9 Mail Flow - Compliance

In This Chapter

- Message Hygiene
- Data Loss Prevention
- Journaling
- Rights Management

In the previous chapter we covered some of the basics of the SMTP protocol in Exchange 2016 and how we can work with it in PowerShell. This chapter will cover the more advanced components of SMTP in Exchange 2016 – message hygiene, Data Loss Prevention (DLP), journaling and Rights Management. Some of the things that will be configured or managed with PowerShell may require additional licensing in Exchange. Some of these features are considered premium and will require an enterprise user CAL to be properly licensed to use the feature.

This chapter covers enterprise level features that are more likely to be used by larger messaging environments. Generally larger environments dictate more strict compliance requirements rather than smaller environments. Legal departments tend to be larger and more structured with policies in place for protecting all forms of communication and email is heavily regulated.

DLP is an interesting feature that was introduced into Exchange Server 2013 and continued in Exchange Server 2016. This features allows for more advanced transport rules for processing emails containing potentially sensitive information in them. Additional knowledge of Regular Expressions (RegEx) and compliance regulations may be needed in order to make the most of this feature. RegEx allows for more complex transport criteria.

The journaling feature is a feature commonly used in Exchange. Typically journaling is used for compliance, business continuity or discovery purposes. Messages can be journaled locally or externally depending on the need. Best practices for journaling will be covered as well.

Rights Management is a particularly interesting feature that also requires outside servers to make the feature work with Exchange. While Rights Management will be covered with respect to Exchange 2016 the build-out of the Rights Management infrastructure will not be covered detail. Sample diagrams will be provided. Some configuration tips will also be included for Rights Management, but no detailed installation or configuration will be provided.

Lastly, message hygiene, covered in the previous chapter, will be covered more in this chapter. Best practices for this feature in Exchange 2016 will be covered as well.

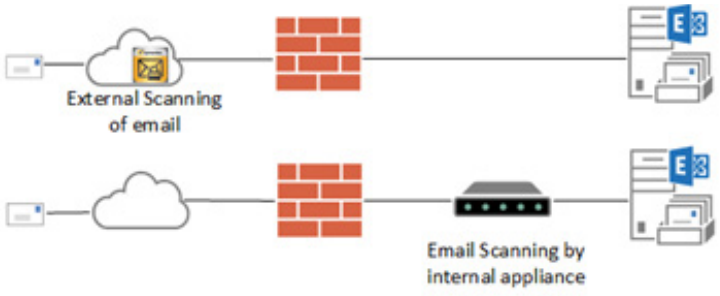
Message Hygiene

Message hygiene is a nebulous topic, but one of importance from an administration and user perspective. For administrators reducing the amount of spam or malware that enters Exchange makes for less supports calls, fixing or troubleshooting issues with email. For the end user, a reduced amount of bad messages makes for a better experience. In Chapter 8, as part of this books coverage of message hygiene, the Edge Transport Role was covered and this role includes many message hygiene features. This chapter will cover this topic a bit more with a brief explanation of external controls for message hygiene, agents on the mailbox servers, malware filter and finally managing it all with PowerShell.

External Controls

In order to block spam prior to the message being delivered to any Exchange Mailbox or Edge Transport servers, there are some additional things we can enable to help control message hygiene.

- **DKIM** – Allows a recipient of a message to verify the sender of the message. Requires a third party product like PowerMTA, DkimX or a DKIM Transport / Signing Agent. Signs the email with a digital signature that is verifiable with via the signers public key.
- **DMARC** – A Special TXT DNS record used by servers on how to handle DKIM or SPF failures.
- **SPF Record** – A special DNS record that provides a list of valid SMTP servers for your domain.
- **Message hygiene appliances or services** – There are many third party vendors available for message hygiene.



** **Note** ** None of the above can be configured with PowerShell, but these items should be on your list to begin protecting Exchange 2016. DKIM and DMARC require outside products and are not native to Exchange 2016.

Mailbox Server Agents

Like the Edge Transport Role, the Mailbox server role also has agents for message hygiene. The agents are not as numerous as what the Edge Transport Role has on them. The most important missing agent is the Connection Filtering agent which prevents the use of Real-time Black Lists (RBL) which are most effective in message hygiene. The agents on mailbox role servers are not enabled by default, been the case with previous versions of Exchange. The agents that are available on this role are:

- Sender Filter agent
- Sender ID agent
- Content Filter agent
- Protocol Analysis agent (sender reputation)

Enabling these agents requires a pre-built Microsoft PowerShell script:

```
& $env:ExchangeInstallPath\Scripts\Install-AntiSpamAgents.ps1
```

After the script is run, the Transport service needs to be restarted to initiate the newly configured agents.

```
Restart-Service MSExchangeTransport
```

Once that is complete, make sure to change the global configuration settings for the Exchange Transport to include all internal mail servers so as to not have any messages blocked by the newly configured agents. For example, if you have three internal SMTP servers that need to bypass the agents and they have IP Addresses of 172.20.1.55, 172.20.1.56 and 172.20.1.57:

```
Set-TransportConfig -InternalSMTPServers @{Add="172.20.1.55",”172.20.1.56”,”172.20.1.57"}
```

Once the settings are configured, verify them with a Get-TransportConfig cmdlet:

```
Get-TransportConfig | fl InternalSMTPServers
```

Managing Transport Agents

If there is no Edge Transport server present, then configuring these agents on the Mailbox servers is one option. All of these agents were detailed with PowerShell cmdlets in Chapter 8, the agents can be configured the same way. See page 179 for configuring these agents. That being said, the agents that can be configured on the Mailbox server only are not as effective as what is provided to the Edge Transport server. A better option would be to use a third party appliance or service to handle these features more effectively.

Data Loss Prevention

Data Loss Prevention (DLP) is a growing feature request among many types of organizations. The draw for these companies is that the DLP provides another line of defense for loss of corporate sensitive data. Key among features built into Exchange Server's DLP are the predefined sensitive data types provided by Microsoft and that DLP can be customized for an environment with templates and policy tips. For the end user, DLP is invisible for some scenarios, for example when administrators are testing rules, when a DLP policy stops a message from either reaching the end user or exiting Exchange DLP is only visible when a Policy Tip is configured to make the end user aware of the information that was being sent out.

** **Note** ** DLP is a premium feature of Exchange Server 2016 and requires an Enterprise CAL.

Features of DLP

- 80+ Sensitive Data Types
- Policy Tips – for OWA and Outlook
- Document Fingerprinting
- Coordinates with Transport Rules
- Customization – Templates
- 'Test Mode' – without affecting users

DLP PowerShell

First, we'll start with the cmdlets that are available for DLP:

```
Get-Command *DLP*
```

```
CommandType     Name
-----------     ----
Function        Export-DlpPolicyCollection
Function        Get-DlpPolicy
Function        Get-DlpPolicyTemplate
Function        Import-DlpPolicyCollection
Function        Import-DlpPolicyTemplate
Function        New-DlpPolicy
Function        Remove-DlpPolicy
Function        Remove-DlpPolicyTemplate
Function        Set-DlpPolicy
```

Exchange Server 2016 has no DLP policies defined by default. With a brand new installation of Exchange 2016 this can be verified with 'Get-DLPPolicy'. The same cmdlet can also be used later for verifying DLP policies.

DLP Templates

DLP Templates are one of the building blocks for DLP in Exchange Server 2016. To begin with the process, first use a template that is custom created or a Microsoft template. A DLP Policy is built based on that Template and then the DLP Policy is used in a Transport Rule.

With Exchange Server 2016, Microsoft has included a few DLP Templates to speed up deployment of DLP:

```
Get-DlpPolicyTemplate | ft –Auto
```

Creating Custom DLP Templates can be done with PowerShell or with an XML editor as there is no option to do so in the EAC. Whichever way the template is created it can be imported into Exchange with PowerShell. Looking at the cmdlets above, the Import-DLPPolicyTemplate looks like the cmdlet to do the job. What cmdlet examples are there:

```
Get-Help Import-DLPPolicyTemplate -Examples
```

```
-------------------------- Example 1 --------------------------

This example imports the DLP policy template file C:\My Documents\External DLP Policy
Template.xml.

Import-DlpPolicyTemplate -FileData ([Byte[]]$(Get-Content -Path "C:\My Documents\External DLP
Policy Template.xml" -Encoding Byte -ReadCount 0))
```

From the above example, we see that an XML file is needed in order to create/import a DLP Template into Exchange. Creating an XML file takes a bit of time and is somewhat complicated. These XML files can be created with a PowerShell script that has a series of questions. This script was written by one of the authors of the book and can be found here:

https://justaucguy.wordpress.com/2015/01/27/dlp-custom-xml-generation-script/

In practical terms, creating a one-off XML file is easier if you can use the Microsoft help and TechNet pages that are provided. Skipping forward, assuming an XML file has been created (BigBox-PII.XML) we can import the template using the example above for guidance.

```
Import-DlpPolicyTemplate -FileData ([Byte[]]$(Get-Content -Path "C:\DLPTemplate\BigBox-PII.xml"
-Encoding Byte -ReadCount 0))
```

Once the template is created, we can proceed to the creating of a DLP Policy based off of this template. There is no

real limit to the number of templates that can be created. The advantage of a custom XML for a custom template is that a RegEx query can be used to query for custom criteria – bank account numbers, custom card numbers, etc. The process for DLP rules is the same from here on whether the XML file is a custom or predefined template.

DLP Policies

DLP policies are built off of either the built in templates provided by Microsoft (see above) or custom templates like the one created in the example on the previous page. Transport Rules use DLP policies as matching criteria for SMTP messages traversing through an Exchange Server. Pre-canned templates exist for more common data types (financial data for Canada, UK or the US).

Creating a new DLP Policy with PowerShell requires the 'New-DLPPolicy' cmdlet. Here is an example of the cmdlet:

```
Get-Help New-DLPPolicy –Examples
-------------------------- Example 1 --------------------------

This example creates a new DLP policy named Contoso PII with the following values:

The DLP policy is enabled and set to audit only.

The DLP policy is based on the existing "U.S. Personally Identifiable Information (PII) Data"
DLP policy template.
New-DlpPolicy -Name "Contoso PII" -Template "U.S. Personally Identifiable Information (PII)
Data"
```

Other options that are available for configuring a new DLP Policy that should be considered are:

- **Mode <Audit | AuditAndNotify | Enforce>** - How the policy notifies the end users
- **State <Enabled | Disabled>** - Policies are enabled by default

Sample Policy creations of these two new DLP policies will be based off of an existing Microsoft templates ("Japan Financial Data") and a template we created in the previous section ("Big Box PII"):

```
New-DlpPolicy -Name "Big Box Personal Info" -Template "Big Box PII"
New-DlpPolicy -Name "Japanese Subsidiary Finance Data" -Template "Japan Financial Data"
```

The Japanese DLP Policy did generate a notification when it was created:

```
WARNING: The rule contains NotifySender action with an option that may reject the message. In case the
 message gets rejected, other actions won't be applied.
```

Once created, verifying the policies is the next step:

```
Get-DLPPolicy | ft -Auto
Name                              Publisher      State   Mode
----                              ---------      -----   ----
Japanese Subsidiary Finance Data Microsoft       Enabled Audit
Big Box PII                       Data Big Box   Enabled Audit
```

Now there are two DLP policies that can be called by Transport Rules to affect messages that meet the policy's criteria.

Policy Tips

Policy Tips are like any other Exchange Server Tips (MailTips is one example) that provide a visual indicator of a problem or something that the end user (the message sender) should be aware of. DLP policy tips will work in

either Outlook or OWA. For Outlook, make sure that the latest version of Outlook 2013 or 2016 are used in order to get the most out of the tips. Outlook can cache DLP information and any changes that are made may show up immediately. Previous versions of Outlook do not work with Policy Tips, nor a standalone installation of Outlook.

** **Note** ** Policy tips do not work if the full Office Suite is also not installed. Standalone Outlook will not work with policy tips - https://support.microsoft.com/en-us/kb/2823263.

With regards to PowerShell and Policy Tips, the wording of the tip can be customized and the tip can be turned on or off depending on the scenario. For example, a DLP Policy, tied to a Transport Rule that looks for sensitive data, can be flagged for 'Testing with no Policy Tips'. To configure the rule, we need to look at options for this rule ('mode' parameter):

Get-Help Set-TransportRule –Full

```
-Mode <Audit | AuditAndNotify | Enforce>
    The Mode parameter specifies in which mode this rule will operate. Valid values include:

    * Audit The rule is turned on, and what would have happened if the rule was enforced is
      logged in message tracking logs. Exchange doesn't take any action that impacts the
      delivery of the message.
    * AuditAndNotify The rule is turned on, and it operates the same way it would in Audit
      mode, but notifications are also enabled.
    * Enforce The rule is turned on, and all actions specified in the rule are taken.
    The default value is Enforce.

    Required?                    false
    Position?                    Named
    Default value
    Accept pipeline input?       False
    Accept wildcard characters?  false
```

Using the switch '-Mode Audit' and no Policy Tips would be visible to the end user. For either '-Mode AuditAndNotify' or '-Mode Enforce', policy tips would be visible in the mail client if a message matches the rule (and the associated DLP Policy). Policy Tips can also be customized. If, instead of the pre-canned tips, there is a need for a custom message for end users, these can be done with PowerShell. First, what cmdlets are available:

Get-Command *PolicyTip*

```
CommandType     Name
-----------     ----
Function        Get-PolicyTipConfig
Function        New-PolicyTipConfig
Function        Remove-PolicyTipConfig
Function        Set-PolicyTipConfig
```

First, we start with what is already configured for Policy Tips:

Get-PolicyTipConfig

As expected, no results are to be found. We will need to create our own set of Policy Tips to notify end users with these custom messages.

Get-Help New-PolicyTipConfig –Examples

The most important parameter is '-Value' as it determines what the end message will be provided to the end user. Notice that the second example provides a URL for the end user. This might be more appropriate if there is a fully fleshed out policy for these restricted attachments or PII. When creating a new Policy Tip, the name of the tip needs to reference two criteria: the locale and the action to be performed. A working example for the English language would be:

-Name "en\NotifyOnly"

If for example the wrong name is chosen, then a lot of errors are generated:

```
Name must be in the form locale\action where action can be: "zh-CHS, en, fr, de, ja, zh-CHT, it, ko, pt, ru, es, ar,
cs, da, nl, fi, el, he, hu, no, pl, pt-PT, sv, tr, ro, th, fil-PH, hi, id, lv, ns, uk, vi, bg, hr, et, lt, sr, sk, sl,
eu, ca, zh-HK, fa, gl, is, kk, sr-Cyrl-CS, ur, af, sq, am-ET, hy, as-IN, bn-IN, bn-BD, bs-Cyrl-BA, bs-Latn-BA, ka, gu,
ha-Latn-NG, ig-NG, iu-Latn-CA, ga-IE, xh-ZA, zu-ZA, kn, km-KH, qut-GT, rw-RW, sw, kok, ky, lo-LA, lb-LU, mk, ms-BN,
ml-IN, mt-MT, mi-NZ, mr, ne-NP, nn-NO, or-IN, ps-AF, pa, quz-PE, nso-ZA, tn-ZA, si-LK, ta, tt, te, uz, cy-GB, wo-SN,
yo-NG" and locale can be: "NotifyOnly, RejectOverride, Reject, Url". If action is URL, then name must be "Url" with
no locale.
    + CategoryInfo          : InvalidArgument: (:) [New-PolicyTipConfig], NewPolicyTipConfigInvalidNameException
    + FullyQualifiedErrorId : [Server=16-TAP-EX02,RequestId=c29978a8-adbe-4cc1-b881-30fd2474b531,TimeStamp=8/30/2016 2
   :16:59 AM] [FailureCategory=Cmdlet-NewPolicyTipConfigInvalidNameException] D715A4C9,Microsoft.Exchange.Management.
  PolicyNudges.NewPolicyTipConfig
    + PSComputerName        : 16-tap-ex02.16-tap.local
```

A complete cmdlet would look like this:

> New-PolicyTipConfig -Name en\NotifyOnly -Value 'This message contains private information that should not be shared outside of this company.'

To verify the cmdlet worked:

> Get-PolicyTipConfig | ft –Auto

```
Identity      Value
--------      -----
en\NotifyOnly This message contains private information that should not be shared outside of thi...
```

Other possible actions are RejectOverride and Reject. Reject will block the message completely whereas RejectOverride allows for an override if the user puts "Override" in the subject line of the message.

> New-PolicyTipConfig -Name en\RejectOverride -Value 'This message contains private information that should not be shared outside of this company.'

Document Fingerprinting

DLP's Document Fingerprinting feature allows for DLP to search for very specific content that is sent through e-mail, in this case a matching attachment. The fingerprint is basically a hash of the properties of the document in question. The document itself is not stored in Exchange. Outlook will also evaluate a document locally and the document is not sent over the wire between Exchange and Outlook. The process is similar to creating a template, with the source being a document to import. Then build Transport Rules around the document to restrict, allow or log when a rule processes the e-mail. Below is a scenario which will provide a better idea as to what can be done with this, and what PowerShell can provide.

Scenario

HR has some confidential forms that are to be used internally by the company. They've provided IT with three forms that need to prevent from being emailed to anyone external to the organization. First, place a copy of the document on the Exchange server so that it can be imported for creating the DLP Policy. Any form or document to be 'fingerprinted' should be blank so that no information interferes with the evaluation.

For PowerShell cmdlets, start with Get-Content (used to store the file in a variable) and then use New-FingerPrint to create the fingerprint based off the content from the Get-Content variable. Follow this by creating a new Data Classification to be used by Transport Rules later.

There are three forms to be protected:

- EmployeePII-Form.docx
- Employee-Review-2016.docx
- Termination-RequestForm.docx

Next, store the document content in a variable in preparation for Transport Rules to use the content. Let's walk through the process of taking these documents and creating Transport Rules to handle them:

Import each individual document into a separate variable to be used by New-Fingerprint:

```
$HRDoc1 = Get-Content "C:\Documents\HR\EmployeePII-Form.docx" -Encoding Byte
$HRDoc2 = Get-Content "C:\Documents\HR\Employee-Review-2016.docx" -Encoding Byte
$HRDoc3 = Get-Content "C:\Documents\HR\Termination-RequestForm.docx" -Encoding Byte
```

Notice that the documents are encoded as a 'byte' type document. According to the help file on the 'Get-Help' cmdlet, there are a few data types that can be used:

```
ASCII, BigEndianUnicode, Byte, String, Unicode, UTF7, UTF8 and Unknown.
```

In choosing a Word document (which is a binary file) we need to choose 'byte' for the encoding to properly ingest the hash from the file. The 'Get-Content' cmdlet does have other parameters, but for the purposes of fingerprinting itself, no others are required. Simply put in a location of the file and what encoding to use for the document for fingerprinting and store that in a variable.

Create a fingerprint based off the document stored in each variable:

```
$HRDoc1_Fingerprint = New-Fingerprint -FileData $HRDoc1 -Description "Employee PII Form"
$HRDoc2_Fingerprint = New-Fingerprint -FileData $HRDoc2 -Description "Employee Review 2016"
$HRDoc3_Fingerprint = New-Fingerprint -FileData $HRDoc3 -Description "Termination Request Form"
```

** No cmdlet can query fingerprints that were created, to see the fingerprints raw data, you can simply 'dump' the variable contents to the PowerShell window:

```
$HRDoc1_Fingerprint | fl
```

The New-Fingerprint cmdlet has even less options than the Get-Content cmdlet and examples from the cmdlet use only the two parameters chosen above – FileData and Description. FileData references the document stored in the variable.

Now that the Fingerprint has been created, it can be used by the New-DataClassification cmdlet to create a data classification for a Transport Rule:

```
New-DataClassification -Name "HR Confidential Form 1" -Fingerprints $HRDoc1_Fingerprint
-Description "Message contains confidential employee information."

New-DataClassification -Name "HR Confidential Form 2" -Fingerprints $HRDoc2_Fingerprint
-Description "Message contains confidential employee information."

New-DataClassification -Name "HR Confidential Form 3" -Fingerprints $HRDoc3_Fingerprint
-Description "Message contains confidential employee information."
```

The New-DataClassification cmdlet can be used to create individual classifications or it can group multiple Fingerprints together into one classification as the parameter used for this is 'Fingerprints' not 'Fingerprint'. Make sure to separate multiple Fingerprints with a comma.

** **Note** ** Document fingerprints can also be added to existing data classifications using the Set-DataClassification cmdlet and the –Fingerprints parameter:

```
Set-DataClassification -Name "HR Confidential Form 3" -Fingerprints $HRDoc3_Fingerprint
```

To verify the Fingerprints were successful in being converted to an Exchange Data Classifications, run the following:

```
Get-DataClassification
```

```
Name                                LocalizedName                       Publisher             ClassificationType
----                                -------------                       ---------             ------------------
Croatia Identity Card Number        Croatia Identity Card Number        Microsoft Corporation             Entity
Czech National Identity Card Number Czech National Identity Card Number Microsoft Corporation             Entity
Greece National ID Card             Greece National ID Card             Microsoft Corporation             Entity
South Africa Identification Number  South Africa Identification Number  Microsoft Corporation             Entity
HR Confidential Form 1              HR Confidential Form 1              16-TAP                        Fingerprint
HR Confidential Form 2              HR Confidential Form 2              16-TAP                        Fingerprint
HR Confidential Form 3              HR Confidential Form 3              16-TAP                        Fingerprint
```

Notice the header fields of Invariant Name, Localized Name, Publisher and Classification Type. The other Data Classification entries show the Publisher to be Microsoft and the Classification Type to be Entity. Can we change ours to something more meaningful? First, what other cmdlets are available for Data Classifications:

```
Get-Command *DataClass*
```

```
Name                                Category
----                                --------
Get-DataClassification              Cmdlet
New-DataClassification              Cmdlet
Remove-DataClassification           Cmdlet
Set-DataClassification              Cmdlet
Test-DataClassification             Cmdlet
```

Upon reviewing the parameters for these cmdlets reveals that this cannot be changed. The Classification type is set once a Fingerprint is used. The publisher simply matches the name of the server it was created on.

Continuing on the Fingerprints are created and stored as new Data Classifications. This Data Classification can be used by a Transport Rule to block these emails (and their attachments) from leaving Exchange.

New-TransportRule -Name ‘Notify :External Recipient BigBox confidential’ -RejectMessageReasonText 'This file is restricted and may not be emailed outside the company.' -NotifySender $Null -Mode Enforce -SentToScope NotInOrganization -MessageContainsDataClassification @{Name=’ HR Confidential Form 1’}

New-TransportRule -Name ‘Notify :External Recipient BigBox confidential #2’ -RejectMessageReasonText 'This file is restricted and may not be emailed outside the company.' -NotifySender $Null -Mode Enforce -SentToScope NotInOrganization -MessageContainsDataClassification @{Name=’ HR Confidential Form 2’}

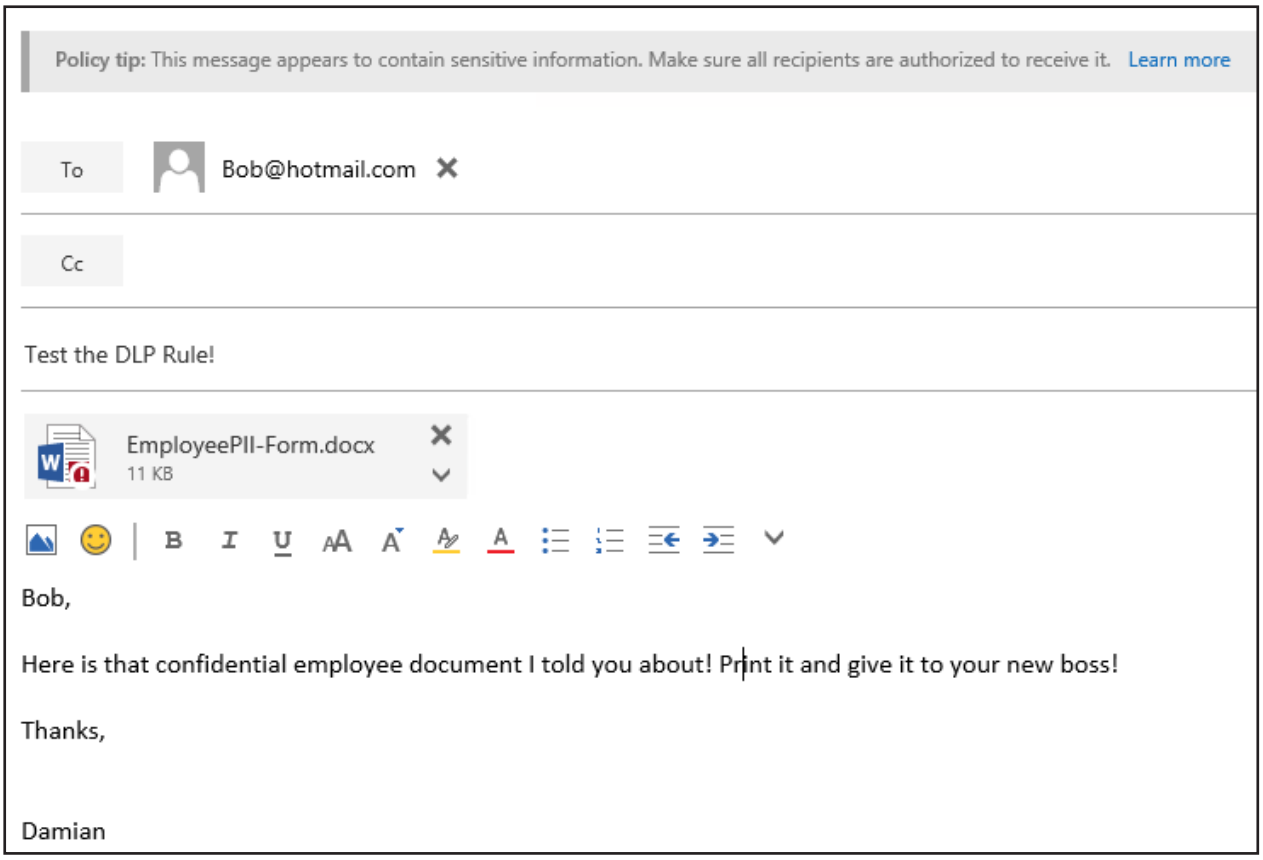
New-TransportRule -Name ‘Notify :External Recipient BigBox confidential #3’ -RejectMessageReasonText 'This file is restricted and may not be emailed outside the company.' -NotifySender $Null -Mode Enforce -SentToScope NotInOrganization -MessageContainsDataClassification @{Name=’ HR Confidential Form 3’}

Verify Transport Rules were created:

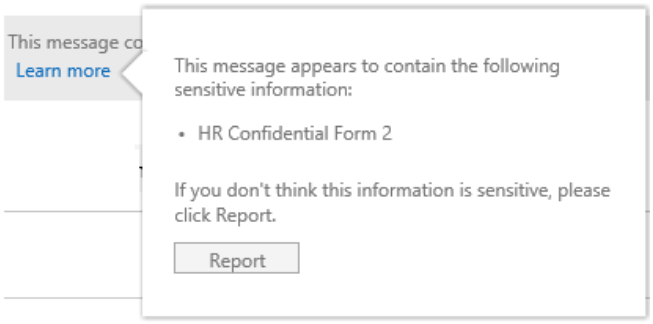
Get-TransportRule

```
Name                                          State    Mode         Priority Comments
----                                          -----    ----         -------- --------
Notify :External Recipient BigBox confide...  Enabled  Enforce      3
Notify :External Recipient BigBox confide...  Enabled  Enforce      4
Notify :External Recipient BigBox confide...  Enabled  Enforce      5
```

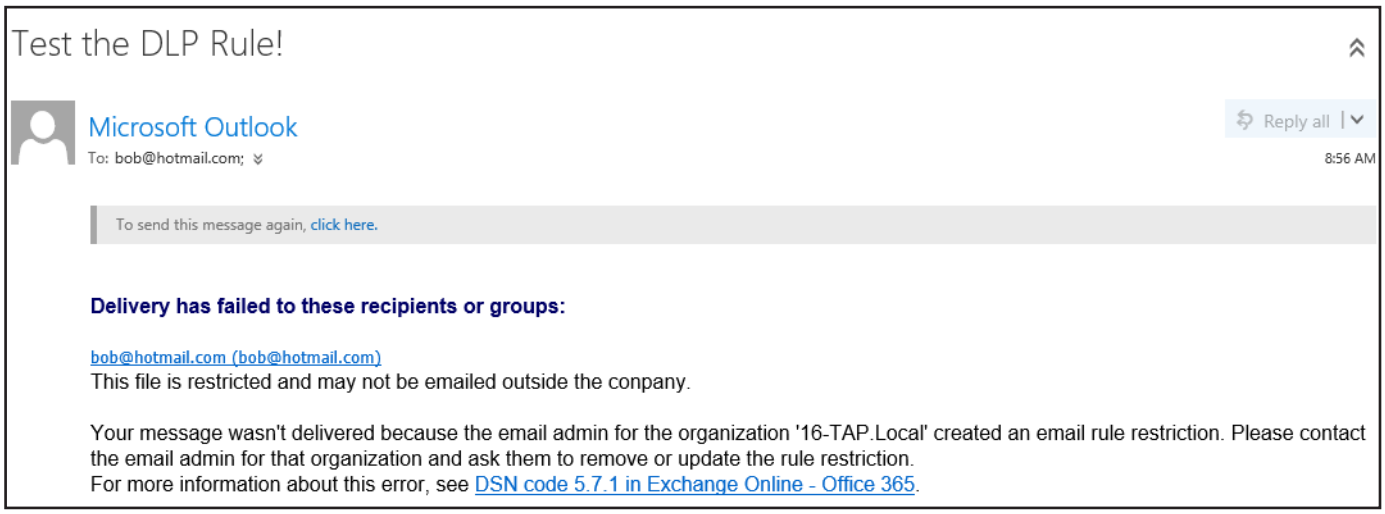
How does this work in practice? First, a message created in OWA is created with one of the three HR documents attached and addresses to an external recipient:



Notice the Policy Tip at the top of the email as well as the fact that the document itself has a red exclamation point to indicate that it has been recognized by its content. If a user is unsure why this is occurring, there is a 'Learn More' button at the top which provides this tidbit (as well as an option for feedback). The option show is based on which Action was chosen in the Policy Tip.



After the message is sent, an NDR is generated:



Going back to PowerShell, how can these DLP flagged messages be tracked? More importantly, how can a meaningful report be generated for management to show the effectiveness of the rules?

Examining the Message Tracking Logs, we need criteria to create a list of blocked messages. Taking the example above, start by reviewing all messages that were sent to 'bob@hotmail.com':

```
Get-MessageTrackingLog -Start "8/27/16" -Recipient Bob@hotmail.com | ft
```

```
EventId          Source      Sender                    Recipients
-------          ------      ------                    ----------
HAREDIRECT       SMTP        Administrator@16-t...  {Bob@hotmail.com}
RECEIVE          SMTP        Administrator@16-t...  {Bob@hotmail.com}
AGENTINFO        AGENT       Administrator@16-t...  {Bob@hotmail.com}
TRANSFER         ROUTING     Administrator@16-t...  {Bob@hotmail.com}
HARECEIVE        SMTP        Administrator@16-t...  {bob@hotmail.com}
HADISCARD        SMTP        Administrator@16-t...  {bob@hotmail.com}
```

Running the same cmdlet with '| fl' instead of '| ft –auto' reveals that there may be some useful information:

```
ServerIp                :
ServerHostname          : 16-TAP-EX02
SourceContext           : ExplicitlyDiscarded
ConnectorId             :
Source                  : SMTP
EventId                 : HADISCARD
InternalMessageId       : 5205500362875
```

However, using any criteria fails to provide sufficient information for finding just these messages. For example, filtering for an EventID of 'HADiscard' ends up with internal heath status information:

Timestamp	EventId	Source	Sender	Recipients
8/27/2016 12:00:48 AM	HADISCARD	SMTP	HealthMailbox5d415ab7450045518cea8ce583d3d9a3@16-TAP.Local	{HealthMailbox5d415ab74
8/27/2016 12:00:48 AM	HADISCARD	SMTP		{HealthMailboxa645272c1
8/27/2016 12:03:19 AM	HADISCARD	SMTP		{HealthMailboxee31399d0
8/27/2016 12:03:19 AM	HADISCARD	SMTP		{HealthMailbox5d415ab74
8/27/2016 12:05:51 AM	HADISCARD	SMTP	HealthMailbox5d415ab7450045518cea8ce583d3d9a3@16-TAP.Local	{HealthMailbox5d415ab74
8/27/2016 12:08:17 AM	HADISCARD	SMTP	HealthMailbox6dffd6acf4bb461cb3fa2a6a188eff24@16-tap.local	{HealthMailbox6dffd6acf
8/27/2016 12:08:17 AM	HADISCARD	SMTP		{HealthMailbox5d415ab74
8/27/2016 12:08:17 AM	HADISCARD	SMTP	HealthMailbox58d467eb6ad04993a08b5ebdfcd9c73f@16-tap.local	{HealthMailbox58d467eb6
8/27/2016 12:10:50 AM	HADISCARD	SMTP	HealthMailbox5d415ab7450045518cea8ce583d3d9a3@16-TAP.Local	{HealthMailbox5d415ab74

Hardly the information that we are looking for.

What other criteria can be used? There is a curious field near the bottom called 'EventData' and inside that field there are a few references to 'Rule' in the properties. The 'Rule' looks like a GUID, perhaps the ID of the rule we are looking for?

```
EventData       : {[AMA, SUM¦v=0¦action=¦error=¦atch=1], [AMA, EV¦engine=M¦v=0¦sig=1.227.840.0¦name=¦file=],
                  [TRA, ETR¦ruleId=8542dcbe-a919-4b16-bb80-a491395d4152¦st=8/15/2016 11:26:31
                  AM¦action=ApplyHtmlDisclaimer¦sev=1¦mode=Enforce], [TRA, ETRP¦ruleId=8542dcbe-a919-4b16-bb80-
                  a491395d4152¦st=2016-08-15T11:26:31.0000000Z¦ExecW=583¦ExecC=31¦Actions=ApplyHtmlDisclaimer,6
                  4¦Conditions=MatchesRegexPredicate,Message.Subject,133;MatchesRegexPredicate,Message.Body,279
                  ], [TRA, ETRP¦ruleId=00946fa5-09c3-4d2b-a30e-0165616068c3¦st=2016-08-17T01:31:14.0000000Z¦Exe
                  cW=779¦ExecC=46¦Conditions=MatchesRegexPredicate,Message.AttachmentNames,622¦Components=FIPS_
                  TE,0,158], [TRA, ETR¦ruleId=8a756e55-aea1-49dd-973e-84481ca555bd¦st=8/17/2016 11:16:18
                  PM¦action=ApplyHtmlDisclaimer¦sev=1¦mode=Enforce], [TRA, ETRP¦ruleId=8a756e55-aea1-49dd-973e-
                  84481ca555bd¦st=2016-08-17T23:16:18.0000000Z¦ExecW=4¦ExecC=15¦Actions=ApplyHtmlDisclaimer,3],
                   [TRA, DC¦dcid=77a80d36-ef82-4ff6-bab2-00d391165051¦count=1¦conf=75], [TRA,
                  DC¦dcid=a49fee1d-2443-46aa-bd32-75185b09161b¦count=1¦conf=75], [TRA,
                  ETR¦ruleId=4fca7be8-a5ee-4f61-93ca-1295e9c5cae5¦st=8/27/2016 2:21:57
                  AM¦action=NotifySender¦sev=1¦mode=Enforce¦dcid=77a80d36-ef82-4ff6-bab2-00d391165051], [TRA, E
                  TRP¦ruleId=4fca7be8-a5ee-4f61-93ca-1295e9c5cae5¦st=2016-08-27T02:21:57.0000000Z¦ExecW=5393¦Ex
                  ecC=46¦Conditions=ContainsDataClassificationPredicate,Message.DataClassifications,9;ContainsD
                  ataClassificationPredicate,Message.DataClassifications,5392;ContainsDataClassificationPredica
```

If that is true, we could use PowerShell to find the Transport Rule by the GUID [picking one from EventData (in red rectangles)]:

```
Get-TransportRule -Identity 4fca7be8-a5ee-4f61-93ca-1295e9c5cae5
```

Name	State	Mode	Priority	Comments
Notify :External Recipient BigBox confide...	Enabled	Enforce	3	...

The rule looks correct, and now a Message Tracking Log trace will be done with that criteria and a report created:

```
Get-MessageTrackingLog -Start "8/27/16" -Recipients bob@hotmail.com | Where {$_.EventData -Match
"4fca7be8-a5ee-4f61-93ca-1295e9c5cae5"}
```

Timestamp	EventId	Source	Sender	Recipients	MessageSubject
8/27/2016 6:39:26 AM	AGENTINFO	AGENT	Administrator@16-tap.local	{Bob@hotmail.com}	Test the DLP Rule!

The above is one of the messages that were caught by this Transport Rule.

What other ways can the Document Fingerprinting feature be used? Maybe in a set of documents that should never leave HR or be sent in email or some intellectual property documents (patent forms) that should never leave R&D or never even be sent through email. These scenarios can also be controlled via the Document Fingerprinting feature. Each scenario should be created with its own unique fingerprint, data classification and Transport Rules to keep track of each particular scenario in a company. This will make troubleshooting much easier if issues or discrepancies occur (document is updated or message is or is not delivered).

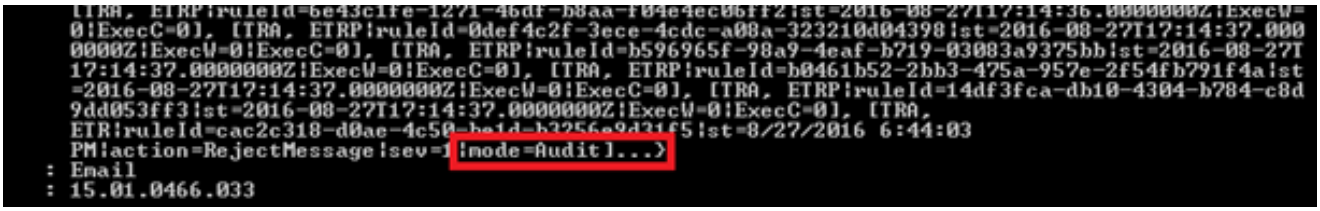
Test Mode

The advantage of test mode for DLP Policies, templates and rules is that an IT department can create the DLP policies and Transport Rules driven by Legal, HR, management, auditors and government regulation to test the policy without an end users knowledge. Doing so will allow IT to validate a rules effectiveness to test parameters as well as to validate a rules effect on the end users. The later could be done with an auditing report that allows IT to query what messages would have been touched by what Transport Rules and thus blocked or redirected or whatever action is desired. This allows for real time data collection of messages to determine the effectiveness.

To change the setting in PowerShell, use the following:

```
Get-TransportRule "Name of Rule" | Set-TransportRule -Mode Audit
```

With auditing in place, messages processed by this rule will have an EventID of 'AgentInfo' and have rules applied, but the message will not be rejected. Look for the rule with Mode=Audit, like so:



Now the rule can be tested with production and reports produces for the rule requestor.

Journaling

Journaling is the process of making a copy of an email and storing it in a location routed via an email address. A message can be journaled to a local database or an external service. Either method is supported using the Journaling rule cmdlets below. Messages can be journaled for compliance or for business continuity. Business continuity is one of the features external services tote as a reason to use their product.

PowerShell

For journaling, a small subset of cmdlets is available for managing Journal rules in Exchange Server 2016:

```
Get-Command *journ*
```

```
CommandType     Name
-----------     ----
Function        Disable-JournalRule
Function        Enable-JournalRule
Function        Export-JournalRuleCollection
Function        Get-JournalRule
Function        Import-JournalRuleCollection
Function        New-JournalRule
Function        Remove-JournalRule
Function        Set-JournalRule
```

By default, there are no Journal rules configured by default which can be confirmed by running 'Get-JournalRule' in a new Exchange Server environment. To begin exploring PowerShell journaling, create some Journaling rules using the 'New-JournalRule' cmdlet. See below for an example:

New-JournalRule

Commonly Used Options

JournalEmailAddress
Name - Name of the new Journal Rule
Enabled <$true | $false> - Whether the rule is enabled or not
Recipient <SmtpAddress>
Scope <Internal | External | Global>

- Global - Global rules process all email messages that pass through a Transport service.
- This includes email messages that were already processed by the external and internal rules.
- Internal - Internal rules process email messages sent to and received by recipients in your organization.
- External - External rules process email messages sent to recipients or from senders outside your organization.

Sample One-Liner

New-JournalRule -Name "Personal Data (US)" -JournalEmailAddress "US Journal Mailbox" -Scope Global -Recipient USJournaling@BigBox.Com -Enabled $True

Example Usage

As the email administrator of the legal department determined that with a new implementation of an Exchange 2016 server environment, all messages need to be journaled. The reason for the journaling is to comply with the current government regulations for email retention.

New-JournalRule -Name "Test" -JournalEmailAddress "JournalingMailbox@16-tap.local" -Scope Global -Enabled $True

Script Scenario

You work for a company with 12,000 users. There are offices all over the world with major concentrations of users in the US and Europe. There are different compliance requirements for the different regions.

All the users from Europe need to be journaled for new compliance regulations, separate from current archiving and business continuity of the US branch. The users are in three different countries, each on different Exchange servers - Poland, Italy and Germany. Each of these groups comprise of about 1,000 workers. The legal department wants each group to be journaled to a local database. Your IT manager wants separate rules and a way to track the configuration. HR provides a complete list of users to help verify that the correct employees are being journaled.

Here are the steps that need to be taken in order to make this possible:

- Assign user a custom attribute - this can be used for a query when assigning
- Create a journaling database on each server
- Turn on circular logging - reduces the size of the database on the disk
- Create mailbox local to the region - keeps traffic local
- Add the journal rule for each user with the custom attribute, which makes this a custom journal rule and thus requires an enterprise Client Access License (CAL)

Active Directory is not organized by geographical location, but by business unit, so trying to get lists of users by Organizational Unit (OU) will not provide valid results. Mailboxes may not be in the correct OU and users need to be tagged by country.

Knowing this we need some sort of reference point for the script to pull users from. What attribute on the user account and what attribute should be used to be queried later?

Take a look at the help for the Set-Mailbox cmdlet looking for valid parameters:

```
Get-Help Set-Mailbox -full
```

In the list of attributes that can be used is a set of custom attributes:

```
CustomAttribute1 to 15
```

The CustomAttribute1 to CustomAttribute15 parameters allow the configuration of custom attributes. You can use these attributes to store additional custom information.

For simplicity's sake, set the attribute value to a regional code:

1 for Germany
2 for Italy
3 for Poland

Using the list of users from the CSV file provided by HR, assign a region code to each mailbox in the CSV file. Like so:

```
$Csv = Import-Csv "c:\downloads\MailboxList.csv"

Foreach ($Line in $Csv) {
   Get-Mailbox $Line.Mailbox | Set-Mailbox -CustomAttribute1 $Line.Region
}
```

CSV File Format

```
Mailbox,Region
Damian,1
DStork,2
TestUser01,3
TUser02,1
```

Next, let's create a new mailbox database for journaling (see Chapter 7 of Server Management for cmdlets):

```
# Create Journaling database for the German Exchange Server
New-MailboxDatabase -Name "Journaling-Germany-Db" -Server SRV-GB-EX01 -EdbFilePath D:\
Databases\Journaling\Journaling.EDB -LogFolderPath E:\Logs\Journaling -IsExcludedFromProvisioning
$True -AutoDagExcludeFromMonitoring $True

# Create Journaling database for the Italian Exchange Server
New-MailboxDatabase -Name "Journaling-Italy-Db" -Server SRV-IT-EX01 -EdbFilePath D:\Databases\
Journaling\Journaling.EDB -LogFolderPath E:\Logs\Journaling -IsExcludedFromProvisioning $True
-AutoDagExcludeFromMonitoring $$True

# Create Journaling database for the Polish Exchange Server
New-MailboxDatabase -Name "Journaling-Poland-Db" -Server SRV-PO-EX01 -EdbFilePath D:\Databases\
Journaling\Journaling.EDB -LogFolderPath E:\Logs\Journaling -IsExcludedFromProvisioning $True
-AutoDagExcludeFromMonitoring $True
```

Options Chosen - Options below were chosen because the database is a journaling database
IsExcludedFromProvisioning - no new mailboxes are automatically added to the database – set this to $true
AutoDagExcludeFromMonitoring - suppresses error messages related to a database not having a copy in a DAG environment – set this to $true

Sample Results

```
Name                           Server          Recovery        ReplicationType
----                           ------          --------        ---------------
Journaling-Italy-Db            16-TAP-EX01     False           None
```

Good Reference
https://blogs.technet.microsoft.com/scottschnoll/2014/06/27/keeping-up-to-date-with-whats-happening-with-set-mailboxdatabase-ii/

After the databases are created, PowerShell can be used to enable circular logging and then dismount and remount the database. Then the script will provide an option to restart the Information Store service (best practice Microsoft).

Script

Breaking down the script requirements, the script functions on a simple yes/no basis. If the question to restart the service is 'y', then the service is restarted and if the answer is 'n' then the script notifies you that the service needs to be restarted. The reason for this setup is that if a database is created during the day you probably will not want to restart the IS service.

For the question, using 'read-host –prompt' allows for a text question and an answer right after.

The IF..ELSE code block handles the 'what to do' with the answer given. There is no built-in error checking, but the script is relatively simple:

```
# Restart IS
$Answer = Read-Host -Prompt "Do you wish to restart the Information Store at this point? [y or n]"
If ($Answer -eq 'y') {
   Get-Service MSExchangeIS | Restart-Service
} Else {
   Write-Host "Make sure to restart the Exchange Information Store for memory management."
   -ForegroundColor Cyan
}
```

```
[PS] C:\>.\Restart.ps1
Do you wish to restart the Information Store at this point? y or n: y
WARNING: Waiting for service 'Microsoft Exchange Information Store (MSExchangeIS)' to start...
WARNING: Waiting for service 'Microsoft Exchange Information Store (MSExchangeIS)' to start...
WARNING: Waiting for service 'Microsoft Exchange Information Store (MSExchangeIS)' to start...
[PS] C:\Downloads>.\Restart.ps1
Do you wish to restart the Information Store at this point? y or n: n
Make sure to restart the Exchange Information Store for memory management.
```

Create a mailbox on each regional database:

```
# Create Journaling mailbox on the German Exchange Server
New-Mailbox -Shared -Name "Journaling-Germany" -DisplayName "Journaling-Germany" -Alias
Journaling-Germany -Database "Journaling-Germany-Db"

# Create Journaling mailbox on the Italian Exchange Server
New-Mailbox -Shared -Name "Journaling-Italy" -DisplayName "Journaling-Italy" -Alias Journaling-Italy
-Database "Journaling-Italy-Db"

# Create Journaling mailbox on the Polish Exchange Server
New-Mailbox -Shared -Name "Journaling-Poland" -DisplayName "Journaling-Poland" -Alias Journaling-
Poland -Database "Journaling-Poland-Db"
```

These same one-liners can be simplified into a single line of code:

```
'Germany','Italy','Poland'|%{New-Mailbox -Shared -Name "Journaling-$_" -DisplayName "Journaling-$_"
-Alias "Journaling-$_" -Database "Journaling-$_-Db" }
```

What this line does is specify a series of values which when read in with the '%' (an alias for ForEach-Object) and places the value into the '$_' variable. Thus this one line will run three times, one for each value provided at the beginning of the line.

Sample Output

```
Name                 Alias                ServerName      ProhibitSendQuota
----                 -----                ----------      -----------------
Journaling-Germany   Journaling-Germany   16-tap-ex02     Unlimited
```

Create a new Journal Rule and Dynamic Distribution List (DDL) for emails to recipients from a certain region to a certain journaling database:

```
# Create Dynamic list of users and a journal rule that uses that group
New-DynamicDistributionGroup -Name GermanyJournaling -RecipientFilter {(CustomAttribute1 -eq
"1")}
$smtp = ((Get-DynamicDistributionGroup GermanJournaling).PrimarySmtpAddress).Address

New-JournalRule -Name "Journaling for Germany mailboxes" -JournalEmailAddress "Journaling-
Germany" -Scope Global -Recipient $Smtp -Enabled $True

# Create Dynamic list of users and a journal rule that uses that group
New-DynamicDistributionGroup -Name ItalyJournaling -RecipientFilter {(CustomAttribute1 -eq "2")}
$smtp = ((Get-DynamicDistributionGroup ItalyJournaling).PrimarySmtpAddress).Address

New-JournalRule -Name "Journaling for Italy mailboxes" -JournalEmailAddress "Journaling-Italy"
-Scope Global -Recipient $Smtp -Enabled $True

# Create Dynamic list of users and a journal rule that uses that group
New-DynamicDistributionGroup -Name PolandJournaling -RecipientFilter {(CustomAttribute1 -eq "3")}
$smtp = ((Get-DynamicDistributionGroup PolandJournaling).PrimarySmtpAddress).Address

New-JournalRule -Name "Journaling for Poland mailboxes" -JournalEmailAddress "Journaling-Poland"
-Scope Global -Recipient $smtp -Enabled $True
```

Sample Output

```
Name                  : Journaling for German Users
Recipient             : GermanJournaling@16-tap.local
JournalEmailAddress   : Journaling-Germany@16-tap.local
Scope                 : Global
Enabled               : True
```

With the above setup, any email that goes to a user who is in the Dynamic Distribution List for a particular country will have their messages journaled to a local journaling database.

Best Practices (and the PowerShell to configure them…)

When working with the journaling process there are a few things to remember for the configuration of Journaling in Exchange Server 2016 (and any other version of Exchange up to this point):

- Journal Recipient is hidden from the Global Address List (GAL)
- Journal mailbox is on its own database and the database is on separate disks
- Journal mailbox database should be excluded from automatic provisioning and excluded from DAG monitoring

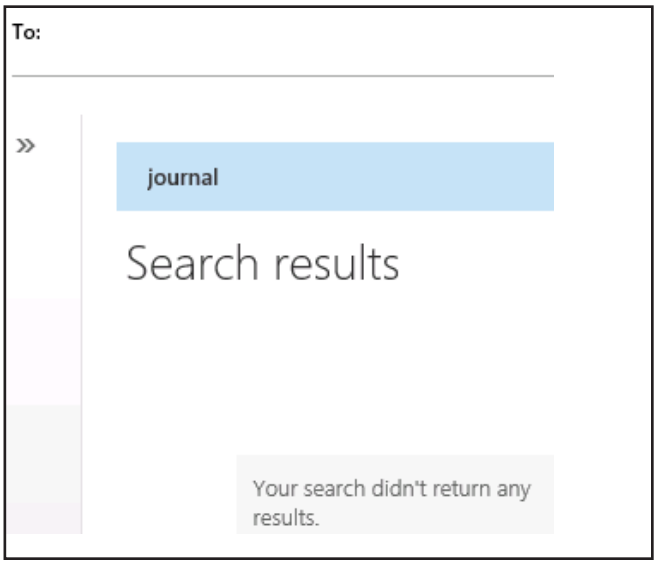
For the above best practices, items 2 and 3 were taken care of in the creation of the database and mailbox. A separate database was created, it was placed on its own disk drives (separate from other databases), it was excluded from automatic provisioning (-IsExcludedFromProvisioning $true) and the database will not generate errors about not having a secondary copy in a DAG environment (-AutoDagExcludeFromMonitoring $true).

This leaves hiding the mailbox from view. We do not want end users sending emails to this user or being about to see the user in the Global Address List (GAL). Since the rest of the configuration process is complete, we can now hide the mailbox from the GAL (note the wildcard in the name of the mailbox this will get all mailboxes that begin with 'journal'):

```
Get-Mailbox Journal* | Set-Mailbox -HiddenFromAddressListsEnabled $True
```

Before

After



Journaling Verification

Now that the Journaling is in place, how can we verify that journaling is working as expected? There are a few places where we can verify the proper operation of the journaling setup can be verified – contents of the journaling mailbox (if journaling is kept internal), Message Tracking Logs or Mailbox Statistics:

Mailbox Contents

The easiest method is to grant an administrator 'Full Access' to the mailbox and then open the mailbox in

Outlook to verify the contents of the mailbox. The mailbox can also be opened in OWA. Once open an administrator can verify that journaling messages appeared.

Mailbox Statistics

Using mailbox statistics to determine the usage of the Journaling mailbox is not the most insightful way to figure out whether or not the journaling rule is working as expected, but it can provide one vital statistic – does the item count of the mailbox go up. If the mailbox is hidden, then emails destined for it should be ones that were generated by the journaling process. To see the changes, use the Get-MailboxStatistics cmdlet to monitor the item count for the mailbox increment over time. For this one, we can script something small that will run each hour to see how many emails appear in the mailbox over time [in this example every hour for 48 hours]:

```
# Check the statistics on the mailbox each hour, every hour for the next 48 hours
$Hours = 0
Do {
   Write-Host "Mailbox statistics @ $Hours hours."
   Get-Mailbox Journal* | Get-MailboxStatistics -WarningAction 0| ft DisplayName,ItemCount
   Start-Sleep 3600
   $Hours = $Hours++
} While ($Hours -lt 48)
```

Using a Do…While loop, the code visually indicates the hour the script is on. Then the script will report back the item counts for any mailbox with the name 'journal' in the front of it. Notice there is a '-warningaction 0' (a.k.a. SilentlyContinue) for Get-MailboxStatistics; without this, a mailbox that has not been logged into (typically a mailbox like this) will receive a warning message to that affect:

```
WARNING: The user hasn't logged on to mailbox '16-TAP.Local/Users/Journaling-Germany'
('4e88519a-48d3-47a8-acc1-b00607d0b001'), so there is no data to return. After the user logs on, this warning will no
longer appear.
```

The switch will suppress this warning message. After the mailbox statistics displays the current item counts, the script sleeps for an hour. When it starts back up, the hour counter increments by one and so on for 48 hours.

Message Tracking

Following in the footsteps of Chapter 8, a review of tracking logs can provide information on how many messages are getting delivered to a journaling mailbox like so:

```
Get-MessageTrackingLog -start 8/25/16 -ResultSize Unlimited | Where {$_.Recipients -Match "Journal*"}
| ft TimeStamp, EventId, Recipients, Sender -Auto
```

Results will look like something like this:

```
Timestamp              EventId   Recipients                              Sender
---------              -------   ----------                              ------
8/25/2016 2:22:02 AM RECEIVE   {Journaling-Germany@16-tap.local} MicrosoftExchange329e71
8/25/2016 2:22:02 AM AGENTINFO {Journaling-Germany@16-tap.local} MicrosoftExchange329e71
8/25/2016 2:22:02 AM SEND      {Journaling-Germany@16-tap.local} MicrosoftExchange329e71
8/25/2016 2:22:32 AM RECEIVE   {Journaling-Germany@16-tap.local} MicrosoftExchange329e71
8/25/2016 2:22:32 AM AGENTINFO {Journaling-Germany@16-tap.local} MicrosoftExchange329e71
8/25/2016 2:22:32 AM SEND      {Journaling-Germany@16-tap.local} MicrosoftExchange329e71
8/25/2016 2:23:03 AM RECEIVE   {Journaling-Germany@16-tap.local} MicrosoftExchange329e71
8/25/2016 2:23:03 AM AGENTINFO {Journaling-Germany@16-tap.local} MicrosoftExchange329e71
8/25/2016 2:23:03 AM SEND      {Journaling-Germany@16-tap.local} MicrosoftExchange329e71
```

Reporting on Journaling Rules

Knowing what rules are in place can be important if there is a need to troubleshoot a journaling process, especially if there is an external product or process that is analyzing these messages. Understanding what destination email address is being used to journal for what recipient(s). 'Get-JournalRule' is the cmdlet to provide the needed information:

```
Get-JournalRule | ft Name, Recipient, JournalEmailAddress, Scope, Enabled –Auto
```

```
Name                          Recipient                  JournalEmailAddress              Scope Enabled
----                          ---------                  -------------------              ----- -------
Journaling for German Users   GermanJournaling@16-tap.local Journaling-Germany@16-tap.local Global    True
Journaling for Poland mailboxes PolandJournaling@16-tap.local Journaling-Poland@16-tap.local  Global    True
```

Disabling Rules

Why disable a Journaling Rule? Normally the disabling of Journaling Rules is done when the original need has past or a new rule needs to be created or if a different destination server or to who to journal. The Disable-Journal cmdlet is the cmdlet for the job. First we need to use the Get-JournalRule on the rule to be disabled and then pipe '|' that journal rule to the 'Disable-JournalRule' cmdlet:

```
Get-JournalRule "Journaling for German Users" | Disable-JournalRule
```

```
Confirm
Are you sure you want to perform this action?
Disabling journal rule "Journaling for German Users".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [?] Help (default is "Y"): y
```

After disabling the rule, verify that the rule is disabled:

```
Get-JournalRule
```

```
Name                : Journaling for German Users
Recipient           : PolandJournaling@16-tap.local
JournalEmailAddress : Journaling-Poland@16-tap.local
Scope               : Global
Enabled             : False
```

Enabled is now set to 'False'.

If the rule needs to be re-enabled, simply use the same process as the Disable-JournalRule:

```
Get-JournalRule "Journaling for German Users" | Enable-JournalRule
```

This cmdlet does not provide any feedback, only by running Get-JournalRule will you be able to verify if the rule is enabled again.

Removing Rules

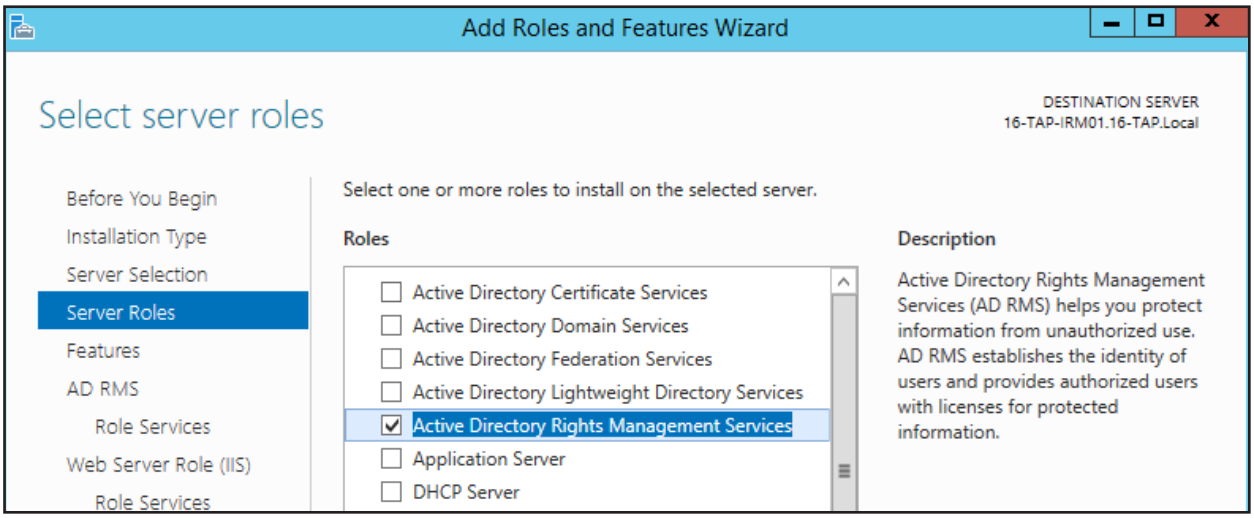
Removing a rule is similar to disabling the Journaling Rule – use the Get-JournalRule and pass that information along to the Remove-JournalRule:

```
Get-JournalRule "Journaling for German Users" | Remove-JournalRule
```

```
Confirm
Are you sure you want to perform this action?
Removing journal rule "Journaling for German Users".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [?] Help (default is "Y"): y
```

Rights Management

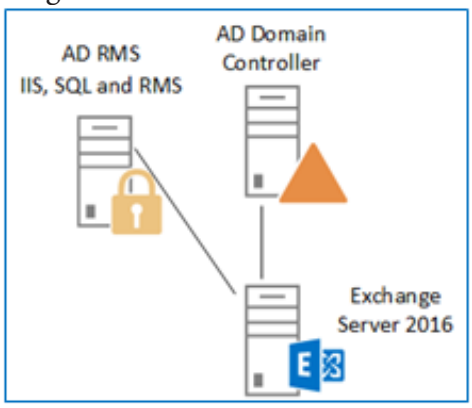
Rights management in Exchange 2016 relies on Active Directory Right Management Services (ADRMS) as an additional security feature that can be added. Exchange refers to RMS as Information Rights Services or IRM. This is important to remember because PowerShell in Exchange uses IRM and not RMS, which is key to knowing what PowerShell cmdlets are available. In order to enable Rights Management in Exchange Server 2016, a Windows Server 2012 (R2) needs to be setup and configured for RMS in Active Directory. This new server needs to have the Active Directory Rights Management role installed on it.



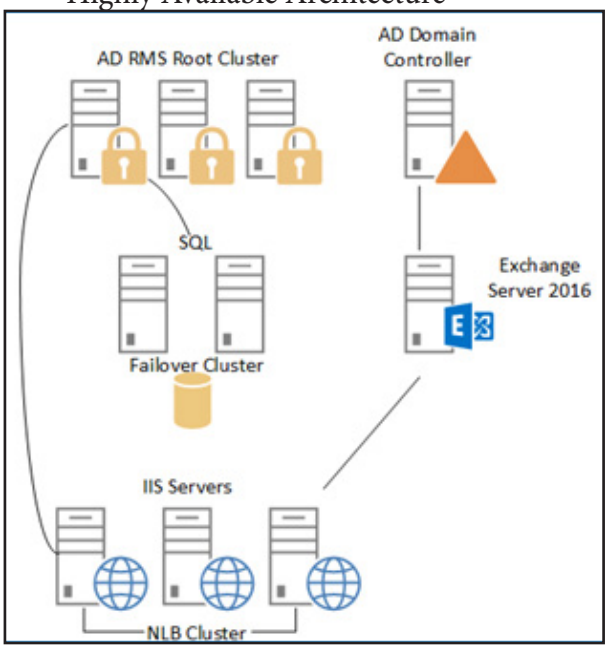
This book will not cover the installation or configuration of a Rights Management infrastructure, but suffice it to say that a single RMS server will be sufficient to provide Exchange with what it needs. However, a redundant architecture can be configured (see below):

Sample RMS Architecture

Single RMS Server



Highly Available Architecture



Once the RMS infrastructure is in place, we can configure Exchange Server to work in conjunction with it.

PowerShell

Get-Command *irm*

```
CommandType     Name
-----------     ----
Alias           irm -> Invoke-RestMethod
Function        Get-IRMConfiguration
Function        Set-IRMConfiguration
Function        Test-IRMConfiguration
```

First explore the current IRM configuration on an Exchange 2016 server.

Get-IRMConfiguration

```
InternalLicensingEnabled        : False
ExternalLicensingEnabled        : False
JournalReportDecryptionEnabled : True
ClientAccessServerEnabled       : True
SearchEnabled                   : True
TransportDecryptionSetting      : Optional
EDiscoverySuperUserEnabled      : True
RMSOnlineKeySharingLocation     :
RMSOnlineVersion                :
ServiceLocation                 :
PublishingLocation              :
LicensingLocation               : {}
```

Notice that by default that IRM is enabled for client access on the Exchange server. However notice that no licensing is enabled or that the IRM configuration is published. Below is a list of parameters that can be configured for IRM:

Get-Help Set-IRMConfiguration -Full

Parameters

ClientAccessServerEnabled <$true | $false> - this option turns on IRM for OWA and ActiveSync
Confirm [<SwitchParameter>]
DomainController <Fqdn>
EDiscoverySuperUserEnabled <$true | $false> -
ExternalLicensingEnabled <$true | $false> - enable or disable IRM for messages sent to external recipients
Force <SwitchParameter>
InternalLicensingEnabled <$true | $false> -
JournalReportDecryptionEnabled <$true | $false>
LicensingLocation <MultiValuedProperty>
PublishingLocation <Uri>
RefreshServerCertificates <SwitchParameter>
RMSOnlineKeySharingLocation <Uri>
SearchEnabled <$true | $false>
TransportDecryptionSetting <Disabled | Optional | Mandatory>

When further customizing the IRM configuration, the following setting should also be considered:

EDiscoverySuperUserEnabled – If this is set to true, users with eDiscovery privileges can access IRM protected emails
SearchEnabled – On by default, it enables OWA to search for IRM messages
ExternalLicensingEnabled - Allows for the use of RMS on external emails
InternalLicensingEnabled – Allows for the use of RMS on internal emails
JournalReportDecryptionEnabled – If journaling is present in Exchange, any IRM messages that are journaled have an unencrypted copy stored with the Journal message

For a scenario where internal messages should be protected by IRM and legal needs to perform legal discovery on emails that are IRM protected, the IRM configuration should be updated like so:

```
Set-IRMConfiguration –EDiscoverySuperUserEnabled $True –InternalLicensingEnabled $True
```

Once configured, RMS is ready to use internally and templates and configuration within RMS should be configured. Both internal and external licensing can be enabled in the same configuration if need be.

```
Set-IRMConfiguration –InternalLicensingEnabled $True – ExternalLicensingEnabled $False
```

In addition to configuring the IRM settings, the same settings can be tested / verified using the Test-IRMConfiguration Cmdlet. To test the configuration for external recipients, the following syntax can be used:

```
Test-IRMConfiguration -Recipient damian@geek.com -Sender damian@BigCorp.Com
```

The test will go through a series of steps:

```
Checking Exchange Server ...
    - PASS: Exchange Server is running in Enterprise.
Loading IRM configuration ...
    - PASS: IRM configuration loaded successfully.
Retrieving RMS Certification Uri ...
    - PASS: RMS Certification Uri: https://adrms.16-tap.local/_wmcs/certification.
Verifying RMS version for https://adrms.16-tap.local/_wmcs/certification ...
    - WARNING: Failed to verify RMS version. IRM features require AD RMS on Windows Server 200
article 973247 (http://go.microsoft.com/fwlink/?linkid=3052&kbid=973247) or AD RMS on Windows
------------------------------------
```

And a final result at the end:

```
------------------------------------

OVERALL RESULT: PASS with warnings on disabled features
```

Outlook Protection Rules

Transport Rules can be created to utilize RMS to protected messages at the Exchange Server level. However, Outlook Protection Rules will protect messages at the Outlook level even before the email has left the Outlook client. PowerShell cmdlets for Outlook Protection Rules:

```
Get-Command *OutlookProt*
```

```
CommandType     Name
-----------     ----
Function        Disable-OutlookProtectionRule
Function        Enable-OutlookProtectionRule
Function        Get-OutlookProtectionRule
Function        New-OutlookProtectionRule
Function        Remove-OutlookProtectionRule
Function        Set-OutlookProtectionRule
```

Like many other protections in Exchange, there are no Outlook rules by default which can be verified with the Get-OutlookProtectionRule cmdlet. In order to get started let's review the New-OutlookProtectionRule examples:

```
Get-Help New-OutlookProtectionRule –Examples
```

```
-------------------------- Example 1 --------------------------

This example applies the AD RMS template Template-Contoso to messages sent to the SMTP address Joe@contoso.com.
New-OutlookProtectionRule -Name "Project Contoso" -SentTo Joe@contoso.com -ApplyRightsProtectionTemplate
"Template-Contoso"
```

Other parameters to consider when working with Outlook Protection Rules are –ApplyRightsProtectionTemplate, SentTo, SentToScope, UserCanOverride and enabled. A sample command would look like this:

```
New-OutlookProtectionRule -Name "R and D" -SentTo Sam@HotMail.Com
-ApplyRightsProtectionTemplate "Big Box – Outlook Rule 1" -UserCaOoverride $False
```

Mobile Protection

Mobile protection via IRM is enabled when the 'ClientAccessServerEnabled' setting is configured for $true:

```
Set-IRMConfiguration –ClientAccessServerEnabled $True
```

Now ActiveSync devices can be protected as well. Microsoft also recommends that certain settings for the Microsoft ActiveSync policy should also be configured:

DevicePasswordEnabled - $True
RequireDeviceEncryption - $True
AllowNonProvisionableDevices - $False

Depending on the name of the ActiveSync policy being applies to your mobile devices, the one-liners to make these changes might be a bit different. In the case below, we will modify the default policy to these settings:

```
Get-MobileDeviceMailboxPolicy | Where {$_.IsDefault -eq "True"} | Set-MobileDeviceMailboxPolicy
-PasswordEnabled $True -RequireDeviceEncryption $True -AllowNonProvisionableDevices $False
```

If IRM is not enabled on the mobile device policy, that should be enabled as well:

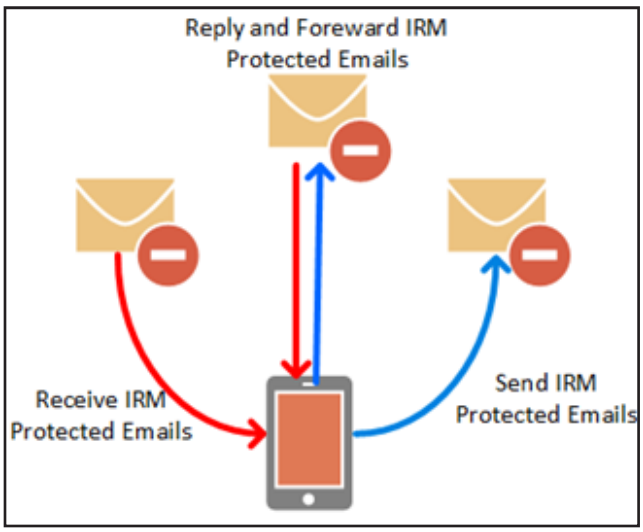
```
Get-MobileDeviceMailboxPolicy | Where {$_.IsDefault -eq "True"} | Set-MobileDeviceMailboxPolicy –
IRMEnabled $True
```

Last configuration items:

Add the Federation mailbox (a system mailbox created by Exchange 2013 and Exchange 2010 Setup) to the super users group in AD RMS. This can be done with a simple one-liner:

```
Add-DistributionGroupMember ADRMSSuperUsers -Member FederatedEmail.4c1f4d8b-8179-4148-
93bf-00a95fa1e042
```

Mobile devices users can now perform the following actions:



Same Message tracking applies to mobile devices as would have been present with OWA and Outlook client emails.

IRM Logging

The Transport Service contains a configuration for IRM logging. These settings can be verified with a Get-TransportService command like so:

```
Get-TransportService | ft IRM* -Auto
```

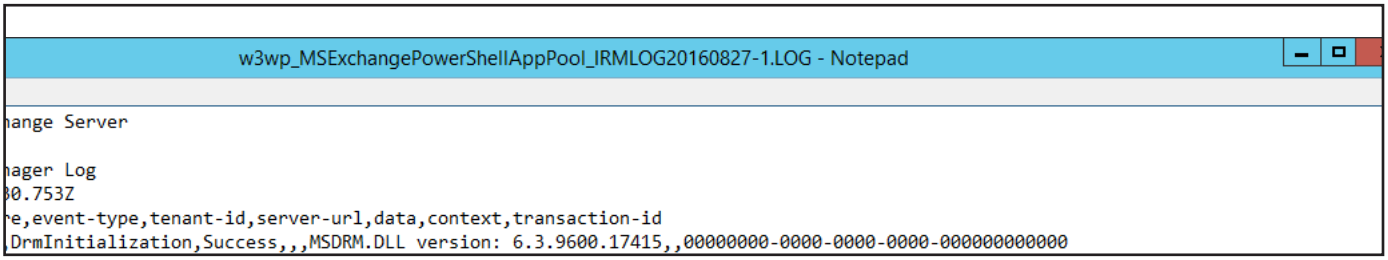
```
Name           IrmLogEnabled IrmLogMaxAge IrmLogMaxDirectorySize
----           ------------- ------------ ----------------------
16-TAP-EX01             True 90.00:00:00  1.221 GB (1,310,720,000 bytes)
16-TAP-EX02             True 90.00:00:00  1.221 GB (1,310,720,000 bytes)
16-04-EDGE-01           True 30.00:00:00  Unlimited
```

```
IrmLogMaxFileSize         IrmLogPath
-----------------         ----------
10 MB (10,485,760 bytes) C:\Program Files\Microsoft\Exchange Server\V15\Logging\IRMLogs
10 MB (10,485,760 bytes) C:\Program Files\Microsoft\Exchange Server\V15\Logging\IRMLogs
10 MB (10,485,760 bytes)
```

Browsing to the log directory for IRM logs and we see there are a two types of files that can be examined:

```
Name
----
w3wp_MSExchangeOWAAppPool_IRMLOG20160813-1.LOG
w3wp_MSExchangeOWAAppPool_IRMLOG20160827-1.LOG
w3wp_MSExchangePowerShellAppPool_IRMLOG20160827-1.LOG
```

Opening up the last file, we find that the RMS server in AD was discovered to be used in Exchange:



This file is used to log all IRM RMS transactions run from PowerShell, while other IRM log files present are for OWA transactions. When in use, up to four total log file types are present:

- Log for Transport transactions for RMS
- Log for Search and Index requests for RMS transactions
- Log for OWA RMS transactions
- Log for PowerShell RMS transactions

These logs make a good place to begin a search for any issues with Exchange and RMS.

One thing to note is that the logs are rather small at 250 MB in size. The age of the logs is also set to a relatively short timeframe of 30 days. A recommended setting, especially on a busy server, it may be necessary to adjust the max age to 90 days and the size to something over 1 GB in size. To change this, the Set-TransportService needs to be utilized:

```
Get-TransportService | Set-TransportService -IrmLogMaxAge 90.00:00:00 -IrmLogMaxDirectorySize
1250MB
```

For the Edge Transport Role, the command needs to be run locally:

```
You can't use this command to configure an Edge Transport server on a machine that is on your
internal network. You must perform this operation directly on the Edge Transport server.
    + CategoryInfo          : InvalidOperation: (:) [Set-TransportService], CannotSetEdgeTr...erOn
   AdException
    + FullyQualifiedErrorId : [Server=16-TAP-EX02,RequestId=0b3ca335-9689-4042-9848-c4f36032ad00,T
   imeStamp=1/16/2017 3:50:05 AM] [FailureCategory=Cmdlet-CannotSetEdgeTransportServerOnAdExcepti
  on] 1ABD3ACD,Microsoft.Exchange.Management.SystemConfigurationTasks.SetTransportService
    + PSComputerName        : 16-tap-ex02.16-tap.local
```

Once the cmdlet is run locally on the Edge Transport server, the new results are listed below:

```
Get-TransportService 16-04-edge-01 | Set-TransportService -IrmLogMaxAge 90.00:00:00
-IrmLogMaxDirectorySize 1250MB
```