outlook Anywhere on-premises to pull Exchange data to Office 365
    - Does not require Outlook profiles to be recreated
    - Not supported with Exchange 2010/2013/2016
    - Requires Directory Synchronization before/during migration
    - Migrations to Exchange Online Only. Cannot migrate back on-premises.
- IMAP Migration
    - Commonly used when migrating from a third-party messaging system
    - Migrations to Exchange Online Only. Cannot migrate back on-premises.
- Exchange Remote Move (MRSProxy)
    - Similar to a cross-forest mailbox move using the MRSProxy component
    - Not currently fully supported as a means to migrate to Exchange Online unless in a Hybrid Configuration
- Third-Party Migration Tool
    - Typically utilizing EWS, MAPI, IMAP, or custom method to move Exchange data

Moving Mailboxes

There are various different reasons for moving mailboxes in Exchange environments. Some as part of a migration effort, and others for administrative or troubleshooting purposes. TechNet lists some common reasons why mailboxes need to be moved within an Exchange organization.

- **Transition**: When the time comes to upgrade to a new version of Exchange or to introduce a new mailbox server into the organization, the only available method to transfer workload is to move mailboxes from the existing Exchange servers.
- **Realignment**: Over time, it's likely that the workload assigned to mailbox servers becomes unbalanced due to user activity or mailbox growth. Moving mailboxes between databases or between servers allows workload to be redistributed on a more equitable basis. The Exchange Preferred Architecture and Server Role Requirements Calculator also dictate a balanced database layout, so if you follow their recommendations, you'll probably have to move some mailboxes around to rebalance workload.
- **Investigating an issue**: Moving a mailbox can help resolve issues. For instance, moving some mailboxes from one server to another can reduce the workload on a server and improve service to end users.
- **Corrupted mailboxes**: Mailbox corruption can be resolved by moving the affected mailboxes to a different database or server. This is because the mailbox move process drops any corrupted items that it finds. Moving mailboxes also rebuilds content indexes and can help to resolve search issues.
- **Physical location changes**: It is common for organizations to group mailboxes as close as possible in physical and network terms to user locations. If people move within the company, you might have to move their mailboxes to a server that's closer to their new location.
- **Separation of administrative roles**: You may want to separate Exchange administration from Windows operating system account administration. In this scenario, the Exchange servers will be located in a separate resource forest that is controlled by the Exchange administrators. The Windows accounts associated with the mailboxes reside in a separate forest. You might have to move mailboxes from one forest to another to enable the separation. A similar situation might occur if you outsource the administration of Exchange to a third party and have to move mailboxes into a forest that the third party manages.
- **Integrating e-mail and user account administration**: To reduce the overall complexity of the Active Directory operational model in use, you might decide to remove resource forests and move their resources back into the account forest. This work also requires mailbox moves, in this case out of the resource forest and into the account forest.

Having spoken with many Exchange administrators, the most common reason for moving mailboxes are:

- To facilitate system housekeeping. For example, to move all mailboxes from a server to allow the server software to be upgraded.
- To move mailboxes so that they are geographically closer to their owners.
- To implement an evenly distributed environment (per the sizing calculator's recommendations).
- As a means to balance mailbox capacity and load.

I'd add resolving mailbox corruption to the list as this is a reason that I often encounter, probably because of my position in a support organization.

In Microsoft's eyes, the practice of placing mailboxes in databases based on department or hierarchy within the company (we all know the joke about keeping everyone who can fire you on the same, closely monitored and regularly backed up database) should be discouraged. Most Exchange experts would also agree, as it makes planning availability much more difficult. If a single Exchange server fails, there's no easy method to ensure the remaining servers handle a balanced user load upon failover.

Consider the following example:

- 5,000 mailbox environment
- 2,000 high-usage mailboxes which send/receive 150 messages per day and have 5 GB mailboxes

- 3,000 low-usage mailboxes which send/receive 50 messages per day and have 512 MB mailboxes

In such an environment, if each mailbox database contains 100 mailboxes, 40 would be high-usage mailboxes while 60 would be low-usage mailboxes. This would ensure a balanced performance and storage capacity load across each mailbox database in the environment. This would result in even distribution if a database failover occurs, consistent database backup/restore times, as well as consistent performance and capacity load on each volume.

Of course, this balance doesn't come at an administrative cost. Monitoring of the environment to ensure mailboxes are still generating the expected load, and then rebalancing the environment as needed by moving mailboxes to different databases will be required. Of course, the frequency at which this will be required depends entirely on the environment's rate of change. It has been said that Microsoft is constantly moving mailboxes in Office 365 to balance workload across available servers. However, this cannot be taken as normal operating practice because Exchange Online is a massive environment that is very different to any other.

If you're working to achieve this mailbox distribution balance, it's import to understand how to move mailboxes without affecting user work, to throttle the moves as needed so performance is not impacted, and monitor/manage the moves as efficiently as possible.

Note: Although Microsoft promotes the Preferred Architecture for all on-premises Exchange deployments, not all environments will be deployed in such a manner. Smaller environments will have a hard time justifying the hardware/licensing expenses that allow for three copies of every database and site resiliency. Many environments will have one to three Exchange servers and will not have evenly distributed databases. However, I still recommend the practice as it allows for easier growth planning and balanced performance. Even in single server environments, having multiple balanced databases will allow flexibility when timing backup/restore jobs.

Exchange 2010 introduced asynchronous (behind the scenes) mailbox moves in the form of Move Requests. The new approach delivers some significant advantages, including:

- Mailbox moves are asynchronous and are performed by the Microsoft Exchange Mailbox Replication service (MRS). A request is made to MRS to move a mailbox that is processed as system load and resources allow. Any instance of the MRS in the Active Directory Site can process a move request. Servers/services can also be restarted while the move is in process and the move request will automatically resume once services again become available
- Mailboxes remain online during the asynchronous moves. Users can still access their mailbox data while their mailbox is being moved. Behind the scenes, MRS copies the mailbox in the destination database. Once the copy is complete, the "switch is flipped" (of pointers in Active Directory) to activate the mailbox copy in the the destination database. The final redirection to the new mailbox can be controlled by the administrator, as move requests can be configured to suspend and await approval after copying is complete.
- The items in a mailbox's Recoverable Items folder are moved with the mailbox. Therefore, deleted or purged items remain with the mailbox when it is moved. This is vital in compliance scenarios where all mail items must be retained in the mailbox.
- As soon as MRS begins to copy data to the target mailbox, the indexing service on that server starts to index it so that fast searches are available to users immediately after the mailbox move completes.
- You can configure throttling for each MRS instance, each mailbox database, or each Mailbox server. Throttling behavior can be controlled by editing the *MSExchangeMailboxReplication.exe.config* file that exists on every Exchange mailbox servers

- Remote mailbox moves work over the Internet by way of the Microsoft Exchange Mailbox Replication Proxy (MRSProxy) service. There's no need to configure direct Active Directory access between the forests as this web service handles all authentication and mailbox moves over HTTPS.
- Mailbox moves can be managed from any Exchange server within the organization
- Mailbox content is moved through the MRS without any need to move through another computer.
- The mailbox's move history (last five move operations) is maintained in the mailbox

Initiating Mailbox Moves

Whatever your reason for moving a mailbox, be it migration or troubleshooting, initiating a move request is fairly simple. In its simplest form, a move request cmdlet only requires the mailbox which is to be moved and the destination mailbox database. For example:

```
[PS] C:> New-MoveRequest –Identity <MailboxName> -TargetDatabase <TargetDatabaseName>
```

If a group of mailboxes, such as all mailboxes on a given database, need to be moved at once, then a Batch Move Request can be created:

```
[PS] C:> Get-Mailbox -Database <SourceDatabaseName> | New-MoveRequest -TargetDatabase
<TargetDatabaseName> -BatchName "DB01toDB02" –SuspendWhenReadyToComplete
```

This should not be confused with a Migration Batch in Exchange 2013/2016. The Migration Batch feature introduced in Exchange 2013 will be discussed later in this module. The "BatchName" parameter is only used to assign an identifier to a group of mailboxes to be moved and allows you to query them using the Get-MoveRequest cmdlet while specifying the BatchName parameter:

```
[PS] C:> Get-MoveRequest –BatchName "DB01toDB02"
```

If you want to have the move request suspend at 95% completion, the "SuspendWhenReadyToComplete" parameter can be used. This allows the majority of the mailbox data to be moved in the background while the mailboxes are still online. Then when ready, an Administrator can complete the moves, which will involve a brief outage for the mailbox. As there are many reasons to move mailboxes as a means of troubleshooting, I highly recommend performing mailbox moves in a way which will have as little impact to production as possible. Therefore, the SuspendWhenReadyToComplete parameter should be used to move the mailbox data, suspend the move request before completion, and allow you to resume the request using the Resume-MoveRequest cmdlet. The above example using BatchName "DB01toDB02" will automatically suspend at 95% completion. To complete the automatically suspended batch of move requests, we would use the below command:

```
[PS] C:> Get-MoveRequest –BatchName "DB01toDB02" | Resume-MoveRequest
```

Note: Timing mailbox moves is vital to maintaining availability in an Exchange environment. While the moves are an online process in Exchange 2007 SP3 and newer, as the move completes there is still a short window where the users will be unable to access their mailbox. Even if this is only a few seconds, the possibility of generating help desk calls still exists, especially if the move generates an Outlook pop-up regarding connectivity loss. It's best to notify users about this potential outage before it happens to mitigate helpdesk calls.

Monitoring and Managing Mailbox Moves

The two primary cmdlets for monitoring Move Requests are Get-MoveRequest and Get-MoveRequestStatistics, and they're certainly simple enough to utilize after a few minutes of reading their TechNet pages. The first cmdlet, with no parameters or filters, simply lists all Move Requests in the organization; whether they are in progress, completed, failed, or suspended. The second cmdlet pulls data from the Mailbox Replication Service as well as the system mailbox on the destination database which houses temporary mailbox move data. Allow me to list a few parameters I've found useful while monitoring or troubleshooting mailbox moves:

```
[PS] C:> Get-MoveRequest | Get-MoveRequestStatistics
```

This is a simple cmdlet which will provide the information contained in Figure 11-1.

```
[PS] C:\>Get-MoveRequest | Get-MoveRequestStatistics

DisplayName                StatusDetail               TotalMailboxSize           TotalArchiveSize           PercentComplete
-----------                ------------               ----------------           ----------------           ---------------
Administrator              Completion                 8.464 MB (8,875,406 by...                             95
```

Figure 11-1: Simple view of active move requests and useful data

When you simply wish to know which mailboxes are being moved, their size, and % completed, this is the simplest way to gather that information. However, should you want further details, including a report of any failures encountered during the move, I recommend the below command:

```
[PS] C:> Get-MoveRequest | Get-MoveRequestStatistics -IncludeReport | Format-List
DisplayName,Status,StatusDetail,SourceDatabase,TargetDatabase,TotalMailboxSize,BytesTransferred,
PercentComplete,BadItemLimit,BadItemsEncountered,Report
```

I've found this particularly useful when trying to diagnose a failed move request. The below output in Figure 11-2 is simply an example of the command's output and displays the data captured. Should bad (corrupted) items be encountered, they will be displayed here. It's not uncommon for a mailbox (in particular, large mailboxes) to have corrupted items which cannot be copied. When creating a move request you can specify the maximum acceptable bad items using the *BadItemLimit* parameter. If this limit is exceeded, the move will fail. One must decide the amount of data loss which is tolerable, but I will add that if an item is corrupted in the mailbox you'll unlikely be able to open it with Outlook anyways.

Note: If a Move Request's *BadItemLimit* parameter is configured higher than 50, the AcceptLargeDataLoss parameter must be provided.

```
[PS] C:\>Get-MoveRequest | Get-MoveRequestStatistics -IncludeReport | Format-List DisplayName,Status,StatusDetail,Source
Database,TargetDatabase,TotalMailboxSize,BytesTransferred,PercentComplete,BadItemLimit,BadItemsEncountered,Report


DisplayName          : Administrator
Status               : InProgress
StatusDetail         : CopyingMessages
SourceDatabase       : DB1
TargetDatabase       : DB4
TotalMailboxSize     : 8.511 MB (8,924,736 bytes)
BytesTransferred     : 8.626 MB (9,044,999 bytes)
PercentComplete      : 95
BadItemLimit         : 0
BadItemsEncountered : 0
Report               : 1/19/2016 11:50:03 PM [ASH-EX4] 'ASH.NET/Company_Users/Admin_Users/Administrator' created move
                       request.
                       1/19/2016 11:50:09 PM [ASH-EX4] The Microsoft Exchange Mailbox Replication service
                       'ASH-EX4.ASH.NET' (15.1.225.37 caps:7FFF) is examining the request.
                       1/19/2016 11:50:09 PM [ASH-EX4] Connected to target mailbox
                       'ab34dbe9-d2f7-43ac-afa7-a6d1c5218164 (Primary)', database 'DB4', Mailbox server
                       'ASH-EX4.ASH.NET' Version 15.1 (Build 225.0).
                       1/19/2016 11:50:10 PM [ASH-EX4] Connected to source mailbox
                       'ab34dbe9-d2f7-43ac-afa7-a6d1c5218164 (Primary)', database 'DB1', Mailbox server
                       'ASH-EX1.ASH.NET' Version 15.0 (Build 1130.0), proxy server 'ASH-EX1.ASH.NET' 15.0.1130.5
                       caps:1F7FFFFFCB07FFFF.
                       1/19/2016 11:50:10 PM [ASH-EX4] Request processing started.
                       1/19/2016 11:50:10 PM [ASH-EX4] Source mailbox information:
                       Regular Items: 23, 8.458 MB (8,868,387 bytes)
                       Regular Deleted Items: 0, 5.559 KB (5,692 bytes)
                       FAI Items: 33, 43.91 KB (44,965 bytes)
                       FAI Deleted Items: 1, 5.559 KB (5,692 bytes)
                       1/19/2016 11:50:10 PM [ASH-EX4] Cleared sync state for request
                       ab34dbe9-d2f7-43ac-afa7-a6d1c5218164 due to 'CleanupOrphanedMailbox'.
                       1/19/2016 11:50:10 PM [ASH-EX4] Stage: CreatingFolderHierarchy. Percent complete: 10.
                       1/19/2016 11:50:10 PM [ASH-EX4] Initializing folder hierarchy from mailbox
                       'ab34dbe9-d2f7-43ac-afa7-a6d1c5218164 (Primary)': 75 folders total.
                       1/19/2016 11:50:10 PM [ASH-EX4] Folder creation progress: 0 folders created in mailbox
                       'ab34dbe9-d2f7-43ac-afa7-a6d1c5218164 (Primary)'.
                       1/19/2016 11:50:12 PM [ASH-EX4] Folder hierarchy initialized for mailbox
                       'ab34dbe9-d2f7-43ac-afa7-a6d1c5218164 (Primary)': 74 folders created.
                       1/19/2016 11:50:12 PM [ASH-EX4] Stage: CreatingInitialSyncCheckpoint. Percent complete: 15.
                       1/19/2016 11:50:13 PM [ASH-EX4] Initial sync checkpoint progress: 0/75 folders processed.
                       Currently processing mailbox 'ab34dbe9-d2f7-43ac-afa7-a6d1c5218164 (Primary)'.
                       1/19/2016 11:50:14 PM [ASH-EX4] Initial sync checkpoint completed: 65 folders processed.
                       1/19/2016 11:50:14 PM [ASH-EX4] Stage: LoadingMessages. Percent complete: 20.
                       1/19/2016 11:50:15 PM [ASH-EX4] Messages have been enumerated successfully. 56 items loaded.
                       Total size: 8.462 MB (8,873,236 bytes).
                       1/19/2016 11:50:15 PM [ASH-EX4] Stage: CopyingMessages. Percent complete: 25.
                       1/19/2016 11:50:15 PM [ASH-EX4] Copy progress: 0/56 messages, 0 B (0 bytes)/8.462 MB (8,873,236
                       bytes), 54/75 folders completed.
                       1/19/2016 11:50:17 PM [ASH-EX4] Copying messages is complete. Copying rules and security
                       descriptors.
                       1/19/2016 11:50:19 PM [ASH-EX4] Initial seeding completed, 56 items copied, total size 8.462 MB
                       (8,873,236 bytes).
                       1/19/2016 11:50:19 PM [ASH-EX4] Stage: IncrementalSync. Percent complete: 95.
```

Figure 11-2: Move request statistics and a report detailing the phases of the move

When providing the IncludeReport parameter to the Get-MoveRequestStatistics cmdlet, you're provided with a detailed log of the actions performed during the move. If a move request fails for any reason, the report will help in the diagnosis. Common reasons for a move request failing are:

- Move exceeded BadItemLimit
- Move exceeded LargeItemLimit
- Source or Destination Mailbox Database dismounted
- System (hidden) mailbox used for processing inbound move requests on destination database is inaccessible (See Mailbox Move System Mailboxes module)
- Connectivity failure between source database, server processing the move request, and the destination database
- Mailbox in quarantined status (See Overcoming Mailbox Corruption module)
- Repeated store crashes

The *BadItemLimit* parameter can also be modified on an existing Move Request using the Set-MoveRequest cmdlet. Other actions which can be performed against an existing request are suspending it using Suspend-MoveRequest, resuming it using Resume-MoveRequest, and Remove-MoveRequest. While a Move Request can be manually suspended and resumed using the above aptly named cmdlets, it's more common to create Move Requests which will suspend before completion; allowing you to manually resume them when ready (Figure 11-3).

```
[PS] C:\>New-MoveRequest Administrator -TargetDatabase db4 -SuspendWhenReadyToComplete

DisplayName                StatusDetail               TotalMailboxSize           TotalArchiveSize           PercentComplete
-----------                ------------               ----------------           ----------------           ---------------
Administrator              Queued                     8.52 MB (8,934,141 bytes)                             0

[PS] C:\>Get-MoveRequest | Get-MoveRequestStatistics

DisplayName                StatusDetail               TotalMailboxSize           TotalArchiveSize           PercentComplete
-----------                ------------               ----------------           ----------------           ---------------
Administrator              AutoSuspended              8.52 MB (8,934,141 bytes)                             95

[PS] C:\>Resume-MoveRequest administrator
[PS] C:\>Get-MoveRequest | Get-MoveRequestStatistics

DisplayName                StatusDetail               TotalMailboxSize           TotalArchiveSize           PercentComplete
-----------                ------------               ----------------           ----------------           ---------------
Administrator              Completed                  8.52 MB (8,934,141 bytes)                             100
```

Figure 11-3: A Move Request resumed after being configured to automatically suspend

When a Move Request has completed it should be removed, as the mailbox cannot be moved again until you do so. When Exchange 2010 first released, it was a very common support call to have customers be unable to move a mailbox because a previous request had not been removed.

Migration Batches

Migration Batches are not necessarily a new vehicle for moving mailboxes, but more akin to a new paint job and improved handling. Migration Batches allow superior manageability of the mass movement of mailboxes both inter and intra-forest. They allow an administrator to manage large amounts of mailbox moves, submitted manually or via CSV file, with new features such as:

- Incremental synchronizations - This feature enables a sync every 24 hours to keep source and destination mailboxes updated
- Automatic retry of moves
- Automatic cleanup of Move Requests upon batch removal - Using the *–Force* parameter of the *Remove-MigrationBatch* cmdlet will not remove the move requests within the Migration Batch. The parameter is typically used to remove corrupted batches.
- Move Report generation – Email reports can be sent to a specified address.
- Ability to select multiple target databases for even distribution.
- Support for Migration Endpoints - Used in IMAP, Cutover, Staged, Hybrid, and Cross-Forest migrations
- Not required for intra-org migrations

Below are the various cmdlets used to create, manage, and remove Migration Batches, along with explanations and use cases for the non-intuitive cmdlets:

- Get-MigrationBatch
- New-MigrationBatch
- Set-MigrationBatch - Modify parameters such as BadItemLimit, LargeItemLimit, etc. of an already created Migration Batch
- Start-MigrationBatch - Begin a Migration Batch which was stopped or not created to automatically start
- Stop-MigrationBatch
- Complete-MigrationBatch - Finalize a Migration Batch which has successfully completed initial synchronization
- Remove-MigrationBatch
- Get-MigrationUser - View details of a user which is a part of a Migration Batch
- Remove-MigrationUser - Remove a user from a Migration Batch
- Get-MigrationUserStatistics - Similar to Get-MoveRequestStatistics but with less details of the MRS process itself

- New-MigrationEndpoint - Define a remote endpoint (for on-premises environments, the EWS URL of a server in another Exchange Organization). If migrating from on-premises to Office 365, this endpoint is defined in the Office 365 tenant and resolves to the on-premises environment
- Set-MigrationEndpoint
- Remove-MigrationEndpoint
- Get-MigrationConfig
- Set-MigrationConfig - Configure Organization-wide settings for Migration Batches, such as MaxConcurrentMigrations and MaxNumberOfBatches
- Get-MigrationStatistics - Display high-level statistics such as TotalCount, ActiveCount, StoppedCount, FailedCount, etc.
- Test-MigrationServerAvailability - Test Migration Endpoints for various migration types (Outlook Anywhere, IMAP, MRSProxy, etc.). Can be used on-premises and in the Office 365 tenant depending on parameters used. For example, you cannot test Outlook Anywhere Migration endpoint capability, as such an endpoint can only be used when migrating from on-premises to Office 365 via a Cutover or Staged migration.

Data for Migration Batches are stored within an arbitration mailbox called "Migration mailbox. (*Migration.8f3e7716-2011-43e4-96b1-aba62d229136*)" which can be viewed using *Get-Mailbox –Arbitration*. If this mailbox is removed, inaccessible, or past its storage limit, Migration Batches will fail. If this mailbox does not exist, it needs to be recreated.

Start the Active Directory Users and Computers snap-in. Click **Users**, and look at the accounts, or perform a search, to verify that an account named " Migration.8f3e7716-2011-43e4-96b1-aba62d229136" does not exist.

If this account exists in the Users container, skip straight to the step for running *Enable-Mailbox*.

To recreate the mailbox, first run the following command using the Exchange setup files.

```
C:>  Setup.exe /preparead/IAcceptExchangeServerLicenseTerms
```

Next, in the Exchange Management Shell run the following commands.

```
[PS] C:> Enable-Mailbox -Arbitration -Identity "Migration.8f3e7716-2011-43e4-96b1-aba62d229136"

[PS] C:> Set-Mailbox "Migration.8f3e7716-2011-43e4-96b1-aba62d229136" -Arbitration –Management:$true
```

Note: It is expected to have excessive transaction log generation on the mailbox database which hosts the Migration mailbox when using Migration Batches. This is explained by Microsoft in the below Knowledge Base article:

Large transaction logs are generated when you move mailboxes in Exchange Server 2013 or Exchange Server 2016 Administration Center

Microsoft recommends two workarounds for this behavior. Either enable Circular Logging on the database hosting the Migration Mailbox or do not use Migration Batches; instead using New-MoveRequest in Exchange Management Shell to move mailboxes.

Mailbox Move System Mailboxes

Each Mailbox Database has a system mailbox which is used to process mailbox move data. If this mailbox is unavailable, exceeded its storage limit, or corrupted, mailbox moves to that database will fail. Executing Get-MoveRequestStatistics for moves to that database may also generate an error. To view this system mailbox run the below command:

```
[PS] C:> Get-MailboxStatistics -Database <DatabaseName> | Where {$_.displayname -like
'*SystemMailbox*'} | Format-List
```

Obviously, the mailbox should be visible, but also verify the value for StorageLimitStatus is blank. If the value is "MailboxDisabled" then it has exceeded its storage limits and moves will fail. This can occur when a large amount of mailbox moves have been created to this database. The simplest solution is simply to increase the storage quota for this mailbox.

If this system mailbox is missing, options are fairly limited. While other system/arbitration mailboxes can be easily recreated (see below references), I'm unaware of a supported method to recreate this mailbox. In my experience, the only successful remediation step is to move all mailboxes from the database and delete it. Remember, the system mailbox on the destination database is what is used to process move requests. So even though the system mailbox is missing from DatabaseA, it will not prevent you from moving mailboxes from DatabaseA to DatabaseB.

Additional References:

- System Attendant Mailbox and Exchange 2013
- Exchange – Issues if system mailboxes are not correctly configured
- Exchange 2013 recreate arbitration mailboxes

Mailbox Move History

If you need to determine which past databases have housed a particular mailbox (in a mailbox restore scenario for example), mailbox move history is maintained within the mailbox itself. This data can be retrieved using the following command:

```
[PS] C:> (Get-MailboxStatistics <MailboxName> -IncludeMoveHistory).MoveHistory | Select
CompletionTimestamp,SourceDatabase,TargetDatabase | Format-Table –Auto
```

Exchange 2010 will keep the previous 2 moves, whereas 2013/2016 will keep move history for the previous 5 moves by default. This value can be changed to as high as 100 by modifying the MaxMoveHistoryLength value of the "Program Files\Microsoft\Exchange Server\V15\Bin\MsExchangeMailboxReplication.exe.config" file.

See Paul Cunningham's Retrieving the Move History for an Exchange Server 2010/2013 Mailbox blog post for more details on viewing move history.

Mailboxes Move Throttling and Performance

The Mailbox Replication Service is governed/throttled by the settings contained within the "%ExchangeInstallPath%\Bin\MsExchangeMailboxReplication.exe.config" configuration file. Figure 11-4 lists just some of the configurable settings in the MRS configuration file on an Exchange 2016 Server:

```
<!-- Mailbox Replication Service configuration

  Setting Name - Default, MinValue, MaxValue

  MaxRetries - 60, 0, 1000
  MaxCleanupRetries - 480, 0, 600
  RetryDelay - 00:00:30, 00:00:10, 00:30:00
  MaxMoveHistoryLength - 5, 0, 100
  MaxActiveMovesPerSourceMDB - 20, 0, 100
  MaxActiveMovesPerTargetMDB - 20, 0, 100
  MaxActiveMovesPerSourceServer - 100, 0, 1000
  MaxActiveMovesPerTargetServer - 100, 0, 1000
  MaxActiveJobsPerSourceMailbox - 5, 0, 100
  MaxActiveJobsPerTargetMailbox - 2, 0, 100
  MaxTotalRequestsPerMRS - 100, 0, 1024
  FullScanMoveJobsPollingPeriod - 00:15:00, 00:03:00, 1.00:00:00
  FullScanLightJobsPollingPeriod - 00:15:00, 00:00:30, 1.00:00:00
  ADInconsistencyCleanUpPeriod - 1.00:00:00, 00:00:00, 10.00:00:00
  WLMResourceStatsLoggingPeriod - 00:30:00, 00:00:00, 10.00:00:00
  WLMResourceStatsLogEnabled = false, false, true
  WLMResourceStatsLogMaxDirSize = 50000000, 0, 1048576000 (bytes)
  WLMResourceStatsLogMaxFileSize = 500000, 0, 10485760 (bytes)
  RequestStateLoggingPeriod - 02:00:00, 00:00:30, 10.00:00:00
  RequestStateLogEnabled - true, false, true
  RequestStateLogMaxDirSize - 50000000, 0, 1048576000 (bytes)
  RequestStateLogMaxFileSize - 500000, 0, 10485760 (bytes)
  MRSSettingsLoggingPeriod - 08:00:00, 00:00:00, 10.00:00:00
  MRSSettingsLogEnabled - false, false, true
  MRSSettingsLogMaxDirSize - 50000000, 0, 1048576000 (bytes)
  MRSSettingsLogMaxFileSize - 500000, 0, 10485760 (bytes)
  MRSScheduledLogsCheckFrequency - 00:05:00, 00:00:00, 10.00:00:00
  MRSSettingsLogList - "MaxTotalRequestsPerMRS;MaxActiveJobsPerSourceMailbox;MaxActiveJobsPerTargetMailbox", null, null
  HeavyJobPickupPeriod - 00:00:05, 00:00:00, 10.00:00:00
  LightJobPickupPeriod - 00:00:10, 00:00:00, 10.00:00:00
  MinimumDatabaseScanInterval - 00:01:00, 00:00:00, 00:30:00
  BackoffIntervalForProxyConnectionLimitReached - 00:05:00, 00:00:30, 1.00:00:00
  DataGuaranteeCheckPeriod - 00:00:05, 00:00:01, 02:00:00
  DataGuaranteeTimeout = 00:10:00, 00:00:00, 12:00:00
  DataGuaranteeLogRollDelay = 00:01:00, 00:00:00, 12:00:00
  DataGuaranteeRetryInterval = 00:15:00, 00:00:00, 12:00:00
  DataGuaranteeMaxWait = 1.00:00:00, 00:00:00, 7.00:00:00
  DelayCheckPeriod = 00:00:05, 00:00:01, 00:01:00
  MailboxLockoutTimeout = 02:00:00, 00:00:00, 12:00:00
  MailboxLockoutRetryInterval = 00:05:00, 00:00:30, 12:00:00
  EnableDataGuaranteeCheck = true, false, true
  DisableMrsProxyBuffering = false, false, true
  WlmThrottlingJobTimeout = 00:05:00, 00:00:00, 12:00:00
  WlmThrottlingJobRetryInterval = 00:10:00, 00:00:00, 12:00:00
  MRSProxyLongOperationTimeout = 00:20:00, 00:01:00, 02:00:00
  ContentVerificationIgnoreFAI = false, false, true
  ContentVerificationIgnorableMsgClasses = "" (semicolon-separated list of wildcarded message classes)
  ContentVerificationMissingItemThreshold = 0, 0, MaxInt
  DisableContentVerification = false, false, true
  PoisonLimit = 5, 0, 100
```

Figure 11-4: Some of the configurable settings of the Mailbox Replication Service configuration file

The file lists the defaults as well as the minimum and maximum values which can be configured. When the goal is to increase migration performance, settings such as MaxActive* can be modified to increase migration throughput. However, take care when modifying the values as they are set to their default values by Microsoft for a reason. Most settings are in place to prevent mailbox moves from overwhelming a server's resources. However, if you system is robust (CPU/RAM/Disk), you can certainly tweak the settings to squeeze extra performance from the server. These settings aren't just for migration performance troubleshooting of course. When moving tens of thousands of mailboxes, even a 10% increase in performance can have drastic improvements at scale. The Microsoft Exchange Mailbox Replication Service must be restarted for any changes to take effect.

Other instances of throttling may be experienced when migrating to Office 365. The Microsoft Migration Team has provided the following AnalyzeMoveRequestsStats script to aid in identifying the source of performance issues. The article also details several possible causes of slowness:

- High Transient Failures

- Misconfigured Load Balancers
- High Network Latency
- Resource Constrained Source Servers
- Scale Issues
- Office 365-side Issues
    - Office 365 system resources
    - Content Indexing-related slowness

This Exchange Online migration performance and best practices article is an excellent resource for diagnosing Exchange Online migration performance issues. Also, while migrating large amounts of data to Office 365 it may be required to create a support ticket requesting that Office 365 support ease throttling limits for your tenant. This can increase migration throughput, but know that Microsoft typically only leaves this exception in place for 90 days.

Note: While not directly related to mailbox moves, other forms of EWS throttling can impact mailbox performance, in particular service accounts using EWS to access Exchange. A new Throttling Policy may need to be created and customized to prevent the service account from being adversely impacted by the Exchange EWS Throttling.

While working with mailbox moves, you may be presented with move stalls which have a "MoveRequestStatus" of RelinquishedWlmStall. This is often happening due to Workload Management, which can throttle Exchange actions if system resources such as CPU, Content Indexing, Replication Health, etc. are unhealthy. The Move Report may indicate why the throttling is occurring. However, in certain cases the behavior is unexplained to the Administrator. If the unhealthy system state cannot be corrected or identified, the best approach would be to contact Microsoft Support for assistance. However, there are a few options. The recommended approach would be to set the "Priority" of the Move request to "Emergency", which should bypass the Workload Management settings.

```
[PS] C:> New-MoveRequest –Identity <MailboxName> -TargetDatabase <DatabaseName> -Priority Emergency
```

This should allow the Move request to complete in a timely fashion. If you are configuring this setting on an existing Move request using Set-MoveRequest, then I recommend restarting the Microsoft Exchange Mailbox Replication Service afterwards for it to take immediate effect. However, another option (which should only be used if the above method fails) is to bypass WLM for all Move Requests. This option is not recommended unless all other avenues are first pursued and it's recommended to revert the setting once your task is accomplished. I only mention it here because I've had colleagues tell me sometimes the above method is not successful.

```
[PS] C:> Get-ExchangeServer | ?{$_.AdminDisplayVersion -like "*15*"} | ForEach {New-SettingOverride -
Component "WorkloadManagement" -Name "$_ MRS Override" -Server $_.Name -Section
MailboxReplicationService -Reason "Please move already!" -Parameters Classification=Urgent -MinVersion
15.0}
```

Additional References:

- Exchange 2013 Local Mailbox Moves
- Modifying the MaxMRSConnections value

Overcoming corruption during mailbox moves

Mailbox corruption is not a new problem for Exchange Administrators, Support Engineers, and Consultants. It has plagued them in every version of Exchange since its inception. Causes of corruption can vary and include:

- Hardware/Storage failures
- Power failures
- File System failures or corruption
- Faulty ActiveSync clients
- Faulty third-party Outlook add-ins
- Anti-Virus software (faulty or missing proper exclusions)

Much of this corruption may go undetected during normal operations. Event logging for physical database corruption can be found in the Application logs, but logging for corruption within individual mailboxes is somewhat sparse. In fact, most people will only discover mailbox corruption when attempting to move a mailbox either to new databases within the company or across Exchange organizations. They'll find the mailbox moves will fail due to bad items being encountered, and because the default BadItemLimit will be zero for all move requests it's not at all uncommon. In some cases, after a database corruption event where the mailbox database was left with bad checksums or an ESEUTIL /P (database repair) had to be run against it, the mailboxes within the database are left in a corrupted state (hence Microsoft's new support policy stating that a database must be vacated after an ESEUTIL /P was run against it). Let's discuss behaviors when corruption is present as well as how to recover from them.

Mailbox Quarantine

Often times, it is possible mailboxes may become so corrupted and unstable that they become quarantined by Exchange. A quarantined mailbox means the user will be unable to access the mailbox using any client. A mailbox will become quarantined if it causes the Information Store processes to hang or crash repeatedly. That mailbox will then be inaccessible by any mail client (such as Outlook, Outlook Web App, ActiveSync, etc.) until a given time period has expired. The default for this "penalty box" time period is six hours and can be customized by modifying the registry. The Mailbox Quarantine feature was created in Exchange 2010's lifetime but options were limited when it came to detecting and configuring it. You were limited to searching the below Windows Registry key for quarantined entries and then clearing them by deleting the registry entry for a particular mailbox.

HKLM\SYSTEM\CurrentControlSet\Services\MSExchangeIS\<ServerName>\Private-{dbguid}\QuarantinedMailboxes\{mailbox guid}

Fortunately, with Exchange 2013 came cmdlets to help administer Mailbox Quarantine. These commands were:

- Enable-MailboxQuarantine
- Disable-MailboxQuarantine

These commands replaced the manual deletion of registry keys as a means to disable mailbox quarantine for mailboxes. In addition to the above commands, the below command can be used to detect which mailboxes are currently quarantined.

```
[PS] C:> Get-Mailbox -ResultSize unlimited | Get-MailboxStatistics | Select DisplayName,
IsQuarantined | Format-Table –AutoSize
```

Types of Corruption

Sometimes you'll have mailboxes that frequently get quarantined, requiring they be repaired. Let's first discuss the two categories of corruption in an Exchange mailbox or mailbox database - physical corruption and logical corruption. The analogy I like to use when discussing Exchange physical vs. logical corruption is that of a damaged book. If a book's pages are torn out and its binding is damaged, I compare that to physical corruption in a database. This is where we would need to run ESEUTIL against the database to repair its physical structure, either by running an ESEUTIL /R (recovery) or the dreaded ESEUTIL /P (repair). For details on ESEUTIL, please see the Backup and Disaster Recovery chapter. Now, once the book's pages and binding have been repaired (with all pages now in the correct order), it could still have logical corruption if the words on the pages don't make sense to the reader, if the letters are smudged, the words are out of order, or it's now in the wrong language. For this, Exchange had a utility called ISINTEG, which would correct any logical corruption that resulted in Exchange being unable to properly decipher and process data in the Jet database. The symptoms of logical corruption were often display or search-related issues in Exchange clients, or in some cases, messages not displaying at all. I worked with one customer who were unable to sync ActiveSync devices because of a combination of physical and logical corruption in their mailboxes. While ISINTEG was certainly useful for recovering from this, starting with Exchange 2010 SP1, ISINTEG was replaced with the New-MailboxRepairRequest Exchange Management Shell cmdlet.

Repairing Logical Corruption in a Mailbox

New-MailboxRepairRequest allows you to target individual mailboxes instead of taking the entire database offline. However, it should be known that while an individual mailbox is being scanned, the mailbox data is made unavailable to the end user. The New-MailboxRepairRequest cmdlet takes the following format:

```
[PS] C:> New-MailboxRepairRequest -Mailbox <MailboxName> -CorruptionType <CorruptionType>
```

Common Corruption Types:

- AggregateCounts
- Searchfolder
- Folderview
- Provisionedfolder

Additional, less used, Corruption Types exist. If a repair request fail using the common Corruption Types, you can attempt a repair of all Corruption Types using the following command.

```
[PS] C:\> New-MailboxRepairRequest -Mailbox <MailboxName> -CorruptionType
SearchFolder,FolderView,AggregateCounts,ProvisionedFolder,ReplState,MessagePTAGCn,MessageID,RuleMessageClass
,RestrictionFolder,FolderACL,UniqueMidIndex,CorruptJunkRule,MissingSpecialFolders,DropAllLazyIndexes,ImapID,
ScheduledCheck,Extension1,Extension2,Extension3,Extension4,Extension5
```

In some situations, you can also use the CorruptionType of LockedMoveTarget if you encounter a mailbox move that has become locked.

In Exchange 2010, you monitor the progress of mailbox repair requests in Event Viewer. However, in Exchange 2013/2016, you instead use the Get-MailboxRepairRequest cmdlet. A common example of the command would be:

```
[PS] C:> Get-MailboxDatabase | Get-MailboxRepairRequest | Format-Table Identity
```

Repairing Physical Corruption

If a repair request does not resolve the corruption issues, we would need to repair physical corruption by moving the mailbox to a new database. This action effectively creates a new mailbox by copying the old mailbox data to it. Take care however, as this action will likely result in mailbox data being purged if it is deemed corrupt. You'll likely need to use the *BadItemLimit* and *AcceptLargeDataLoss* parameters of the New-MoveRequest command to allow it to complete successfully. Be sure to weigh the risks of this action before you take it. Below is an example of the New-MoveRequest cmdlet being used in a data loss scenario:

```
[PS] C:> New-MoveRequest –Identity <MailboxName> -TargetDatabase <TargetDB>–BadItemLimit 200
–AcceptLargeDataLoss
```

In certain circumstances, even moving the mailbox may not be successful. I've encountered scenarios where the move request would fail because the mailbox kept being quarantined or some other unidentifiable error occurred. While our options were limited in Exchange 2010, Exchange 2013/2016 gave us a very useful parameter in the New-MoveRequest cmdlet; the *ForceOffline* parameter. Starting in Exchange 2007 SP3, mailbox moves were an online process where users could still access their mailbox while it was being moved. The *ForceOffline* parameter forces the move to be an offline process, similar to Exchange 2007 SP2 and older. I've found much success with using this command to overcome mailbox corruption issues during move requests. Using this switch will prevent a mailbox from becoming quarantined or crashing the store during the move, as well as keep the user from accessing an already corrupted mailbox. A common example of the command in action is as follows:

```
[PS] C:> New-MoveRequest –Identity <MailboxName> -TargetDatabase <TargetDB>–BadItemLimit 200
–AcceptLargeDataLoss -ForceOffline
```

If all of these recovery actions fail and the mailbox still will not move, the remaining options are to:

- Perform an Offline Defrag against the mailbox database holding the corrupted mailbox
- Export the contents of the corrupt mailbox to .PST using New-MailboxExportRequest
- Export the contents of the corrupt mailbox to .PST using an Outlook client

In each of the last two options, once the data is exported you should disable the mailbox, create a new mailbox, and import the data using either the New-MailboxImportRequest or an Outlook client.

Public Folders

Historically, the mention of Public Folders to an Exchange Server Support Engineer meant induced anxiety and potentially a four-letter expletive. This was because Legacy Public Folders (pre-Exchange 2013) utilized a multi-master replication technology enabling multiple copies of public folder content to be replicated (SMTP-based content replication) across potentially geographically disperse Exchange Servers. Not only did this allow for data redundancy, but also for users to have speedy access to their local replicas. Unfortunately, this replication technology was also the catalyst of many Public Folder support cases to Microsoft. Common Legacy Public Folder support issues included (with common causes):

- Inability to access Public Folder data
    - Dismounted Public Folder Database/Lack of Public Folder permissions/Referrals disabled on Routing Group Connector (RGC)
- Viewing stale content due to Public Folder replication delay
    - Network latency/Mail queue full/Firewall

- New Public Folder content not being displayed due to replication failure
    - Corrupted Public Folder content or database/Network failure/Firewall
- Mail flow issues related to Mail Enabled Public Folders
    - Mail queue full or slow/no valid route to destination
- Public Folder replication storms due to large content being replicated across the WAN
    - Large amount of data placed in Public Folder/Mass changes or additions
- Complications during Public Folder data restoration

Note: For the last bullet, I highly recommend reading this excellent article series on Recovering Public Folders After Accidental Deletion.

If troubleshooting Public Folder issues during normal operations were no easy task, troubleshooting Legacy Public Folder Migrations could test the patience and aptitude of any Exchange Professional. At a high level, the Legacy Public Folder migration process was as follows:

- Create Public Folder Database on new Exchange version
- Add Replicas of existing Public Folders to the new Public Folder Database
- Ensure new Public Folder Databases have Replicas of all Public Folder content held on old servers
- Verify all mailboxes can access Public Folder content held in new Exchange Servers' Public Folder Database
- Remove Replicas from old servers and wait for content to be removed
- Dismount and remove old Public Folder Database

At this point, mailboxes on all Exchange versions (2003/2007/2010) could still access the Legacy Public Folder content. This is because after a decade of Exchange development, the underlying architecture of Public Folders had not changed. The only considerable change was that newer Outlook clients (Outlook 2007/2010 on Exchange 2007/2010 mailboxes) did not require Public Folders for accessing Free/Busy, Out Of Office, or Offline Address Book information. Instead they relied on new web services introduced in Exchange 2007, such as the OAB and EWS Virtual Directories.

Common issues experienced during Legacy-to-Legacy Public Folder migrations involved:

- Data not being replicated to new Public Folder Databases after Replicas are added
    - Replication failure (Check Event Logs and Queue Viewer)/Routing Group Connector missing or misconfigured (recreate RGC)
- Mail-Enabled Public Folders routing issues
    - Routing Group Connector missing or misconfigured (recreate)/Receive Connector missing or misconfigured/No valid route to destination/Deny ACL (Suggest running Setup /PrepareAD again)/ Verify mail flow between versions (port 25 connectivity, Exchange Server Auth on Receive Connectors, and Windows Integrated Auth enabled on SMTP Virtual Server)
- Permissions issues with Routing Group Connector (2003-to-2007/2010)
    - Delete and recreate RGC
- Issues with legacy mailboxes querying Free/Busy data for new mailboxes
    - Verify Public Folder replication/Verify Referrals are enabled on RGC (Set-RoutingGroupConnector)
- Inability to remove old Public Folder Database due to ghosted Replicas
    - Verify all content has been replicated to new Public Folder Database (Get-PublicFolderStatistics)

Many of these migration issues had similar root causes to Public Folder operational issues. Namely around Public Folder Replication (SMTP-based replication) health. Replication health became such a burdensome task that it drove Microsoft to remove the technology entirely from Modern Public Folders.

I won't spend as much time discussing troubleshooting Legacy-to-Legacy Public Folder migrations, as they should now be much less frequent. The only scenario where I would still expect to see such a migration would be for those few poor souls still on Exchange 2003 and would require moving to Exchange 2010 (via a legacy-to-legacy migration) before moving to 2016 (via a Legacy-to-Modern Public Folder migration). If a customer were still using Legacy Public Folders on Exchange 2007, I would recommend they move to Modern Public Folders on Exchange 2013 and then move to Exchange 2016.

However, here are a few useful commands which can be used to troubleshoot a Legacy-to-Legacy Public Folder migration:

Get-PublicFolderStatistics - Gather item count for Public Folders (suggest viewing on each replica to ensure consistency). Useful for comparing content replicated between the source and destination servers.

```
[PS] C:> Get-PublicFolderStatistics –Identity "\FolderName" –ResultSize Unlimited
```

Get-PublicFolder – View Public Folders and their Replicas

```
[PS] C:> Get-PublicFolder -Identity "\FolderName" -Recurse | Format-List Name,Replicas
```

Update-PublicFolderHierarchy – Manually replicating the Public Folder hierarchy. Useful for when adding a new Public Folder Database and ensuring it has the hierarchy replicated to it.

```
[PS] C:> Update-PublicFolderHierarchy -Server "SourceServer"
```

Update-PublicFolder – Manually initiate synchronization of Public Folder content.

```
[PS] C:> Update-PublicFolder "\Folder\SubFolder" -Server "SourceServer"
```

Additional References:

- Public Folder replication troubleshooter
- Public Folder Hierarchy Replication Problems
- Managing Exchange Public Folder Permissions
- Exchange 2010 FAQ: How Do I Migrate Public Folders to Exchange Server 2010?
- Public Folder Replication – Troubleshooting Basics
- Troubleshooting the Replication of New Changes
- Troubleshooting the Replication of Existing Data
- Troubleshooting Replica Deletion and Common Problems
- Exchange Server 2007/2010 tips

Modern Public Folders

In developing Exchange 2013, the Exchange Product Team removed almost all of the pain points of Legacy Public Folders to create what we now know as Modern Public Folders. From an end-user perspective, Modern Public Folders behave almost exactly like Legacy Public Folders. Mail, Contact, and Calendar items can still be stored within them and accessed by anyone in the organization. They can still be accessed or created via Outlook and OWA (limited functionality) from any location. In fact, if planned and performed correctly, an end user will not notice their Public Folders have been migrated to Exchange 2013/2016. Actually, an argument

could be made that a Modern Public Folder isn't much different (from an end-user perspective) than a Shared Mailbox, with the exception that Public Folders maintain individual read/unread status.

The big change came with the removal of multi-master SMTP-based replication of Public Folder content. With Modern Public Folders there is only ever one instance of a folder and its contents in the entire Exchange organization. Gone also are Public Folder Databases, as all content is now stored in Public Folder Mailboxes, which are effectively system mailboxes used for serving Public Folder hierarchy and content to clients. Public Folder high availability is now achieved through DAG replication. So gone is much of the complexity around both Public Folder operations and migrations.

However, that's not to say there aren't caveats migrating to and managing Modern Public Folders. The biggest challenge is the fact that there is no coexistence between Legacy and Modern Public Folders. It is not supported (yet technically possible) to have Legacy and Modern Public Folders accessed at the same time.

Note: It is technically possible for Exchange 2013/2016 mailboxes to access Modern Public Folders while Exchange 2007/2010 mailboxes in the same Exchange Organization are accessing Legacy Public Folders. This is done by unlocking the Legacy Public Folders once the migration has been complete. While all access and production activity still happens on Modern Public Folders, this is not supported. No changes on the Legacy Public Folders will replicate to Modern Public Folders or vice versa. The only use case I've encountered was a customer who prematurely completed the Modern Public Folder migration and allowed it to run in production for several weeks before realizing several key folders were missing. They used the above method and an Exchange 2007 mailbox to extract the contents from the Legacy Public Folders via Outlook to .PST.

Since coexistence is impossible for Legacy and Modern Public Folders, there will come a time when a cutover must be performed. This is done after all mailboxes are migrated to Exchange 2013/2016, since Exchange 2007/2010 mailboxes are unable to access Modern Public Folders. Fortunately, Exchange 2013/2016 mailboxes can access Legacy Public Folders, so general Exchange coexistence (Exchange 2007 and 2013 as an example) is possible for as long as needed. The overview of a Legacy-to-Modern Public Folder migration is as follows:

- Implement Exchange 2013/2016 into an existing Exchange environment which contains Legacy Public Folders which the customer wishes to migrate
    - You may consider SharePoint or Shared Mailboxes as an acceptable alternative to Public Folders, and therefore would not require a migration to Modern Public Folders
- Implement required steps allowing Exchange 2013/2016 to access Legacy Public Folders
    - On-Premises Legacy Public Folder Coexistence for Exchange 2013 Cumulative Update 7 and Beyond (A **must read** if you want successful access to Legacy Public Folders)
- Move namespaces to the new version of Exchange
    - Outlook Anywhere clients will connect to the new version of Exchange before being proxied to the legacy version
    - Outlook Anywhere configuration must be modified for 2013/2016 mailboxes to access Legacy Public Folders (Reference – Common issue, a **must read**)
- Migrate all mailboxes to the new version of Exchange
    - This phase may last many months depending on the migration project timeline
- Migrate Legacy Public Folder content to Modern Public Folders
    - Data will be synchronized in the background until the cutover is ready to occur
- Decommission Legacy Public Folder databases
    - Remove Public Folder Databases

Note: If the prerequisites for accessing Legacy Public Folders not be configured in an Exchange 2013 CU7 or newer environment as stated above, Outlook clients will be unable to open Public Folders.

Creating the Public Folder Discovery Mailboxes allows AutoDiscover to instruct the Outlook client which connectivity settings/endpoints to use for accessing Public Folders.

Additional References:

- Exchange Server 2010 to 2013 Migration – Moving Public Folders
- Legacy Public Folders to Exchange 2013 migration tips

When Exchange 2013 first released, the mechanism to actually move the content from Legacy to Modern Public Folders was referred to as a Serial Migration. This migration method is deprecated and no longer supported by Microsoft. The new preferred method is referred to as the Batch Migration method:

- Use batch migration to migrate public folders to Exchange 2013 from previous versions
- Use batch migration to migrate public folders to Exchange 2016 from previous versions

The Exchange 2013 and 2016 version of these articles are virtually the same. Each involve the following steps:

- Download migration scripts
- Prepare for migration by gathering public folder statistical data and preparing the organization
- Generate CSV files to be used for Public Folder Mailbox mapping
- Create Public Folder Mailboxes in Exchange 2013/2016
- Begin migration request by executing New-MigrationBatch and awaiting data to be copied
- Lock down Legacy Public Folders (Involves Downtime)
- Finalize Public Folder migration by completing Migration Batch
- Test access and unlock Modern Public Folders for access
- Verify all data was migrated by using Get-PublicFolder and Get-PublicFolderStatistics
- (Optional) Roll back migration by unlocking Legacy Public Folders and deleting Modern Public Folder Mailboxes

In my experience, most issues experienced during these phases are simply syntax issues with the commands which can easily be remedied by re-reading the TechNet articles and practicing in a lab. Other issues revolve around the Migration Failing or being stuck in a particular status. Using the methods previously mentioned in this chapter for viewing Migration batches is recommend for initial analysis. However, I've found that at times restarting the Microsoft Exchange Mailbox Replication Service on the target server or restarting the Microsoft Exchange Information Store Service on the source legacy server will unlock the batch and allow it to complete. Also, be sure to verify Active Directory Replication in a large environment.

Miscellaneous Coexistence Issues

Updated Outlook Clients

I cannot stress the importance of Outlook clients being at the latest updates, in general, but especially during period's coexistence. Microsoft performs extensive testing of their products, however there's only so much testing they can be expected to do against all supported server and client products interacting with each other. For instance, Exchange 2013 CU11 supports accessing Legacy Public Folders on Exchange 2007 SP3 UR10. It also supports an Outlook 2007 SP3 November 2012 update connecting to an Exchange 2013 mailbox in this scenario, and accessing Legacy Public Folders. However, let's just think about the situation we're dealing with here. An environment running the latest version of Exchange 2013, but the mere supported minimum from a client and coexistence perspective. A client version released in Nov 2012, a legacy Exchange Server released February 2013, and a server handling connectivity between the two running code released 3 years later.

In my opinion, there's a frustrating philosophy amongst some IT professionals that updating software introduces risk and therefore only update if they are required to do so. Therefore, they want to run the minimum supported software version as a means of avoiding risk. While they are not incorrect in the opinion that change is a risk, they often fail to realize something important when dealing with software products which interact with each other. The more disparate in version codependent products become, the more risk is introduced by way of instability, lack of security, and performance.

I claim no inside knowledge of the Exchange and Outlook team's product testing. Yet I feel they do not possess the resources or time to test a new Cumulative Update against every single past Exchange update version as well as every past Outlook update version, in every possible coexistence scenario. In my above example, the end user is going to have a much better experience if the Exchange 2007 server were running the latest Update Rollup (19 as of this writing) and their Outlook 2007 client were at the latest update (February 2015 as of this writing). Of course, ideally they would be running Outlook 2010/2013/2016 instead, all which have updates as recent as January 2016 (as of this writing).

I've been personally told by Exchange Product Team members of several coexistence issues which were resolved simply by updating Outlook. There are many moving parts to an Outlook client connecting to Exchange, being proxied to a legacy version, and accessing a legacy resource. There are also new features constantly being released to the Office suite. All of these result in bugs that may pop-up along the way. You can either choose to never update any software product in an Exchange environment (leaving you in an unsupported state that business might be impacted) or accept that there is a constantly moving window of updates which your clients must all be within to properly operate with each other. In addition to Microsoft's guidance, I've experienced several issues myself in coexistence which were resolved either by an Outlook or Exchange update:

- Poor performance when bring proxied from a newer version of Exchange to a legacy version
- Poor performance when accessing a legacy Public Folder or shared/delegated mailbox on a legacy Exchange Server
- Inability to modify items or calendar of a mailbox you've been given permissions to
- Connection failures or random disconnects in Outlook when accessing a legacy resource
- Connectivity failures when using MAPI/HTTP with older (but supported) Outlook updates
- Issues accessing an Office 365 hosted Archive Mailbox

It's important to stress that while my example used Outlook 2007, similar issues can occur with Outlook 2010/2013/2016 on older update versions. Outlook 2013 has experienced over 30 code revisions since its release, many to resolve coexistence or compatibility issues. In one of my examples above, the scenario I witnessed with a customer was resolved by updating Outlook 2013 to a version that was only 6 months newer. I've found it's a good rule of thumb to use the N-2 rule not only for Exchange Server update cadence, but also for Outlook updates.

Additional References:

- How to install the latest applicable updates for Microsoft Outlook
- General Microsoft Support article for diagnosing Outlook issues frequently resolved with updates
- Latest Outlook Updates
- Exchange Server and Update Rollup Build Numbers
- Exchange 2013 System Requirements
- Exchange 2016 system requirements

Accessing Legacy Resources

As previously mentioned, accessing legacy resources in a coexistence scenario is a common pain point during migrations. During a migration, namespaces are to be pointed to the newest Exchange version and connectivity is proxied back for legacy mailboxes or when new mailboxes access legacy resources. A couple key issues have been common call generators and are still very valid for future migrations:

- Users of Exchange Server 2013 or later or Exchange Online can't open public folders or shared mailboxes on a legacy Exchange server
    - As legacy Exchange versions do not support Anonymous authentication, so the Outlook Anywhere settings must be modified if shared resources or Legacy Public Folders are to be accessed on the legacy Exchange Servers
- Outlook Anywhere users prompted for credentials when they try to connect to Exchange Server 2013 or Exchange Server 2016
    - A bug in Server 2008 R2 causes proxied sessions to have authentication failures. Therefore when the newer Exchange version proxy's sessions to the legacy Exchange versions, they will be repeatedly prompted for authentication. A Windows Server 2008 R2 update resolves the issue.

Speaking of Shared Mailboxes and delegated mailboxes (mailboxes which another mailbox has been assigned permissions to), care must be taken in environments where a large number of such mailboxes exist. This is a great example of something being supported not necessarily ensuring a great end user experience. It's of my opinion, and the opinion of many Exchange Consultants that these mailboxes and the users accessing them should be kept on the same Exchange version. Meaning that if they all currently exist on Exchange 2007 and you're migrating to Exchange 2013, they should all be moved at the same time to ensure the best possible user experience. If a Secretary or Assistant manages 10 individuals' mailboxes, then all 11 mailboxes should be moved at the same time. I've worked several escalations where an Assistant could not properly manage an Executives calendar because the Exec's mailbox had been moved to the newer Exchange version but their mailbox was still on the legacy Exchange version. While you could certainly spend hours, days, or even weeks on the phone with Microsoft trying to resolve the issue (since it may be supported), in almost every case it's much simpler to just move the Assistant to the same server version as the Exec. I would even say moving them to the same Mailbox Database would be ideal, as it may slightly improve performance of the repeated access.

Delays in accessing mailbox after migration to Exchange 2013/2016

A change in behavior was implemented in Exchange 2013/2016 where data was cached in AutoDiscover Application Pools for an extended period of time before new data was populated. While this change had the goal of improved performance, in some environments (typically smaller environments with fewer AutoDiscover queries) the cache could remain stale and result in connectivity issues such as those experienced in the Microsoft Knowledge Base article below:

- Outlook logon fails after mailbox moves from Exchange 2010 to Exchange 2013 or Exchange 2016

The symptom is that Outlook clients could fail to connect after the mailbox is moved to Exchange 2013/2016. This is due to the AutoDiscover cache located in the AutoDiscover Application Pool on each Exchange Server having outdated data which still references the mailbox being located on the legacy Exchange Server. Upon opening Outlook after the move completes, AutoDiscover will issue an HTTP 302 response, which is a redirect resulting in a loop. This can be seen in the Log tab of the Test E-mail AutoConfiguration tool within Outlook. The workaround is to restart the AutoDiscover Application Pool on each Exchange Server where the Outlook client could potentially retrieve AutoDiscover information from, immediately following the completion of a mailbox migration to Exchange 2013/2016.

While it's uncertain whether this behavior will change in future Exchange Server updates, for the time being, to prevent a negative user experience during a migration you should plan an Active Directory site-wide AutoDiscover Application Pool restart after the completion of any move requests to Exchange 2013-2016. Of course the effect of this behavior may differ drastically between environments, so I would recommend testing the post-move behavior using test mailboxes early in the migration.

Exchange Management Shell and Mailbox Anchoring

In December 2015, Microsoft announced a change in the method Exchange Management Shell uses to connect to Exchange Servers. In summary, when Exchange 2013 CU11 is installed, EMS connects to the Exchange Server where the user connecting's mailbox is hosted. If they do not have a mailbox, then it will connect to the server hosting the Arbitration mailbox (specifically the mailbox named *SystemMailbox{bb558c35-97f1-4cb9-8ff7-d53741dc928c}*). If these mailboxes are unavailable (database dismounted or mailbox deleted) the EMS session will fail to connect. However, another scenario was experienced specifically during migrations. If migrating from Exchange 2007/2010 to Exchange 2013, and the version of Exchange 2013 initially installed was CU11, 2013 EMS would not properly function. This is because both the connecting user's mailbox as well as the arbitration mailboxes were still located on the legacy Exchange Servers. Normally it's recommended to move administration and arbitration mailboxes to the new version of Exchange as soon as possible. However, in this scenario (as 2013 CU11 was only just installed) this was not yet possible. If this scenario is encountered, follow the guidance in this article: Mailbox Anchoring affecting new deployments & upgrades

Additional reading

Migration Overview

- Negotiate Authentication
- Remote Mailbox Moves
- Linked Mailboxes
- Cross-Forest Mail Flow
- Cross-Forest Connectors
- Mailbox Replication Service Proxy (MRSProxy)
- Exchange Hybrid Configuration
- A user can't access a mailbox by using Outlook after a remote mailbox move from an on-premises Exchange Server environment to Office 365
- Do not convert synced mailboxes to shared in a hybrid environment
- Cutover Migration
- Staged Migration
- IMAP Migration
- Exchange Remote Move (MRSProxy)

Moving Mailboxes

- Preferred Architecture
- Server Role Requirements Calculator
- Move Requests
- Exchange Mailbox Dumpster
- New-MoveRequest
- Throttling the Mailbox Replication Service

- Moving Exchange Server 2013 Mailboxes
- Resume-MoveRequest
- Get-MoveRequest
- Get-MoveRequestStatistics
- The Bad Item Conundrum (or, How much data would you like MRS to drop?)
- Set-MoveRequest
- Suspend-MoveRequest
- Resume-MoveRequest
- Remove-MoveRequest
- Migration Batches
- New-MigrationEndpoint
- Get-MigrationBatch
- New-MigrationBatch
- Set-MigrationBatch
- Start-MigrationBatch
- Stop-MigrationBatch
- Complete-MigrationBatch
- Remove-MigrationBatch
- Get-MigrationUser
- Remove-MigrationUser
- Get-MigrationUserStatistics
- New-MigrationEndpoint
- Set-MigrationEndpoint
- Remove-MigrationEndpoint
- Get-MigrationConfig
- Set-MigrationConfig
- Get-MigrationStatistics
- Test-MigrationServerAvailability
- Background on Migration Arbitration Mailboxes
- Error message when you try to move a mailbox in Exchange Server 2016: "The migration mailbox for the organization is either missing or invalid
- Large transaction logs are generated when you move mailboxes in Exchange Server 2013 or Exchange Server 2016 Administration Center
- System Attendant Mailbox and Exchange 2013
- Exchange – Issues if system mailboxes are not correctly configured
- Exchange 2013 recreate arbitration mailboxes
- Retrieving the Move History for an Exchange Server 2010/2013 Mailbox
- Throttling the Mailbox Replication Service
- Tweaking the Mailbox Replication Service configuration file
- Mailbox Migration Performance Analysis
- Exchange Online migration performance and best practices
- EWS Throttling in Exchange
- Bypassing throttling for a service account
- New-ThrottlingPolicy
- Set-ThrottlingPolicy
- Exchange 2013 Local Mailbox Moves
- Modifying the MaxMRSConnections value

Overcoming Corruption During Mailbox Moves

- Event logging for physical database corruption
- New Support Policy for Repaired Exchange Databases
- Mailbox quarantining in Exchange 2010 and Exchange 2013
- How To Set Mailbox Quarantine In Exchange
- Enable-MailboxQuarantine
- Disable-MailboxQuarantine
- ActiveSync Issues due to Mailbox corruption
- New-MailboxRepairRequest
- Troubleshooting MRS Health Set
- View Mailbox Repair Request Entries in Event Viewer (Exchange 2010)
- Get-MailboxRepairRequest (Exchange 2013/2016)
- How to Defrag an Exchange 2010 Mailbox Database
- New-MailboxExportRequest
- Disable-Mailbox
- New-MailboxImportRequest

Public Folders

- Recovering Public Folders After Accidental Deletion
- Get-PublicFolderStatistics
- Get-PublicFolder
- Update-PublicFolderHierarchy
- Update-PublicFolder
- Public Folder replication troubleshooter
- Public Folder Hierarchy Replication Problems
- Managing Exchange Public Folder Permissions
- Exchange 2010 FAQ: How Do I Migrate Public Folders to Exchange Server 2010?
- Public Folder Replication – Troubleshooting Basics
- Troubleshooting the Replication of New Changes
- Troubleshooting the Replication of Existing Data
- Troubleshooting Replica Deletion and Common Problems
- Exchange Server 2007/2010 tips
- On-Premises Legacy Public Folder Coexistence for Exchange 2013 Cumulative Update 7 and Beyond
- Remove Public Folder Databases
- Exchange Server 2010 to 2013 Migration – Moving Public Folders
- Legacy Public Folders to Exchange 2013 migration tips
- Serial Migration
- Use batch migration to migrate public folders to Exchange 2013 from previous versions
- Use batch migration to migrate public folders to Exchange 2016 from previous versions

Miscellaneous Coexistence Issues

- Outlook and Outlook for Mac Update Build Numbers
- How to install the latest applicable updates for Microsoft Outlook
- General Microsoft Support article for diagnosing Outlook issues frequently resolved with updates
- Latest Outlook Updates
- Exchange Server and Update Rollup Build Numbers

- Exchange 2013 System Requirements
- Exchange 2016 system requirements
- Users of Exchange Server 2013 or later or Exchange Online can't open public folders or shared mailboxes on a legacy Exchange server
- Outlook Anywhere users prompted for credentials when they try to connect to Exchange Server 2013 or Exchange Server 2016
- Outlook logon fails after mailbox moves from Exchange 2010 to Exchange 2013 or Exchange 2016
- HTTP 302 Response
- Exchange Management Shell and Mailbox Anchoring
- Mailbox Anchoring affecting new deployments & upgrades

Chapter 12: Troubleshooting Security

Paul Cunningham

Throughout this book we've discussed many scenarios that are security-related, such as user authentication, Active Directory permissions, mailbox permissions, and more. This chapter doesn't deal specifically with troubleshooting of security problems. Rather, it provides you with knowledge of security-related tools in Exchange that you can use in a wide variety of situations.

Think about this chapter as more about answering the security question, "Who did that?", rather than security in terms of granting access to things.

Mailbox Audit Logging

In many organizations, the Exchange administrator will encounter a situation where there is a need to determine who took action on an item in a mailbox. This will most often arise due to actions taken by delegates of a person's mailbox, or people who use a shared mailbox.

For example:

- An email message from a customer was never responded to, and the manager of the customer service team wants to know which person in the team moved or deleted the message from the shared mailbox.
- Information sent to an executive via email has leaked to the press or to a competitor and there is an investigation to determine which of the executive's delegates accessed the message.

In these situations, it is assumed that a delegate, or even a team of people, already have full or read-only access to the mailbox. Based on that assumption the focus is now on which of those people took action with specific items.

Exchange 2010 and later versions can log access to mailboxes by the owner, delegates, and administrators, using a feature called mailbox audit logging. When audit logging is enabled for a mailbox, audit log entries are stored in the Recoverable Items folder of the mailbox, which is not visible to the mailbox user via Outlook or other client interfaces.

Log entries are written for actions taken by the mailbox owner, delegates, or by administrators, depending on the audit logging configuration applied to the mailbox. The mailbox audit log entries are then retained for a configurable period of time, allowing administrators to perform audit log searches to determine who took an action on a mailbox.

A default mailbox audit logging configuration for an Exchange mailbox looks like this.

```
[PS] C:\> Get-Mailbox alan.reid | fl *audit*

AuditEnabled     : False
AuditLogAgeLimit : 90.00:00:00
AuditAdmin       : {Update, Move, MoveToDeletedItems, SoftDelete, HardDelete, FolderBind, SendAs,
                    SendOnBehalf, Create}
```

```
AuditDelegate    : {Update, SoftDelete, HardDelete, SendAs, Create}
AuditOwner       : {}
```

The default mailbox audit logging configuration can be described as follows:

- Mailbox audit logging is disabled. This means that if you do not enable it for your mailbox users, you will not have access to audit log information when it comes to troubleshooting scenarios.
- Audit log entries are retained for 90 days. While this will be adequate for the majority of organizations, you can increase or decrease the retention period to suit your needs. Mailbox audit logs retained for 90 days add between 2-5% to the overall size of the mailbox, based on an analysis I performed for the impact of audit logging in multiple environments.
- No owner actions are logged. Obviously owners are taking action on their own mailbox constantly, and auditing everything they do would cause a massive amount of audit logging to be generated. However, there are some owner actions such as deletes that can be useful to capture.
- Some delegate and administrator actions are logged. For the most part the defaults will be sufficient, but additional actions such as *FolderBind* are useful if there is a concern about delegates snooping around mailbox folders they're not supposed to be looking in.

Note: The AuditAdmin settings refer to access via mechanisms such as eDiscovery searches, mailbox import/export operations, or tools such as MFCMAPI. If an administrator is granted permission to a mailbox and accesses it then those actions will be logged according to the AuditDelegate settings.

Configuring Mailbox Audit Logging

The *Set-Mailbox* cmdlet is used to enable audit logging for mailboxes. Typical candidates for mailbox audit logging are executives or VIPs who handle sensitive information, and who have delegates, or shared mailboxes used by teams of people. Discovery mailboxes are often monitored using mailbox audit logging to ensure that no information copied by eDiscovery searches is removed before the data is passed to investigators.

```
[PS] C:>  Set-Mailbox alan.reid -AuditEnabled $true
```

The full list of actions that mailbox audit logging can capture are:

- Copy
- Create
- FolderBind
- HardDelete
- MailboxLogin
- MessageBind
- Move
- MoveToDeletedItems
- SendAs
- SendOnBehalf
- SoftDelete
- Update

To add more actions to an existing mailbox audit logging configuration, we use *Set-Mailbox* again. In this example, *HardDelete*, *SoftDelete*, and *MoveToDeletedItems* actions are added to the owner auditing of the mailbox of Alan Reid.

```
[PS] C:>  Set-Mailbox Alan.Reid -AuditOwner @{add='HardDelete,SoftDelete,MoveToDeletedItems'}
```

Using Mailbox Audit Log Searches

Once mailbox audit logging is enabled and data is being captured, you can use mailbox audit log searches to determine who took an action on a mailbox. As an example, let's say that someone has sent an in appropriate email using the "Help Desk" mailbox, and management wants to track down the culprit. Since you were proactive and enabled mailbox audit logging for all shared mailboxes in your organization, you're able to perform a search.

Using the *Search-MailboxAuditLog* cmdlet we can search the "Help Desk" mailbox for actions taken by delegates between two dates.

```
[PS] C:>  Search-MailboxAuditLog -Identity "Help Desk" -LogonTypes Delegate -StartDate 1/14/2014 -
EndDate 1/15/2014 -ShowDetails

RunspaceId                    : d8142847-166a-488a-b668-f7b84c3f3ceb
Operation                     : SendAs
OperationResult               : Succeeded
LogonType                     : Delegate
ExternalAccess                : False
DestFolderId                  :
DestFolderPathName            :
FolderId                      :
FolderPathName                :
ClientInfoString              : Client=MSExchangeRPC
ClientIPAddress               : 192.168.0.181
ClientMachineName             :
ClientProcessName             : OUTLOOK.EXE
ClientVersion                 : 15.0.4551.1004
InternalLogonType             : Owner
MailboxOwnerUPN               : Sarah.Jones@exchange2013demo.com
MailboxOwnerSid               : S-1-5-21-2175008225-1847283934-4039955522-1471
DestMailboxOwnerUPN           :
DestMailboxOwnerSid           :
DestMailboxGuid               :
CrossMailboxOperation         :
LogonUserDisplayName          : Sarah Jones
LogonUserSid                  : S-1-5-21-2175008225-1847283934-4039955522-1471
SourceItems                   : {}
SourceFolders                 : {}
SourceItemIdsList             :
SourceItemSubjectsList        :
SourceItemFolderPathNamesList :
SourceFolderIdsList           :
SourceFolderPathNamesList     :
ItemId                        :
ItemSubject                   : Wheeee!
DirtyProperties               :
OriginatingServer             : E15MB1 (15.00.0775.022)
MailboxGuid                   : a0f10db1-5268-47a5-8f71-d1e65f55c653
MailboxResolvedOwnerName      : Help Desk
LastAccessed                  : 14/01/2014 9:31:07 PM
Identity                      :
RgAAAAD2fF/dZobvQoWbbV7P6N7eBwD7Y5OF+DDRQZRz1a4+yUyzAABaldDBAAD7Y5OF+DDRQZRz1a4+yUyzAAB
 aldDCAAAJ
IsValid                       : True
ObjectState                   : New
```

In the output above we can see that the user "Sarah Jones" made a successful *SendAs* action on the "Help Desk" mailbox, with the inappropriate subject like of "Wheeee!". Management can now take the appropriate action to deal with the situation.

Real World: Running frequent mailbox audit log searches can become tedious. When there is a regular need to review mailbox audit logs, consider using my Get-MailboxAuditLoggingReport.ps1 PowerShell script to speed up the process. You can even automate regular reports using a scheduled task.

Administrator Audit Logging

Who changed that email address policy? Who dismounted that database? Who granted that person access to the CEO's mailbox? As an Exchange administrator those are all the type of questions you could be asked quite regularly, especially if you work in a large IT team with many administrators making changes on a daily basis. Fortunately, since Exchange 2010 we've been able to answer those questions using administrator audit logging.

Administrator audit logging captures all changes made my administrators using the Exchange management tools (PowerShell cmdlets, or the Exchange Admin Center). Only commands that make changes, for example *Remove-Mailbox*, are logged by administrator audit logging, whereas commands that do not effect data, such as *Get-Mailbox*, are not logged by administrator audit logging.

Configuring Administrator Audit Logging

In an Exchange organization, administrator audit logging has the following default configuration.

- Administrator audit logging is enabled.
- Administrator audit log information is retained for 90 days.
- All Exchange PowerShell cmdlets that can make modifications to objects or settings in the organization are audited.
- All parameters of the above Exchange PowerShell cmdlets are logged.
- Test cmdlets (such as *Test-MAPIConnectivity*) are not logged.
- The log level is set to "none", which doesn't mean nothing is logged, rather it means that the only details that are logged are the command that was run, who ran it, and which object they modified. The other log level option is "verbose", which in addition to those details already mentioned, also logs the old and new properties of the object that were modified by the command.

Administrator audit logging can be disabled, or the configuration modified to limit the cmdlets or parameters that are audited, or to modify the log retention period. For this reason, you should limit the ability of administrators in your organization to modify the administrator audit log settings. By default, this right is granted to members of Organization Management and Records Management. I recommend you review your membership of the Organization Management and Records Management role groups to ensure that only the most trusted administrators are members of those groups.

Note: Any changes made to the administrator audit log configuration are logged in the administrator audit logs, regardless of whether admin audit logging is enabled or disabled. So in theory you should see evidence of any tampering that has occurred.

Searching Administrator Audit Logs

Admin audit logs are reasonably simple to search using the Exchange management shell. There's a few different approaches you can take:

- Search for a specific cmdlet or cmdlets
- Search within a specific date range
- Search for actions taken by a specific administrator
- Search for actions taken against a specific object

You can also combine the above by using multiple parameters in your search.

Let's take a look at a simple example – someone has granted the user Alex Heyne access to the CEO Alannah Shaw's mailbox. We know this is done using the *Add-MailboxPermission* cmdlet, so we can use the *-Cmdlets* parameter for *Search-AdminAuditLog* to run the search.

```
[PS] C:\> Search-AdminAuditLog -Cmdlets Add-MailboxPermission

RunspaceId         : f6553abe-9d57-40bc-8e43-dc919bea2b50
ObjectModified     : exchange2013demo.com/ExchangeUsers/Alannah.Shaw
CmdletName         : Add-MailboxPermission
CmdletParameters   : {User, AccessRights, Identity}
ModifiedProperties : {}
Caller             : exchange2013demo.com/Users/Bob.Helpdesk
ExternalAccess     : False
Succeeded          : True
Error              :
RunDate            : 22/09/2015 4:35:33 PM
OriginatingServer  : SYDEX2 (15.00.1076.011)
Identity           : AAMkADI1NGQyZjhiLTFkYTAtNDhmYy05OTBiLTU4MGZlODY0MDQ3NgBGAAAAAAAkqZy/nl4
                     jSa4VBIka73bMBwCEoBRTwPA6QKt9HgzDn/p6AAAAAAEYAACEoBRTwPA6QK
                     t9HgzDn/p6AACUBtrhAAA=
IsValid            : True
ObjectState        : New
```

Another approach for the same scenario would be to look for modifications to the object "Alannah.Shaw" by using the *-ObjectIds* parameter. In this example we see exactly the same result, but you can imagine that other modifications may have been made to the same object and that multiple log entries would appear in many real world environments.

```
[PS] C:\>Search-AdminAuditLog -ObjectIds Alannah.Shaw

RunspaceId         : f6553abe-9d57-40bc-8e43-dc919bea2b50
ObjectModified     : exchange2013demo.com/ExchangeUsers/Alannah.Shaw
CmdletName         : Add-MailboxPermission
CmdletParameters   : {User, AccessRights, Identity}
ModifiedProperties : {}
Caller             : exchange2013demo.com/Users/Bob.Helpdesk
ExternalAccess     : False
Succeeded          : True
Error              :
RunDate            : 22/09/2015 4:35:33 PM
OriginatingServer  : SYDEX2 (15.00.1076.011)
Identity           : AAMkADI1NGQyZjhiLTFkYTAtNDhmYy05OTBiLTU4MGZlODY0MDQ3NgBGAAAAAAAkqZy/nl
                     4jSa4VBIka73bMBwCEoBRTwPA6QKt9HgzDn/p6AAAAAAEYAACEoBRTwP
                     A6QKt9HgzDn/p6AACUBtrhAAA=
IsValid            : True
ObjectState        : New
```

Searches can be limited to specific date ranges. Here's how to search for modifications made by "Bob.Helpdesk" in the last 30 days.

```
[PS] C:>  Search-AdminAuditLog -UserIds Bob.Helpdesk -StartDate (Get-Date).AddDays(-30)
```

If Bob has been doing his job, it's likely that a lot of results will be returned by that search, most of which should be perfectly normal for a person such as Bob on the help desk. To focus the search a little more, we can look at just the objects that Bob has modified in the last 30 days.

```
[PS] C:\>$logentries = Search-AdminAuditLog -UserIds Bob.Helpdesk -StartDate (Get-Date).AddDays(-30)

[PS] C:\>$logentries.ObjectModified
SYDEX2
SYDEX2\mapi (Default Web Site)
SYDEX2\OAB (Default Web Site)
SYDEX2\EWS (Default Web Site)
SYDEX2\Microsoft-Server-ActiveSync (Default Web Site)
SYDEX2\ecp (Default Web Site)
SYDEX2\owa (Default Web Site)
```

```
SYDEX2\Rpc (Default Web Site)
SYDEX1
SYDEX1\mapi (Default Web Site)
SYDEX1\OAB (Default Web Site)
SYDEX1\EWS (Default Web Site)
SYDEX1\Microsoft-Server-ActiveSync (Default Web Site)
SYDEX1\ecp (Default Web Site)
SYDEX1\owa (Default Web Site)
SYDEX1\Rpc (Default Web Site)
```

Looks like Bob has been messing with virtual directories. Let's make it even more useful and look at the time stamp, cmdlet, and objects modified by Bob in the last 30 days.

```
[PS] C:\>$logentries | Select RunDate,CmdletName,CmdletParameters,ObjectModified

RunDate                  CmdletName                   CmdletParameters              ObjectModified
-------                  ----------                   ----------------              --------------
27/08/2015 11:51:35 AM   Set-ClientAccessServer       {AutoDiscoverServiceIntern... SYDEX2
27/08/2015 11:51:33 AM   Set-MapiVirtualDirectory     {ExternalUrl, InternalUrl,... SYDEX2\mapi
(Default Web S...
27/08/2015 11:51:19 AM   Set-OabVirtualDirectory      {ExternalUrl, InternalUrl,... SYDEX2\OAB (Default
Web Site)
27/08/2015 11:50:51 AM   Set-WebServicesVirtualDire... {ExternalUrl, InternalUrl,... SYDEX2\EWS (Default
Web Site)
27/08/2015 11:49:55 AM   Set-ActiveSyncVirtualDirec... {ExternalUrl, InternalUrl,... SYDEX2\Microsoft-
Server-Ac...
27/08/2015 11:48:56 AM   Set-EcpVirtualDirectory      {ExternalUrl, InternalUrl,... SYDEX2\ecp (Default
Web Site)
27/08/2015 11:47:39 AM   Set-OwaVirtualDirectory      {ExternalUrl, InternalUrl,... SYDEX2\owa (Default
Web Site)
27/08/2015 11:46:13 AM   Set-OutlookAnywhere          {DefaultAuthenticationMeth... SYDEX2\Rpc (Default
Web Site)
27/08/2015 11:46:02 AM   Set-ClientAccessServer       {AutoDiscoverServiceIntern... SYDEX1
27/08/2015 11:46:02 AM   Set-MapiVirtualDirectory     {ExternalUrl, InternalUrl,... SYDEX1\mapi
(Default Web S...
27/08/2015 11:46:01 AM   Set-OabVirtualDirectory      {ExternalUrl, InternalUrl,... SYDEX1\OAB (Default
Web Site)
27/08/2015 11:45:59 AM   Set-WebServicesVirtualDire... {ExternalUrl, InternalUrl,... SYDEX1\EWS (Default
Web Site)
27/08/2015 11:45:57 AM   Set-ActiveSyncVirtualDirec... {ExternalUrl, InternalUrl,... SYDEX1\Microsoft-
Server-Ac...
27/08/2015 11:45:53 AM   Set-EcpVirtualDirectory      {ExternalUrl, InternalUrl,... SYDEX1\ecp (Default
Web Site)
27/08/2015 11:45:47 AM   Set-OwaVirtualDirectory      {ExternalUrl, InternalUrl,... SYDEX1\owa (Default
Web Site)
27/08/2015 11:45:40 AM   Set-OutlookAnywhere          {DefaultAuthenticationMeth... SYDEX1\Rpc (Default
Web Site)
```

As you can see administrator audit logging contains a lot of valuable information to help you identify who has been making changes in your Exchange organization, which could help you in many different troubleshooting scenarios. You can also see why it is important to limit administrative rights to only the minimum that each IT team member needs to do their job.

Additional reading

- Mailbox Audit Logging
- Understanding Role Based Access Control

Conclusion

While any single book certainly cannot cover every possible troubleshooting scenario in the realm of Exchange, we hope we've given enough foundational knowledge to help you in effectively navigating the hurdles you may encounter with the product.

All of us have worked with front-line Support Agents. The good agents are those who ask the right questions, gather the right data, communicate effectively, and don't make the situation worse than when the problem was discovered. Hopefully, by reading this book, you can gain some insight into the skills required to master each phase that leads to a successful resolution of a problem and maybe even some that help in a situation you cannot resolve. By acquiring knowledge about a problem and some inkling into its root cause, you will be in a better position to hand the issue over to Microsoft support or whomever will take over the case.

With that thought in mind, we bid you good luck in your troubleshooting efforts. Be patient, effective, and be positive. Also, as previously mentioned in the Introduction, don't be a Troubleblaster. Please!

We'd love to get your feedback on the book, including any topics you feel we should cover. Please send all feedback to feedback@exchangeserverpro.com.

Helpful companion material for Exchange and Office 365 can be found at http://exchangeserverpro.com/ebooks/.