Using Message Filters to Enforce Email Policies

The appliance contains extensive content scanning and message filtering technology that allows you to enforce corporate policies and act on specific messages as they enter or leave your corporate networks.

This chapter contains information about the powerful combinations of features available for policy enforcement: a content scanning engine, message filters, attachment filters, and content dictionaries.

This chapter contains the following sections:

- Overview, on page 1
- Components of a Message Filter, on page 2
- Message Filter Processing, on page 4
- Message Filter Rules, on page 9
- Message Filter Actions, on page 55
- Attachment Scanning, on page 87
- Detecting Malicious Files in Messages Attachments Using Message Filter , on page 97
- Using the CLI to Manage Message Filters, on page 98
- Message Filter Examples, on page 112
- Configuring Scan Behavior, on page 120

Overview

Message filters allow you to create special rules describing how to handle messages as they are received by the appliance . A message filter specifies that a certain kind of email message should be given special treatment. Cisco message filters also allow you to enforce corporate email policy by scanning the content of messages for words you specify. This chapter contains the following sections:

- **Components of a message filter**. Message filters allow you to create special rules describing how to handle messages as they are received. Filter rules identify messages based on message or attachment content, information about the network, message envelope, message headers, or message body. Filter actions generate notifications or allow messages to be dropped, bounced, archived, blind carbon copied, or altered. For more information, see Components of a Message Filter, on page 2.
- **Processing Message Filters**. When AsyncOS processes message filters, the content that AsyncOS scans, the order of the processing, and the actions taken are based on several factors, including the message filter order, any prior processing that may have altered the message content, the MIME structure of the message, the threshold score configured for content matching, and structure of the query. For more information, see Message Filter Processing, on page 4.

- **Message Filter Rules**. Each filter has a rule that defines the collection of messages that the filter can act upon. You define those rules when you create a message filter. For more information, see Message Filter Rules, on page 2.
- **Message Filter Actions**. Each filter has an action that is performed on a message if the rule evaluates to true . There are two types of actions that can be performed: final actions (such as delivering, dropping, or bouncing a message), or non-final actions (such as stripping or inserting a header) which permit the message to be further processed. For more information, see Message Filter Actions, on page 2.
- **Attachment Scanning Message Filters**. Attachment scanning message filters allow you to strip attachments from messages that are inconsistent with your corporate policies, while still retaining the ability to deliver the original message. You can filter attachments based on their specific file type, fingerprint, or content. You can also scan image attachments using an image analyzer. The image analyzer uses algorithms that measure image attributes to determine the likelihood of inappropriate content. These algorithms can detect, for example, the shapes and color palette in an image. The analyzer can identify the type of shapes in an image and the percentage of any flesh-tone colors relative to the other colors in the image to help identify inappropriate content. Images with a high percentage of flesh-tone colors are more likely to be inappropriate. The algorithms do not discriminate in any way. For more information, see Attachment Scanning, on page 87.
- **Using the CLI to Manage Message Filters**. The CLI accepts commands for working with message filters. For example, you might want to display, reorder, import or export a list of message filters. For more information, see Using the CLI to Manage Message Filters, on page 98.
- **Message Filter Examples**. This section contains some real world examples of filters with a brief discussion of each. For more information, see Message Filter Examples, on page 112.

Components of a Message Filter

Message filters allow you to create special rules describing how to handle messages as they are received. A message filter is comprised of message filter rules and message filter actions.

Related Topics

- Message Filter Rules, on page 2
- Message Filter Actions, on page 2
- Message Filter Example Syntax, on page 3

Message Filter Rules

Message filter rules determine the messages that a filter will act on. Rules may be combined using the logical connectors AND, OR, and NOT to create more complex tests. Rule expressions may also be grouped using parentheses.

Message Filter Actions

The purpose of message filters is to perform actions on selected messages.

The two types of actions are:

- *Final* actions — such as deliver , drop , and bounce — end the processing of a message, and permit no further processing through subsequent filters.
- *Non-final* actions perform an action which permits the message to be processed further.

Note Non-final message filter actions are cumulative. If a message matches multiple filters where each filter specifies a different action, then all actions are accumulated and enforced. However, if a message matches multiple filters specifying the same action, the prior actions are overridden and the final filter action is enforced.

Related Topics

- Filter Actions Summary Table, on page 56
- Action Variables, on page 65
- Matched Content Visibility, on page 67
- Description and Examples of Message Filter Actions, on page 68

Message Filter Example Syntax

The intuitive meaning of a filter specification is:

if the message matches the rule, *then* apply the actions in sequence. If the else clause is present, the actions within the else clause are executed in the event the message does not match the rule.

The name of the filter you specify makes it easier to manage filters when you are activating, deactivating, or deleting them.

Message filters use the following syntax:

Example Syntax	Purpose
expedite:	filter name
if (recv-listener == 'InboundMail' or recv-int == 'notmain')	rule specification

`{`
`  alt-src-host('outbound1');`
`skip-filters();`
`}`
	action specification

`else`
`{`
`  alt-src-host('outbound2');`
`}`
	optional alternative action specification

Note that you can omit any alternative actions:

Example Syntax	Purpose
expedite2:	filter name
`if ((not (recv-listener == 'InboundMail')) and` (not (recv-int == 'notmain')))	rule specification

Example Syntax	Purpose

`{`
`  alt-src-host('outbound2');`
`skip-filters();`
`}`
	action specification

You can combine several filters in sequence within a single text file, one following the other.

You must enclose the values in filters in either single or double quotation marks. Single or double quotation marks must be equally paired on each side of the value; for example, the expressions `notify('customercare@example.com')` and `notify("customercare@example.com")` are both legal, but the expression `notify("customercare@example.com')` causes a syntax error.

Lines beginning with a ‘ # ’ character are considered comments and are ignored; however, they are not preserved by AsyncOS as can be verified by viewing a filter via `filters -> detail` .

Message Filter Processing

When AsyncOS processes message filters, the content that AsyncOS scans, the order of the processing, and the actions taken are based on several factors:

- **Message filter order.** Message filters are maintained in an ordered list. When a message is processed, AsyncOS applies each message filter in the order it appears in the list. If a final action occurs, no further action is taken on the message. For more information, see Message Filter Order, on page 5.
- **Prior processing**. Actions performed on AsyncOS messages may add or remove headers before the message filter is evaluated. AsyncOS processes the message filter process on the headers that are present in the message at the time of processing. For more information, see Message Header Rules and Evaluation, on page 5.
- **The MIME structure of the message**. The MIME structure of the message determines which part of the message is treated as “body,” and which part of the message is treated as an “attachment”. Many message filters are configured to act on just the body or just the attachment part of the message. For more information, see Message Bodies vs. Message Attachments, on page 5.
- **The threshold score configured for the regular expression**. When you match a regular expression, you configure a “score” to tally up the number of times a match must occur before a filter action is taken. This allows you to “weight” the responses to different terms. For more information, see Thresholds for Matches in Content Scanning, on page 6 .
- **The structure of the query**. When evaluating AND or OR tests within message filters, AsyncOS does not evaluate unneeded tests. In addition, it is important to note that the system does not evaluate the tests from left to right. Instead, when AND and OR tests are evaluated, the least expensive test is evaluated first. For more information, see AND Test and OR Tests in Message Filters, on page 9.

Related Topics

- Message Filter Order, on page 5
- Message Header Rules and Evaluation, on page 5
- Message Bodies vs. Message Attachments, on page 5
- Thresholds for Matches in Content Scanning, on page 6
- AND Test and OR Tests in Message Filters, on page 9

Message Filter Order

Message filters are kept in an ordered list and numbered by their position in the list. When a message is processed, the message filters are applied in the associated numeric order. Therefore, filter number 30 will not have a chance to alter the source host of a message if filter number 9 has already executed a final action on (for example, bounced) the message. The position of a filter in the list can be changed via the system user interfaces. Filters imported via a file are ordered based on their relative order in the imported file.

After a final action, no further actions may be taken on the message.

Although a message may match a filter rule, the filter may not act upon that message for any of the following reasons:

- The filter is inactive.
- The filter is invalid.
- The filter has been superseded by an earlier filter that executed a final action for the message.

Message Header Rules and Evaluation

Filters evaluate “processed” headers rather than the original message headers when applying header rules. Thus:

- If a header was added by a previous processing action, it can now be matched by any subsequent header rule.
- If a header was stripped by a previous processing action, it can no longer be matched by any subsequent header rule.
- If a header was modified by a previous processing action, any subsequent header rule will evaluate the modified header and not the original message header.

This behavior is common to both message filters and content filters.

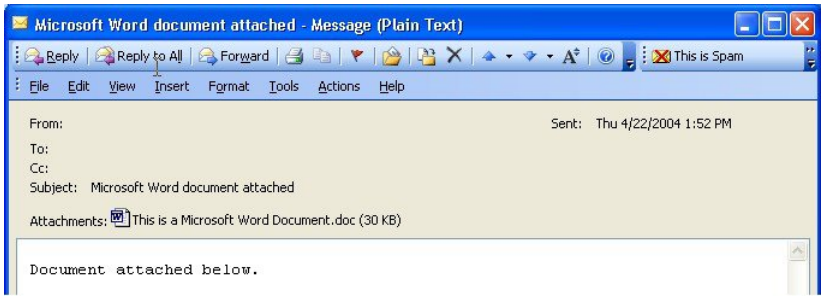
Message Bodies vs. Message Attachments

An email message is composed of multiple parts. Although RFCs define everything that comes after a message’s headers as a multipart “message body,” many users still conceptualize a message’s “body” and its “attachment” differently. When you use any of the Cisco message filters named body- *variable* or `attachment`-*variable* , the appliance attempts to distinguish the parts that most users consider to be the “body” and the “attachment” in the same way that many MUAs attempt to render these parts differently.

For the purposes of writing body - *variable* or `attachment`-*variable* message filter rules, everything after the message headers is considered the message *body* , whose content is considered the first text part of the MIME parts that are within the body. Anything after the content, (that is, any additional MIME parts) is considered an *attachment* . AsyncOS evaluates the different MIME parts of the message, and identifies the parts of the file that is treated as an attachment.

For example, The following figure shows a message in the Microsoft Outlook MUA where the words “ `Document attached below.` ” appear as a plain text message *body* and the document “ `This is a Microsoft Word document.doc` ” appears as an *attachment* . Because many users conceptualize email this way (rather than as a multipart message whose first part is plain text and whose second part is a binary file), the Cisco uses the term “attachment” in message filters to create rules to differentiate and act on the .doc file part (in essence, the second MIME part) as opposed to the “body” of the message (the first, plain text part) — although, according to the language used in RFCSs 1521 and 1522, a message’s *body* is comprised of all MIME parts.

Figure 1: Message with "Attachment"



Because the appliance makes this distinction between the *body* and the *attachment* in multipart messages, there are several cases you should be aware of when using the body - *variable* or `attachment`-*variable* message filter rules in order to achieve the expected behavior:

- If you have a message with a single text part—that is, a message containing a header of "Content-Type: text/plain" or "Content-Type: text/html" — the appliance will consider the entire message as the body. If the content type is anything different, the appliance considers it to be a single attachment.
- Some encoded files (uuencoded, for example) are included in the body of the email message. When this occurs, the encoded file is treated as an attachment, and it is extracted and scanned, while the remaining text is considered to be the body of the text.
- A single, non-text part is always considered an *attachment* . For example, a message consisting of only a.zip file is considered an attachment.

Thresholds for Matches in Content Scanning

When you add filter rules that search for patterns in the message body or attachments, you can specify the minimum threshold for the number of times the pattern must be found. When AsyncOS scans the message, it totals the "score" for the number of matches it finds in the message and attachments. If the minimum threshold is not met, the regular expression does not evaluate to true. You can specify this threshold for the following filter rules:

- body-contains
- only-body-contains
- attachment-contains
- every-attachment-contains
- dictionary-match
- attachment-dictionary-match

You can also specify a threshold value for the `drop-attachments-where-contains` action.

Note You cannot specify thresholds for filter rules that scan headers or envelope recipients and senders.

Related Topics

- Threshold Syntax, on page 7
- Threshold Scoring for Message Bodies and Attachments, on page 7
- Threshold Scoring Multipart/Alternative MIME Parts, on page 7

• Threshold Scoring for Content Dictionaries, on page 8

Threshold Syntax

To specify a threshold for the minimum number of occurrences, specify the pattern and the minimum number of matches required to evaluate to true:

```
if(<filter rule>('<pattern>',<minimum threshold>)){
```

For example, to specify that the body-contains filter rule must find the value “Company Confidential” at least two times, use the following syntax:

```
if(body-contains('Company Confidential',2)){
```

By defeat, when AsyncOS saves a content scanning filter, it compiles the filter and assigns a threshold value of 1, if you have not assigned a value.

You can also specify a minimum number of pattern matches for values in a content dictionary. For more information about content dictionaries, see the “Text Resources” chapter.

Threshold Scoring for Message Bodies and Attachments

An email message may be composed of multiple parts. When you specify threshold values for filter rules that search for patterns in the message body or attachments, AsyncOS counts the number of matches in the message parts and attachments to determine the threshold “score.” Unless the message filter specifies a specific MIME part (such as the `attachment-contains` filter rule), AsyncOS will total the matches found in all parts of the message to determine if the matches total the threshold value. For example, you have a `body-contains` message filter with a threshold of 2. You receive a message in which the body contains one match, and the attachment contains one match. When AsyncOS scores this message, it totals the two matches and determines that the threshold score has been met.

Similarly, if you have multiple attachments, AsyncOS totals the scores for each attachment to determine the score for matches. For example, you have an `attachment-contains` filter rule with a threshold of 3. You receive a message with two attachments, and each attachment contains two matches. AsyncOS would score this message with four matches and determine that the threshold score has been met.

Threshold Scoring Multipart/Alternative MIME Parts

To avoid duplicate counting, if there are two representatives of the same content (plain text and HTML), AsyncOS does not total the matches from the duplicate parts. Instead, it compares the matches in each part and selects the highest value. AsyncOS would then add this value to the scores from other parts of the multipart message to create a total score.

For example, you configure a `body-contains` filter rule and set the threshold to 4. You then receive a message that contains both plain text, HTML and two attachments. The message would use the following structure:

```
multipart/mixed

        multipart/alternative

                text/plain

                text/html
```

```
        application/octet-stream

        application/octet-stream
```

The `body-contains` filter rule would determine the score for this message by first scoring the text/plain and text/html parts of the message. It would then compare the results of these scores and select the highest score from the results. Next, it would add this result to the score from each of the attachments to determine the final score. Suppose the message has the following number of matches:

```
multipart/mixed

        multipart/alternative

                text/plain (2 matches)

                text/html (2 matches)

        application/octet-stream (1 match)

        application/octet-stream
```

Because AsyncOS compares the matches for the text/plain and text/html parts, it returns a score of 3, which does not meet the minimum threshold to trigger the filter rule.

Threshold Scoring for Content Dictionaries

When you use a content dictionary, you can "weight" terms so that certain terms trigger filter actions more easily. For example, you may want not want to trigger a message filter for the term, "bank." However, if the term, "bank" is combined with the term, "account," and accompanied with an ABA routing number, you may want to trigger a filter action. To accomplish this, you can use a weighted dictionary to give increased importance to certain terms or a combination of terms. When a message filter that uses a content dictionary scores the matches for filter rule, it uses these weights to determine the final score. For example, suppose you create a content dictionary with the following contents and weights:

Table 1: Sample Content Dictionary

Term/Smart Identifier	Weight
ABA Routing Number	3
Account	2
Bank	1

When you associate this content dictionary with a `dictionary-match` or `attachment-dictionary-match` message filter rule, AsyncOS would add the weight for the term to the total "score" for each instance of the matching term found in the message. For example, if the message contains three instances of the term, "account" in the message body, AsyncOS would add a value of 6 to the total score. If you set the threshold value for the message filter to 6, AsyncOS would determine that the threshold score has been met. Or, if the message contained one instance of each term, the total value would be 6, and this score would trigger the filter action.

AND Test and OR Tests in Message Filters

When evaluating AND or OR tests within message filters, AsyncOS does not evaluate unneeded tests. So, for example, if one side of an AND test is false, the system will not evaluate the other side. It is important to note that the system does not evaluate the tests from left to right. Instead, when AND and OR tests are evaluated, the least expensive test is evaluated first. For example, in the following filter, the `remote-ip` test will always be processed first because it has a lower cost than the `rcpt-to-group` test (generally LDAP tests are more expensive):

```
andTestFilter:

if (remote-ip == "192.168.100.100" AND rcpt-to-group == "GROUP")

  { ... }
```

Because the least expensive test is performed first, switching the order of the items in the test will have no effect. If you want to guarantee the order in which tests are performed, use nested if statements. This is also the best way to ensure that an expensive test is avoided whenever possible:

```
expensiveAvoid:

if (<simple tests>)

  { if (<expensive test>)

    { <action> }

  }
```

In a somewhat more complicated example, consider:

```
if (test1 AND test2 AND test3) { ... }
```

The system groups the expression from left to right, so this becomes:

```
if ((test1 AND test2) AND test3) { ... }
```

This means the first thing the system does is compare the cost of (`test1` AND `test2`) against the cost of `test3`, evaluating the second AND first. If all three tests have the same cost, then `test3` will be performed first because (`test1` AND `test2`) would be twice as expensive.

Message Filter Rules

Each message filter contains a rule that defines the collection of messages that a filter can act upon. You define the filter rules, and then you define a filter action for messages that return true .

Related Topics

- Filter Rules Summary Table, on page 10
- Regular Expressions in Rules, on page 20
- Smart Identifiers, on page 24

• Description and Examples of Message Filter Actions, on page 68

Filter Rules Summary Table

The following table summarizes the rules you can use in message filters.

Table 2: Message Filter Rules

Rule	Syntax	Description
Subject Header	`subject`	Does the subject header match a certain pattern? See Subject Rule, on page 27.
Body Size	`body-size`	Is the body size within some range? See Body Size Rule, on page 29.
Envelope Sender	`mail-from`	Does the Envelope Sender (i.e., the Envelope From, <MAIL FROM>) match a given pattern? See Envelope Sender Rule, on page 28.
Envelope Sender in Group	`mail-from-group`	Is the Envelope Sender (i.e., the Envelope From, <MAIL FROM>) in a given LDAP group? See Envelope Sender in Group Rule, on page 29.
Sender Group	`sendergroup`	Which sender group was matched in a listener's Host Access Table (HAT)? See Sender Group Rule, on page 29.
Envelope Recipient	`rcpt-to`	Does the Envelope Recipient, (i.e. the Envelope To, <RCPT TO>) match a given pattern? See Envelope Recipient Rule, on page 27. **Note** The `rcpt-to` rule is message-based. If a message has multiple recipients, only one recipient has to match the rule for the specified action to affect the message to all recipients.

Rule	Syntax	Description
Envelope Recipient in Group	`rcpt-to-group`	Is the Envelope Recipient, (i.e. the Envelope To, <RCPT TO>) in a given LDAP group? See Envelope Recipient in Group Rule, on page 28. **Note** The `rcpt-to-group` rule is message-based. If a message has multiple recipients, only one recipient has to be found in a group for The rcptthe specified action to affect the message to all recipients.
Remote IP	`remote-ip`	Was the message sent from a remote host that matches a given IP address or IP block? See Remote IP Rule, on page 30.
Receiving Interface	`recv-int`	Did the message arrive via the named receiving interface? See .Receiving IP Interface Rule, on page 31
Receiving Listener	`recv-listener`	Did the message arrive via the named listener? See Receiving Listener Rule, on page 30.
Date	`date`	Is current time before or after a specific time and date? See Date Rule, on page 31.
Header	`header(<string>)`	Does the message contain a specific header? Does the value of that header match a certain pattern? See Header Rule, on page 31.
Random	`random(<integer>)`	Is a random number in some range? See Random Rule, on page 32.
Recipient Count	`rcpt-count`	How many recipients is this email going to? See Recipient Count Rule, on page 33.

Rule	Syntax	Description
Address Count	`addr-count()`	What is the cumulative number of recipients? This filter differs from the rcpt-count filter rule in that it operates on the message body headers instead of the envelope recipients. See Address Count Rule, on page 33.
SPF Status	`spf-status`	What was the SPF verification status? This filter rule allows you to query for different SPF verification results. You can enter a different action for each valid SPF/SIDF return value. See SPF-Status Rule, on page 39.
SPF Passed	`spf-passed`	Did the SPF/SIDF verification pass? This filter rule generalizes the SPF/SIDF results as a Boolean value. See SPF-Passed Rule, on page 41.
S/MIME Gateway Message	`smime-gateway`	Is the message S/MIME signed, encrypted, or signed and encrypted? See S/MIME Gateway Message Rule, on page 41
S/MIME Gateway Verified	`smime-gateway-verified`	Is the S/MIME message successfully verified, decrypted, or decrypted and verified? See S/MIME Gateway Verified Rule, on page 41.
Image verdict	`image-verdict`	What was the image scanning verdict? This filter rule allows you to query for different image analysis verdicts. See Image Analysis, on page 90.
Workqueue count	`workqueue-count`	Is the work queue count equal to, less than, or greater than the specified value? See Workqueue-count Rule, on page 41.

Rule	Syntax	Description
Body Scanning	`body-contains( <regular expression>)`	Does the message contain text or an attachment that matches a specified pattern? Does the pattern occur the minimum number of times you specified for the threshold value? The engine scans delivery-status parts and associated attachments. See Body Scanning , on page 34.
Body Scanning	`only-body-contains (<regular expression>)`	Does the message body contain text that matches a specified pattern? Does the pattern occur the minimum number of times you specified for the threshold value? Attachments are not scanned. See Body Scanning Rule, on page 33.
Encryption Detection	`encrypted`	Is the message encrypted? See Encryption Detection Rule, on page 34.
Attachment Filename	`attachment-filename`	Does the message contain an attachment with a filename that matches a specific pattern? See Attachment Filename Rule, on page 35.
Attachment Type	`attachment-type`	Does the message contain an attachment of a particular MIME type? See Attachment Type Rule, on page 35.

Rule	Syntax	Description
Attachment File Type	`attachment-filetype`	Does the message contain an attachment of a file type that matches a specific pattern based on its fingerprint (similar to a UNIX `file` command)? If the attachment is an Excel or Word document, you can also search for the following embedded file types: .exe , .dll, .bmp, .tiff, .pcx, .gif, .jpeg, png, and Photoshop images. You must enclose the file type in quotes to create a valid filter. You can use single or double quotes. For example, to search for .exe attachments, use the following syntax: `if (attachment-filetype == "exe")` For more information, see Attachment Filenames and Single Compressed Files within Archive Files, on page 36.
Attachment MIME Type	`attachment-mimetype`	Does the message contain an attachment of a specific MIME type? This rule is similar to the `attachment-type` rule, except only the MIME type given by the MIME attachment is evaluated. (The appliance does not try to "guess" the type of the file by its extension if there is no explicit type given.) See Examples of Attachment Scanning Message Filters, on page 94.
Attachment Protected	`attachment-protected`	Does the message contain an attachment that is password protected? See Quarantining Protected Attachments, on page 97.

Rule	Syntax	Description
Attachment Unprotected	`attachment-unprotected`	The attachment-unprotected filter condition returns true if the scanning engine detects an attachment that is unprotected. A file is considered unprotected if the scanning engine was able to read the attachment. A zip file is considered to be unprotected if any of its members is unprotected. **Note —** The attachment-unprotected filter condition is not mutually exclusive of the attachment-protected filter condition. It is possible for both filter conditions to return true when scanning the same attachment. This can occur, for example, if a zip file contains both protected and unprotected members. See Detecting Unprotected Attachments, on page 97.
Attachment Scanning	`attachment-contains (<regular expression>)`	Does the message contain an attachment that contains text or another attachment that matches a specific pattern? Does the pattern occur the minimum number of times you specified for the threshold value? This rule is similar to the `body-contains()` rule, but it attempts to avoid scanning the entire "body" of the message. That is, it attempts to scan only that which the user would view as being an attachment. See Examples of Attachment Scanning Message Filters, on page 94.
Attachment Scanning	`attachment-binary-contains (<regular expression>)`	Does the message contain an attachment with binary data that matches a specific pattern? This rule is like the `attachment-contains ()` rule, but it searches specifically for patterns in the binary data.

Rule	Syntax	Description
Attachment Scanning	`every-attachment-contains (<regular expression>)`	Do all of the attachments in this message contain text that matches a specific pattern? The text must exist in all attachments and the action performed is, in effect, a logical AND operation of an ' `attachment-contains()` ' for each attachment. The body is not scanned. Does the pattern occur the minimum number of times you specified for the threshold value? See Examples of Attachment Scanning Message Filters, on page 94.
Attachment Size	`attachment-size`	Does the message contain an attachment whose size is within some range? This rule is similar to the `body-size` rule, but it attempts to avoid scanning the entire "body" of the message. That is, it attempts to scan only that which the user would view as being an attachment. The size is evaluated prior to any decoding. See Examples of Attachment Scanning Message Filters, on page 94.
Public Blocked lists	`dnslist(<query server>)`	Does the sender's IP address appear on a public blocked list server (RBL)? See DNS List Rule, on page 36.
SenderBase Reputation	`reputation`	What is the sender's SenderBase Reputation Score? See SenderBase Reputation Rule, on page 37.
No SenderBase Reputation	`no-reputation`	Used to test if SenderBase Reputation Score is "None." See SenderBase Reputation Rule, on page 37.
Dictionary	`dictionary-match (<dictionary_name>)`	Does the message body contain any of the regular expressions or terms in the content dictionary named *dictionary_name* ? Does the pattern occur the minimum number of times you specified for the threshold value? See Dictionary Rules, on page 37.

Rule	Syntax	Description
Attachment Dictionary Match	`attachment-dictionary-match (<dictionary_name>)`	Does the attachment contain any of the regular expressions in the content dictionary named *dictionary_name* ? Does the pattern occur the minimum number of times you specified for the threshold value? See Dictionary Rules, on page 37.
Subject Dictionary Match	`subject-dictionary-match (<dictionary_name>)`	Does the Subject header contain any of the regular expressions or terms in the content dictionary named *dictionary name* ? See Dictionary Rules, on page 37.
Header Dictionary Match	`header-dictionary-match (<dictionary_name>, <header>)`	Does the specified header (case insensitive) contain any of the regular expressions or terms in the content dictionary named *dictionary name* ? See Dictionary Rules, on page 37.
Body Dictionary Match	`body-dictionary-match (<dictionary_name>)`	This filter condition returns true if the dictionary term matches content in the body of the message only. The filter searches for terms within the MIME parts not considered to be an attachment. and it returns true if the user-defined threshold is met (the default threshold value is one). See Dictionary Rules, on page 37.
Envelope Recipient Dictionary Match	`rcpt-to-dictionary-match (<dictionary_name>)`	Does the envelope recipient contain any of the regular expressions or terms in the content dictionary named *dictionary name* ? See Dictionary Rules, on page 37.
Envelope Sender Dictionary Match	`mail-from-dictionary-match (<dictionary_name>)`	Does the envelope sender contain any of the regular expressions or terms in the content dictionary named *dictionary name* ? See Dictionary Rules, on page 37.
SMTP Authenticated User Match	`smtp-auth-id-matches (<target>[, <sieve-char>])`	Does the address of the Envelope Sender and the address in message header match the authenticated SMTP user ID of the sender? See SMTP Authenticated User Match Rule, on page 42.

Rule	Syntax	Description
True	`true`	Matches all messages. See True Rule, on page 26.
Valid	`valid`	Returns false if the message contains unparsable/invalid MIME parts and true otherwise. See Valid Rule, on page 27.
Signed	`signed`	Is the message is signed? See Signed Rule, on page 43.
Signed Certificate	`signed-certificate (<field> [<operator> <regular expression>])`	Does the message signer or X.509 certificate issuer match a certain pattern? See Signed Certificate Rule, on page 44.
Header Repeats	`header-repeats (<target>, <threshold> [, <direction>])`	Returns `true` if at a given point in time, a specified number of messages: • With same subject header are detected in last one hour. • From same envelope-sender are detected in last one hour. See Header Repeats Rule, on page 46.
URL Reputation	`url-reputation` `url-no-reputation`	Is the reputation score of any URL in the message within the specified range? Is a reputation score for a URL unavailable? See URL Reputation Rules , on page 48 and Configuring Email Gateway to Consume External Threat Feeds.
URL Category	`url-category`	Does the category of any URL in the message match the specified categories? See URL Category Rule , on page 49.
Corrupt Attachment	`attachment-corrupt`	Does this message have an attachment that is corrupt? See Corrupt Attachment Rule, on page 49.

Rule	Syntax	Description
Message Language	`message-language`	Is the message (subject and body) in one of the selected languages? See Message Language Rule, on page 49.
Macro Detection	`macro-detection-rule (['file_type-1', 'file_type-2', …,'file_type-n'])`	Does the incoming or outgoing message contain macro-enabled attachments? See Macro Detection Rule, on page 50
Forged Email Detection	`forged-email-detection ("<dictionary_name>", <threshold>)`	Is the sender address of the message forged? The rule checks if the From: header in the message is similar to any of the users in the content dictionary. See Forged Email Detection Rule, on page 51.
Duplicate Boundaries Verification	`duplicate_boundaries`	Does the message contain duplicate MIME boundaries? See Duplicate Boundaries Verification Rule, on page 52.
Malformed MIME Header Detection	`malformed-header`	Does the message contain malformed MIME headers? See Malformed MIME Header Detection Rule, on page 52.
Geolocation	`geolocation-rule (['country_name-1', 'country_name-2', 'country_name-n'])`	Does the incoming message originate from the selected countries? **Note** Enable the Anti-Spam engine on your appliance before you use the Geolocation message filter rule. See Geolocation Rule, on page 53.

Rule	Syntax	Description

Domain Reputation	`Sender Domain Reputation:`
	` - sdr-reputation`
	`  (<'sdr_verdict_range'>,`
	`  <'domain_exception`
	`  _list'>)`
	` - sdr-age (<'unit'>,`
	`  <'operator'>`
	`  <'actual value'>)`
	` - sdr-unscannable`
	`  (<'domain_exception`
	`  _list'>)`
	
	`External Threat Feeds:`
	` domain-external-`
	` threat-feeds`
	` (<'external_threat_`
	` feed_source_name'>,`
	` <'header'> , <'domain`
	`_exception_list'>)`
		Does the sender domain match the specified criteria?
		• Sender Domain Reputation
		• External Threat Feeds
		See Domain Reputation Rule for ETF, on page 53 or Domain Reputation Rule for SDR, on page 53.
		For more information, see the Configuring Email Gateway to Consume External Threat Feeds or Sender Domain Reputation Filtering.

Each message injected into the appliance is processed through all message filters in order, unless you specify a final action, which stops the message from being processed further. (See Message Filter Actions, on page 2.) Filters may also apply to all messages, and rules may also be combined using logical connectors (AND, OR, NOT).

Regular Expressions in Rules

Several of the atomic tests used to define rules use *regular expression matching* . Regular expressions can become complex. Use the following table as a guide for the applying of regular expressions within message filter rules:

Table 3: Regular Expression in Rules

Regular expression (`abc`)	Regular expressions in filter rules match a string if the sequence of directives in the regular expression match any part of the string. For example, the regular expression Georg matches the string `George Of The Jungle` , the string `Georgy Porgy` , the string `La Meson Georgette as well as Georg` .
Carat (^) Dollar sign (`$`)	Rules containing the dollar sign character ($) only match the end of the string, and rules containing the caret symbol (^) only match the beginning of the string. For example, the regular expression `^Georg$` only matches the string `Georg` . Searching for an empty header would look like this: "^$"
Letters, white space and the at sign (`@`) character	Rules containing characters, white space, and the at sign character (@) only match themselves explicitly. For example, the regular expression `^George@admin$` only matches the string `George@admin` .

Period character (.)	Rules containing a period character (.) match any character (except a new line). For example, the regular expression `^...admin$` matches the string `macadmin` as well as the string `sunadmin` but not `win32admin` .
Asterisk (*) directive	Rules containing an asterisk (*) match "zero or more matches of the previous directive." In particular, the sequence of a period and an asterisk (.*) matches any sequence of characters (not containing a new line). For example, the regular expression `^P.*Piper$` matches all of these strings: `PPiper` , `Peter Piper` , `P.Piper` , and `Penelope Penny Piper` .
Backslash special characters (\ )	The backslash character *escapes* special characters. Thus the sequence `\.` only matches a literal period, the sequence `\$` only matches a literal dollar sign, and the sequence `\^` only matches a literal caret symbol. For example, the regular expression `^ik\.ac\.uk$` only matches the string `ik.ac.uk` . **Important Note:** The backslash is also a special escape character for the parser. As a result, if you want to include backslash in your regular expression, you must use *two* backslashes — so that after parsing, only one "real" backslash remains, which is then passed to the regular expression system. So, if you wanted to match the example domain above, you would enter `^ik\\.ac\\.uk$` .
Case-insensitivity `( (?i) )`	The token (?i) that indicates the rest of the regular expression should be treated in case-insensitive mode. Placing this token at the beginning of a case-sensitive regular expression results in a completely insensitive match. For example, the regular expression " `(?i)viagra` " matches `Viagra` , `vIaGrA` , and `VIAGRA` .
Number of repetitions `{min,max}`	The regular expression notation that indicates the number of repetitions of the previous token is supported. For example, the expression " `fo{2,3}` " matches `foo` and `fooo` but not fo or fofo . This statement: `if(header('To') == "^.{500,}")` looks for a "To" header that has 500 or more characters in it.
Or (|)	Alternation, or the "or" operator. If A and B are regular expressions, the expression " `A|B` " will match any string that matches either "A" or "B." For example, the expression " `foo|bar` " will match either `foo` or `bar` , but not `foobar` .

Related Topics

- Using Regular Expressions to Filter Messages, on page 22
- Guidelines for Using Regular Expressions, on page 22
- Regular Expression and Non-ASCII Character Sets, on page 22
- n Tests, on page 23
- Case-sensitivity, on page 23
- Writing Efficient Filters, on page 23

• PDFs and Regular Expressions, on page 24

Using Regular Expressions to Filter Messages

You can use filters to search for strings and patterns in non-ASCII encoded message content (both headers and bodies). Specifically, the system supports regular expression (regex) searching for non-ASCII character sets within:

- Message headers
- MIME attachment filename strings
- Message body:
  - Bodies without MIME headers (i.e. traditional email)
  - Bodies with MIME headers indicating encoding but no MIME parts
  - Multi-part MIME messages with encoding indicated
  - All of the above without the encoding specified in a MIME header

You can use regular expressions (regexes) to match on any part of the message or body, including matching attachments. The various attachment types include text, HTML, MS Word, Excel, and others. Examples of character sets of interest include gb2312, HZ, EUC, JIS, Shift-JIS, Big5, and Unicode. Message filter rules with regular expressions can be created through the content filter GUI, or using a text editor to generate a file that is then imported into the system. For more information, see Using the CLI to Manage Message Filters, on page 98 and Configuring Scan Behavior, on page 120.

Guidelines for Using Regular Expressions

It is important to begin a regular expression with a caret (^) and end it with a dollar sign ($) whenever you want to exactly match a string and not a prefix.

Note When matching an empty string, do not use “” as that actually matches *all* strings. Instead use `"^$"` . For an example, see the second example in Subject Rule, on page 27.

It is also important to remember that if you want to match a literal period, you must use an escaped period in the regular expression. For example, the regular expression sun.com matches the string `thegodsunocommando` , but the regular expression `^sun\.com$` only matched the string `sun.com`.

Technically, the style of regular expressions used are **Python re Module** style regular expressions. For a more detailed discussion of Python style regular expressions, consult the Python Regular Expression HOWTO, accessible from: `http://www.python.org/doc/howto/`

Regular Expression and Non-ASCII Character Sets

In some languages, the concepts of a word or word boundary, or case do not exist.

Complex regular expressions that depend on concepts like what is or is not a character that would compose a word (represented as “ \w ” in regex syntax) cause problems when the locale is unknown or if the encoding is not known for certain.

n Tests

Regular expressions can be tested for matching using the sequence == and for non-matching using the sequence != . For example:

```
rcpt-to ==
 "^goober@dev\\.null\\....$" (matching)


rcpt-to != "^goober@dev\\.null\\....$" (non-matching)
```

Case-sensitivity

Unless otherwise noted, regular expressions are case-sensitive. Thus, if your regular expression is searching for `foo` , it does not match the pattern `FOO` or even `Foo` .

Writing Efficient Filters

This example shows two filters that do the same thing, but the first one takes much more CPU. The second filter uses a regular expression that is more efficient.

```
attachment-filter: if ((recv-listener == "Inbound") AND
((((((((((((((((((((((((((((((((((((((((((((((((((attachment-filename ==


"\\.386$") OR (attachment-filename == "\\.exe$")) OR (attachment-filename == "\\.ad$")) OR

(attachment-filename == "\\.ade$")) OR (attachment-filename == "\\.adp$")) OR
(attachment-filename == "\\.asp$")) OR (attachment-filename == "\\.bas$")) OR
(attachment-filename == "\\.bat$")) OR (attachment-filename == "\\.chm$")) OR
(attachment-filename == "\\.cmd$")) OR (attachment-filename == "\\.com$")) OR
(attachment-filename == "\\.cpl$")) OR (attachment-filename == "\\.crt$")) OR
(attachment-filename == "\\.exe$")) OR (attachment-filename == "\\.hlp$")) OR
(attachment-filename == "\\.hta$")) OR (attachment-filename == "\\.inf$")) OR
(attachment-filename == "\\.ins$")) OR (attachment- filename == "\\.isp$")) OR
(attachment-filename == "\\.js$")) OR (attachment-filename == "\\.jse$")) OR
(attachment- filename == "\\.lnk$")) OR (attachment-filename == "\\.mdb$")) OR
(attachment-filename == "\\.mde$")) OR (attachment-filename == "\\.msc$")) OR
(attachment-filename == "\\.msi$")) OR (attachment-filename == "\\.msp$")) OR
(attachment-filename == "\\.mst$")) OR (attachment-filename == "\\.pcd$")) OR
(attachment-filename == "\\.pif$")) OR (attachment-filename == "\\.reg$")) OR
(attachment-filename == "\\.scr$")) OR (attachment-filename == "\\.sct$")) OR
(attachment-filename == "\\.shb$")) OR (attachment-filename == "\\.shs$")) OR
(attachment-filename == "\\.url$")) OR (attachment-filename == "\\.vb$")) OR
(attachment-filename == "\\.vbe$")) OR (attachment-filename == "\\.vbs$")) OR
(attachment-filename == "\\.vss$")) OR (attachment-filename == "\\.vst$")) OR
(attachment-filename == "\\.vsw$")) OR (attachment-filename == "\\.ws$")) OR
(attachment-filename == "\\.wsc$")) OR (attachment-filename == "\\.wsf$")) OR
(attachment-filename == "\\.wsh$"))) { bounce(); }
```

In this instance, AsyncOS will have to start the regular expression engine 30 times, once for each attachment type and the recv-listener.

Instead, write the filter to look like this:

```
attachment-filter: if (recv-listener == "Inbound") AND (attachment-filename == "\\.
(386|exe|ad|ade|adp|asp|bas|bat|chm|cmd|com|cpl|crt|exe|hlp|hta|inf|ins|isp|js|jse|l
nk|mdb|mde|msc|msi|msp|mst|pcd|pif|reg|scr|sct|shb|shs|
url|vb|vbe|vbs|vss|vst|vsw|ws|wsc|wsf|wsh)$") {
```

The regular expression engine only has to start twice and the filter is arguably easier to maintain as you do not have to worry about adding "()", spelling errors. In contrast to the above, this should show a decrease in CPU overhead.

PDFs and Regular Expressions

Depending on how a PDF is generated, it may contain no spaces or line breaks. When this occurs, the scanning engine attempts to insert logical spaces and line breaks based on the location of the words on the page. For example, when a word is constructed using multiple fonts or font sizes, the PDF code is rendered in a way that makes it difficult for the scanning engine to determine word and line breaks. When you attempt to match a regular expression against a PDF file constructed in this way, the scanning engine may return unexpected results.

For example, you enter a word in a PowerPoint document that uses different fonts and different font sizes for each letter in the word. When a scanning engine reads a PDF generated from this application, it inserts logical spaces and line breaks. Because of the construction of the PDF, it may interpret the word "callout" as "call out" or "c a l lout." Attempting to match either of these renderings against the regular expression, "callout," would result in no matches.

Smart Identifiers

When you use message rules that scan message content, you can use smart identifiers to detect certain patterns in the data.

Smart identifiers can detect the following patterns in data:

- Credit card numbers
- U.S. Social Security numbers
- Committee on Uniform Security Identification Procedures (CUSIP) numbers
- American Banking Association (ABA) routing numbers

To use smart identifiers in a filter, enter the following keywords in a filter rule that scans body or attachment content:

Table 4: Smart Identifiers in Message Filters

Key Word	Smart Identifier	Description
*credit	Credit card number	Identifies 14-, 15-, and 16- digit credit card numbers. NOTE: The smart identifier does not identify enRoute cards.
*aba	ABA routing number	Identifies ABA routing numbers.
*ssn	Social security number	Identifies U.S. social security numbers. The *ssn smart identifier identifies social security numbers with dashes, periods and spaces.
*cusip	CUSIP number	Identifies CUSIP numbers.

Related Topics

- Smart Identifier Syntax, on page 25

Smart Identifier Syntax

When you use a smart identifier in a filter rule, enter the smart-identifier keyword in quotes within a filter rule that scans the body or attachment file, as in the example below:

```
ID_Credit_Cards:

if(body-contains("*credit")){

notify("legaldept@example.com");

}
.
```

You can also use smart identifiers in content filters and as a part of content dictionaries.

Note You cannot combine a smart identifier key word with a normal regular expression or another key word. For example the pattern `*credit|*ssn` would not be valid.

Note To minimize on false positives using the *SSN smart identifier, it may be helpful to use the *ssn smart identifier along with other filter criteria. One example filter that can be used is the "only-body-contains" filter condition. This will only evaluate the expression to be true if the search string is present in all of the message body mime parts. For example, you could create the following filter:

```
SSN-nohtml: if only-body-contains("*ssn") { duplicate-quarantine("Policy");}
```

Description and Examples of Message Filter Rules

The following section describes the various message filter rules in use and their examples.

Related Topics

- True Rule, on page 26
- Valid Rule, on page 27
- Subject Rule, on page 27
- Envelope Recipient Rule, on page 27
- Envelope Recipient in Group Rule, on page 28
- Envelope Sender Rule, on page 28
- Envelope Sender in Group Rule, on page 29
- Sender Group Rule, on page 29
- Body Size Rule, on page 29
- Remote IP Rule, on page 30
- Receiving Listener Rule, on page 30
- Receiving IP Interface Rule, on page 31
- Date Rule, on page 31
- Header Rule, on page 31

• Random Rule, on page 32
• Recipient Count Rule, on page 33
• Address Count Rule, on page 33
• Body Scanning Rule, on page 33
• Body Scanning , on page 34
• Encryption Detection Rule, on page 34
• Attachment Type Rule, on page 35
• Attachment Filename Rule, on page 35
• DNS List Rule, on page 36
• SenderBase Reputation Rule, on page 37
• Dictionary Rules, on page 37
• SPF-Status Rule, on page 39
• SPF-Passed Rule, on page 41
• S/MIME Gateway Message Rule, on page 41
• S/MIME Gateway Verified Rule, on page 41
• Workqueue-count Rule, on page 41
• SMTP Authenticated User Match Rule, on page 42
• Signed Rule, on page 43
• Header Repeats Rule, on page 46
• URL Reputation Rules , on page 48
• URL Category Rule , on page 49
• Corrupt Attachment Rule, on page 49
• Message Language Rule, on page 49
• Macro Detection Rule, on page 50
• Forged Email Detection Rule, on page 51
• Duplicate Boundaries Verification Rule, on page 52
• Malformed MIME Header Detection Rule, on page 52
• Geolocation Rule, on page 53
• Domain Reputation Rule for ETF, on page 53
• Domain Reputation Rule for SDR, on page 53

True Rule

The true rule matches all messages. For example, the following rule changes the IP interface to external for all messages it tests.

```
externalFilter:

   if (true)

   {

        alt-src-host('external');

   }
```

Valid Rule

The `valid` rule returns false if the message contains unparsable/invalid MIME parts and true otherwise. For example, the following rule drops all unparsable messages it tests.

```
not-valid-mime:

if not valid

{

drop();

}
```

Subject Rule

The `subject` rule selects those messages where the value of the subject header matches the given regular expression.

For example, the following filter discards all messages with subjects that start with the phrase Make Money...

```
not-valid-mime:

if not valid

{

drop();

}
```

You can specify non-ASCII characters to search for in the value of the header.

When working with headers, remember that the current value of the header includes changes made during processing (such as with filter actions that add, remove, or modify message headings). See Message Header Rules and Evaluation, on page 5 for more information.

The following filter returns true if the headers are empty or if the headers are missing from the message:

```
EmptySubject_To_filter:

if (header('Subject') != ".") OR

(header('To') != ".") {

drop();

}
```

Note This filter returns true for empty Subject and To headers, but it also returns true for missing headers. If the message does not contain the specified headers, the filter still returns true.

Envelope Recipient Rule

The `rcpt-to` rule selects those messages where any Envelope Recipient matches the given regular expression. For example, the following filter drops all messages sent with an email address containing the string “scarface.”

Note The regular expression for the rcpt-to rule is case *insensitive* .

```
scarfaceFilter:

if (rcpt-to == 'scarface')

{

drop();

}
```

Note The `rcpt-to` rule is message-based. If a message has multiple recipients, only one recipient has to match the rule for the specified action to affect the message to all recipients.

Envelope Recipient in Group Rule

The `rcpt-to-group` rule selects those messages where any Envelope Recipient is found to be a member of the LDAP group given. For example, the following filter drops all messages sent with an email address within the LDAP group "ExpiredAccounts."

```
expiredFilter:

if (rcpt-to-group == 'ExpiredAccounts')

{

drop();

}
```

Note The `rcpt-to-group` rule is message-based. If a message has multiple recipients, only one recipient has to match the rule for the specified action to affect the message to all recipients.

Envelope Sender Rule

The `mail-from` rule selects those messages where the Envelope Sender matches the given regular expression. For example, the following filter immediately delivers any message sent by `admin@yourdomain.com` .

Note The regular expression for the mail-from rule is case *insensitive* . Note that the period character is escaped in the following example.

```
kremFilter:

if (mail-from == '^admin@yourdomain\\.com$')

{
```

```
skip-filters();

}
```

Envelope Sender in Group Rule

The `mail-from-group` rule selects those messages where the Envelope Sender is found to be in an LDAP group given on the right side of the operator (or, in the case of inequality, where the sender's email address is *not* in the given LDAP group). For example, the following filter immediately delivers any message sent by someone whose email address is in the LDAP group "KnownSenders."

```
SenderLDAPGroupFilter:

if (mail-from-group == 'KnownSenders')

{

skip-filters();

}
```

Sender Group Rule

The `sendergroup` message filter selects a message based on which sender group was matched in a listener's Host Access Table (HAT). This rule uses '==' (for matching) or '!=' (for not matching) to test for matching a given regular expression (the right side of the expression). For example, the following message filter rule evaluates to `true` if the sender group of the message matches the regular expression Internal, and, if so, sends the message to an alternate mail host.

```
senderGroupFilter:

if (sendergroup == "Internal")

{

alt-mailhost("[172.17.0.1]");

}
```

Body Size Rule

Body size refers to the size of the message, including both headers and attachments. The `body-size` rule selects those messages where the body size compares as directed to a given number. For example, the following filter bounces any message where the body size is greater than 5 megabytes.

```
BigFilter:

if (body-size > 5M)

{

bounce();

}
```

The `body-size` can be compared in the following ways:

Example	Comparison Type
`body-size < 10M`	Less than

Example	Comparison Type
`body-size <= 10M`	Less than or equal
`body-size > 10M`	Greater than
`body-size >= 10M`	Greater than or equal
`body-size == 10M`	Equal
`body-size != 10M`	Not equal

As a convenience, the size measurement may be specified with a suffix:

Quantity	Description
`10b`	ten bytes (same as 10)
`13k`	thirteen kilobytes
`5M`	five megabytes
`40G`	40 gigabytes (Note: The appliance cannot accept messages larger than 100 megabytes.)

Remote IP Rule

The `remote-ip` rule tests to see if the IP address of the host that sent that message matches a certain pattern. The IP address can be either Internet Protocol version 4 (IPv4) or Internet Protocol version 6 (IPv6). The IP address pattern is specified using the allowed hosts notation described in “Sender Group Syntax”, except for the `SBO , SBRS , dnslist` notations and the special keyword `ALL` .

The allowed hosts notation can only identify sequences and numeric ranges of IP addresses (not hostnames). For example, the following filter bounces any message not injected from IP addresses of form `10.1.1. x` where X is `50 , 51 , 52 , 53 , 54 , or 55` .

```
notMineFilter:

if (remote-ip != '10.1.1.50-55')

{

bounce();

}
```

Receiving Listener Rule

The `recv-listener` rule selects those messages received on the named listener. The listener name must be the nickname of one of the listeners currently configured on the system. For example, the following filter immediately delivers any message arriving from the listener named expedite .

```
expediteFilter:

if (recv-listener == 'expedite')

{

skip-filters();

}
```

Receiving IP Interface Rule

The `recv-int` rule selects those messages received via the named interface. The interface name must be the nickname of one of the interfaces currently configured for the system. For example, the following filter bounces any message arriving from the interface named outside .

```
outsideFilter:

if (recv-int == 'outside')

{

bounce();

}
```

Date Rule

The `date` rule checks the current time and date against a time and date you specify. The date rule is compares against a string containing a timestamp of the format *MM/DD/YYYY hh:mm:ss* . This is useful to specify actions to be performed before or after certain times in US format. (Note that there may be an issue if you are searching messages with non-US date formats.) the following filter bounces all messages from `campaign1@yourdomain.com` that are injected after 1:00pm on July 28th, 2003:

```
TimeOutFilter:

if ((date > '07/28/2003 13:00:00') and (mail-from ==

'campaign1@yourdomain\\.com'))

{

bounce();

}
```

Note Do not confuse the `date` rule with the `$Date` message filter action variable.

Header Rule

The `header()` rule checks the message headers for a specific header, which must be specified quoted in parentheses ("*header name* "). This rule may be compared to a regular expression, much like the subject rule, or may be used without any comparison, in which case it will be "true" if the header is found in the message, and "false" if it is not found. For example, the following example checks to see if the header `X-Sample` is found, and if its value contains the string " `sample text` ". If a match is made, the message is bounced.

```
FooHeaderFilter:

if (header('X-Sample') == 'sample text')

{

bounce();

}
```

You can specify non-ASCII characters to search for in the value of the header.

The following example demonstrates the header rule without a comparison. In this case, if the header `X-DeleteMe` is found, it is removed from the message.

```
DeleteMeHeaderFilter:

if header('X-DeleteMe')

{

strip-header('X-DeleteMe');

}
```

When working with headers, remember that the current value of the header includes changes made during processing (such as with filter actions that add, remove, or modify message headings). See Message Header Rules and Evaluation, on page 5 for more information.

Random Rule

The `random` rule generates a random number from zero to N-1, where N is the integer value supplied in parenthesis after the rule. Like the `header()` rule, this rule may be used in a comparison, or may be used alone in a “unary” form. The rule evaluates to `true` in the unary form if the random number generated is non-zero. For example, both of the following filters are effectively equal, choosing Virtual Gateway address A half the time, and Virtual Gateway address B the other half of the time:

```
load_balance_a:

if (random(10) < 5)
{

alt-src-host('interface_a');
}

else

{

alt-src-host('interface_b');

}

load_balance_b:

if (random(2))

{

alt-src-host('interface_a');

}
```

```
else

{

alt-src-host('interface_b');

}
```

Recipient Count Rule

The `rcpt-count` rule compares the number of recipients of a message against an integer value, in a similar way to the `body-size` rule. This can be useful for preventing users from sending email to large numbers of recipients at once, or for ensuring that such large mailing campaigns go out over a certain Virtual Gateway address. The following example sends any email with more than 100 recipients over a specific Virtual Gateway address:

```
large_list_filter:

if (rcpt-count > 100) {

alt-src-host('mass_mailing_interface');

}
```

Address Count Rule

The `addr-count()` message filter rule takes one or more header strings, counts the number of recipients in each line and reports the cumulative number of recipients. This filter differs from the `rcpt-count` filter rule in that it operates on the message body headers instead of the envelope recipients. The following example shows the filter rule used to replace a long list of recipients with the alias, “undisclosed-recipients”:

```
large_list_filter:


if (rcpt-count > 100) {

alt-src-host('mass_mailing_interface');

}
```

Body Scanning Rule

The `body-contains()` rule scans the incoming email and all its attachments for a particular pattern defined by its parameter. This includes delivery-status parts and associated attachments. The `body-contains()` rule does not perform multi-line matching. The scanning logic can be modified on the Scan Behavior page or using the `scanconfig` command in the CLI to define which MIME types should or should not be scanned. You can also specify a minimum number of matches that the scanning engine must find in order for the scan to evaluate to true.

By default, the system scans all attachments *except* for those with a MIME type of `video/* , audio/* , image/*` . The system scans archive attachments — `.zip , .bzip , .compress , .tar , or .gzip` attachments containing multiple files. You can set the number of “nested” archived attachments to scan (for example, a `.zip` contained within a `.zip`).

For more information, see Configuring Scan Behavior, on page 120.

Body Scanning

When AsyncOS performs body scanning, it scans the body text and attachments for the regular expression. You can assign a minimum threshold value for the expression, and if the scanning engine encounters the regular expression the minimum number of times, the expression evaluates to `true` .

AsyncOS evaluates the different MIME parts of the message, and it scans any MIME part that is textual. AsyncOS identifies the text parts if the MIME type specifies text in the first part. AsyncOS determines the encoding based on the encoding specified in the message, and it converts the text to Unicode. It then searches for the regular expression in Unicode space. If no encoding is specified in the message, AsyncOS uses the encoding you specify on the Scan Behavior page or using the `scanconfig` command.

For more information about how AsyncOS evaluates MIME parts when scanning messages, see Message Bodies vs. Message Attachments, on page 5.

If the MIME part is not textual, AsyncOS extract files from a .zip or .tar archive or decompresses compressed files. After extracting the data, a scanning engine identifies the encoding for the file and returns the data from the file in Unicode. AsyncOS then searches for the regular expression in Unicode space.

The following example searches the body text and attachment for the phrase “Company Confidential.” The example specifies a minimum threshold of two instances, so if the scanning engine finds two or more instances of the phrase, it bounces any matching messages, and notifies the legal department of the attempt:

```
ConfidentialFilter:

if (body-contains('Company Confidential',2)) {

notify ('legaldept@example.domain');

bounce();

}
```

To scan only the body of the message, use `only-body-contains` :

```
disclaimer:

if (not only-body-contains('[dD]isclaimer',1) ) {

notify('hresource@example.com');

}
```

Encryption Detection Rule

The `encrypted` rule examines the contents of a message for encrypted data. It does not attempt to decode the encrypted data, but merely examines the contents of the message for the existence of encrypted data. This can be useful for preventing users from sending encrypted email.

Note The encrypted rule can only detect encrypted data in the content of messages. It does not detect encrypted attachments.

The encrypted rule is similar to the `true` rule in that it takes no parameters and cannot be compared. This rule returns `true` if encrypted data is found and `false` if no encrypted data is found. Because this function requires the message to be scanned, it uses the scanning settings you define on the Scan Behavior page or using the

`scanconfig` command. For more information about configuring these options, see Configuring Scan Behavior, on page 120.

The following filter checks all email sent through the listener, and if a message contains encrypted data, the message is blind-carbon-copied to the legal department and then bounced:

```
prevent_encrypted_data:

if (encrypted) {

bcc ('legaldept@example.domain');

bounce();

}
```

Attachment Type Rule

The `attachment-type` rule checks the MIME types of each attachment in a message to see if it matches the given pattern. The pattern must be of the same form used in the Scan Behavior page or the `scanconfig` command, as described in Configuring Scan Behavior, on page 120, and so may have either side of the slash (/) replaced by an asterisk as a wildcard. If the message contains an attachment that matches this specified MIME type, this rule returns “true.”

Because this function requires the message to be scanned, it obeys all of the options described in Configuring Scan Behavior, on page 120.

See Attachment Scanning, on page 87 for more information on message filter rules you can use to manipulate attachments to messages.

The following filter checks all email sent through the listener, and if a message contains an attachment with a MIME type of `video/*` , the message is bounced:

```
bounce_video_clips:

if (attachment-type == 'video/*') {

bounce();

}
```

Attachment Filename Rule

The `attachment-filename` rule checks the filenames of each attachment in a message to see if it matches the given regular expression. This comparison is case-sensitive. The comparison is, however sensitive to whitespace so if the filename has encoded whitespace at the end, the filter will skip the attachment. If one of the message’s attachments matches the filename, this rule returns “true.”

Please note the following points:

- Each attachment’s filename is captured from the MIME headers. The filename in the MIME header may contain trailing spaces.
- If an attachment is an archive, the appliance will harvest the filenames from inside the archive, and apply scan configuration rules (see Configuring Scan Behavior, on page 120) accordingly.
    - If the attachment is a single compressed file (despite the file extension), it is not considered an archive and the filename of the compressed file is not harvested. This means that the file is not

processed by the `attachment-filename` rule. An example of this type of file is an executable file (.exe) compressed with `gzip` .

- For attachments consisting of a single compressed file, such as foo.exe.gz, use regular expression to search for specific file types within compressed files. See Attachment Filenames and Single Compressed Files within Archive Files, on page 36.

See Attachment Scanning, on page 87 for more information on message filter rules you can use to manipulate attachments to messages.

The following filter checks all email sent through the listener, and if a message contains an attachment with a filename `*.mp3` , the message is bounced:

```
block_mp3s:

if (attachment-filename == '(?i)\\.mp3$') {

bounce();

}
```

Related Topics

- Attachment Filenames and Single Compressed Files within Archive Files, on page 36

Attachment Filenames and Single Compressed Files within Archive Files

This example shows how to match single compressed files in archives such as those created by gzip :

```
quarantine_gzipped_exe_or_pif:

if (attachment-filename == '(?i)\\.(exe|pif)($|.gz$)') {

quarantine("Policy");

}
```

DNS List Rule

The `dnslist()` rule queries a public DNS List server that uses the DNSBL method (sometimes called “ip4r lookups”) of querying. The IP address of the incoming connection is reversed (so an IP of 1.2.3.4 becomes 4.3.2.1) and then added as a prefix to the server name in the parenthesis (a period to separate the two is added if the server name does not start with one). A DNS query is made, and the system is returned with either a DNS failure response (indicating the connection's IP address was not found in the server's list) or an IP address (indicating that the address was found). The IP address returned is *usually* of the form `127.0.0. x` where `x` can be almost any number from 0 to 255 (IP address ranges are not allowed). Some servers actually return different numbers based on the reason for the listing, while others return the same result for all matches.

Like the `header()` rule, `dnslist()` can be used in either a unary or binary comparison. By itself, it simply evaluates to `true` if a response is received and `false` if no response is received (for example, if the DNS server is unreachable).

The following filter immediately delivers a message if the sender has been bonded with the Cisco Bonded Sender information services program:

```
allowedlist_bondedsender:

if (dnslist('query.bondedsender.org')) {
```

```
skip-filters();

}
```

Optionally, you can compare the result to a string using the equality (==) or inequality (!=) expressions.

The following filter drops a message that results in a " 127.0.0.2 " response from the server. If the response is anything else, the rule returns "false" and the filter is ignored.

```
blockedlist:

if (dnslist('dnsbl.example.domain') == '127.0.0.2') {

drop();

}
```

SenderBase Reputation Rule

The reputation rule checks the SenderBase Reputation Score against another value. All the comparison operators are allowed, such as > , == , <=, and so forth. If the message does not have a SenderBase Reputation Score at all (because one was never checked for it, or because the system failed to get a response from the SenderBase Reputation Service query server), any comparison against a reputation fails (the number will not be greater than, less than, equal to, or not equal to any value). You can check for a SBRS score of "none" using the no-reputation rule described below. The following example adjusts the "Subject:" line of a message to be prefixed by " *** BadRep *** " if the reputation score returned from the SenderBase Reputation Service is below a threshold of -7.5..

```
note_bad_reps:

if (reputation < -7.5) {
strip-header ('Subject');
insert-header ('Subject', '*** BadRep $Reputation *** $Subject');
}
```

For more information, see the "Sender Reputation Filtering" chapter. See also Bypass Anti-Spam System Action, on page 81

Values for the SenderBase Reputation rule are -10 through 10, but the value NONE may also be returned. To check specifically for the value NONE , use the no-reputation rule.

```
none_rep:

if (no-reputation) {

strip-header ('Subject');

insert-header ('Subject', '*** Reputation = NONE *** $Subject');

}
```

Dictionary Rules

The dictionary-match(< *dictonary_name* >) rule evaluates to true if the message body contains any of the regular expressions or terms in the content dictionary named "*dictonary_name* ." If the dictionary does

not exist, the rule evaluates to `false` . For more information on defining dictionaries (including their case sensitivity and word boundary settings), see the "Text Resources" chapter.

The following filter blind carbon copies the administrator when the Cisco scans a message that contains any words within the dictionary named "secret_words."

```
copy_codenames:

if (dictionary-match ('secret_words')) {

bcc('administrator@example.com');

}
```

The following example sends the message to the Policy quarantine if the message body contains any words within the dictionary named "secret_words." Unlike the `only-body-contains` condition, the `body-dictionary-match` condition does not require that all the content parts individually match the dictionary. The scores of each content part (taking into account multipart/alternative parts) are added together.

```
quarantine_data_loss_prevention:

if (body-dictionary-match ('secret_words'))

{

quarantine('Policy');

}
```

In the following filter, a subject that matches a term in the specified dictionary is quarantined:

```
quarantine_policy_subject:

if (subject-dictionary-match ('gTest'))

{

quarantine('Policy');

}
```

This example matches an email address in the "to" header and blind copies an administrator:

```
headerTest:

if (header-dictionary-match ('competitorsList', 'to'))

{

bcc('administrator@example.com');

}
```

The `attachment-dictionary-match(<dictonary_name>)` rule works like the `dictionary-match` rule above, except that it looks for matches in the attachment.

The following filter sends the message to the Policy quarantine if the message attachment contains any words found within the dictionary named "secret_words."

```
quarantine_codenames_attachment:
```

```
if (attachment-dictionary-match ('secret_words'))

{

quarantine('Policy');

}
```

The `header-dictionary-match(<dictonary_name>, <header>)` rule works like the dictionary-match rule above, except that it looks for matches in the header specified in <*header*>. The header name is case insensitive, so, for example, “subject” and “Subject” both work.

The following filter sends the message to the Policy quarantine if the message’s “cc” header contains any words found within the dictionary named “ex_employees.”

```
quarantine_codenames_attachment:

if (header-dictionary-match ('ex_employees', 'cc'))

{

quarantine('Policy');

}
```

You can use wild cards within the dictionary terms. You do not have to escape the period in email addresses.

SPF-Status Rule

When you receive SPF/SIDF verified mail, you may want to take different actions depending on the results of the SPF/SIDF verification. The spf-status rule checks against different SPF verification results. For more information, see Verification Results.

Note If you have configured an SPF verification message filter rule without an SPF identity and if a message contains different SPF identities with different verdicts, the rule is triggered if one of the verdicts in the message matches the rule.

You can check against the SPF/SIDF verification results using the following syntax:

```
if (spf-status == "Pass")
```

If you want a single condition to check against multiple status verdicts, you can use the following syntax:

```
if (spf-status == "PermError, TempError")
```

You can also check the verification results against the HELO, MAIL FROM, and PRA identities using the following syntax:

```
if (spf-status("pra") == "Fail")
```

The following example shows the spf-status filter in use:

```
skip-spam-check-for-verified-senders:
```

```
if (sendergroup == "TRUSTED" and spf-status == "Pass"){
skip-spamcheck();
}
quarantine-spf-failed-mail:
if (spf-status("pra") == "Fail") {
if (spf-status("mailfrom") == "Fail"){
# completely malicious mail
quarantine("Policy");
} else {
if(spf-status("mailfrom") == "SoftFail") {
# malicious mail, but tempting
quarantine("Policy");
}
}
} else {
if(spf-status("pra") == "SoftFail"){
if (spf-status("mailfrom") == "Fail"
or spf-status("mailfrom") == "SoftFail"){
# malicious mail, but tempting
quarantine("Policy");
}
}
}
stamp-mail-with-spf-verification-error:
if (spf-status("pra") == "PermError, TempError"

or spf-status("mailfrom") == "PermError, TempError"
or spf-status("helo") == "PermError, TempError"){
# permanent error - stamp message subject
strip-header("Subject");
insert-header("Subject", "[POTENTIAL PHISHING] $Subject"); }
.
```

SPF-Passed Rule

The following example shows an `spf-passed` rule used to quarantine emails that are not marked as spf-passed:

```
quarantine-spf-unauthorized-mail:

if (not spf-passed) {

quarantine("Policy");

}
```

Note Unlike the `spf-status` rule, the spf-passed rule reduces the SPF/SIDF verification values to a simple Boolean. The following verification results are treated as not passed in the `spf-passed` rule: None, Neutral, Softfail, TempError, PermError, and Fail. To perform actions on messages based on more granular results, use the `spf-status` rule.

S/MIME Gateway Message Rule

The S/MIME Gateway Message rule checks if a message is S/MIME signed, encrypted, or signed and encrypted. The following message filter checks if the message is an S/MIME message and quarantines it if the verification or decryption using S/MIME fails.

```
quarantine_smime_messages:
if (smime-gateway-message and not smime-gateway-verified) {
quarantine("Policy");
}
```

For more information, see S/MIME Security Services.

S/MIME Gateway Verified Rule

The S/MIME Gateway Message Verified rule checks if a message is successfully verified, decrypted, or decrypted and verified. The following message filter checks if the message is an S/MIME message and quarantines it if the verification or decryption using S/MIME fails.

```
quarantine_smime_messages:
if (smime-gateway-message and not smime-gateway-verified) {
quarantine("Policy");
}
```

For more information, see S/MIME Security Services

Workqueue-count Rule

The `workqueue-count` rule checks the workqueue-count against a specified value. All the comparison operators are allowed, such as `>` , `==` , `<=`, and so forth.

The following filter checks the workqueue count, and skips spam check if the queue is greater than the specified number.

```
wqfull:

if (workqueue-count > 1000) {
```

```
skip-spamcheck();

}
```

For more information on SPF/SIDF, see Overview of SPF and SIDF Verification.

SMTP Authenticated User Match Rule

If your appliance uses SMTP authentication to send messages, the `smtp-auth-id-matches` (*<target> [, <sieve-char>]*)rule can check a message's headers and Envelope Sender against the sender's SMTP authenticated user ID to identify outgoing messages with spoofed headers. This filter allows the system to quarantine or block potentially spoofed messages.

The `smtp-auth-id-matches` rule compares the SMTP authenticated ID against the following targets:

Target	Description
*EnvelopeFrom	Compares the address of the Envelope Sender (also known as MAIL FROM) in the SMTP conversation
*FromAddress	Compares the addresses parsed out of the From header. Since multiple addresses are permitted in the From: header, only one has to match.
*Sender	Compares the address specified in the Sender header.
*Any	Matches messages that were created during an authenticated SMTP session regardless of identity.
*None	Matches messages that were not created during an authenticated SMTP session. This is useful when authentication is optional (preferred).

The filter performs matches loosely. It is not case-sensitive. If the optional *sieve-char* parameter is supplied, the last portion of an address that follows the specified character will be ignored for the purposes of comparison. For example, if the `+` character is included as a parameter, the filter ignores the portion of the address `joe+folder@example.com` that follows the `+` character. If the address was `joe+smith+folder@example.com` , only the `+folder` portion is ignored. If the SMTP authenticated user ID string is a simple username and not a fully-qualified e-mail address, only the username portion of the target will be examined to determine a match. The domain must be verified in a separate rule.

Also, you can use the $SMTPAuthID variable to insert the STMP authenticated user ID into headers.

The following table shows examples of comparisons between the SMTP authenticated ID and email addresses and whether they would match using the `smtp-auth-id-matches` filter rule:

SMTP Auth ID	Sieve Char	Comparison Address	Matches?
someuser		otheruser@example.com	No
someuser		someuser@example.com	Yes
someuser		someuser@another.com	Yes
SomeUser		someuser@example.com	Yes
someuser		someuser+folder@example.com	No

SMTP Auth ID	Sieve Char	Comparison Address	Matches?
someuser	+	someuser+folder@example.com	Yes
someuser@example.com		someuser@forged.com	No
someuser@example.com		someuser@example.com	Yes
SomeUser@example.com		someuser@example.com	Yes

The following filter checks all messages created during an authenticated SMTP session to verify that the addresses in the From header and the Envelope Sender match the SMTP authenticated user ID. If the addresses and the ID match, the filter verifies the domain. If they do not match, the appliance quarantines the message.

```
Msg_Authentication:

if (smtp-auth-id-matches("*Any"))

{

# Always include the original authentication credentials in a

# special header.

insert-header("X-Auth-ID","$SMTPAuthID");

if (smtp-auth-id-matches("*FromAddress", "+") and

smtp-auth-id-matches("*EnvelopeFrom", "+"))

{

# Username matches. Verify the domain

if header('from') != "(?i)@(?:example\\.com|alternate\\.com)" or

mail-from != "(?i)@(?:example\\.com|alternate\\.com)"

{

# User has specified a domain which cannot be authenticated

quarantine("forged");

}

} else {

# User claims to be an completely different user

quarantine("forged");

}

}
```

Signed Rule

The signed rule checks messages for a signature. The rule returns a boolean value to indicate if the message is signed or not. This rule evaluates whether the signature is encoded according to ASN.1 DER encoding rules

and that it conforms to the CMS SignedData Type structure (RFC 3852, Section 5.1.). It does not aim to validate whether the signature matches the content, nor does it check the validity of the certificate.

The following example shows a `signed` rule used to insert headers into a signed message:

```
signedcheck: if signed { insert-header("X-Signed", "True"); }
```

The following example shows a signed rule used to drop attachments from unsigned messages from a certain sender group:

```
Signed: if ((sendergroup == "NOTTRUSTED") AND NOT signed) {

html-convert();

if (attachment_size > 0)

{

drop_attachments("");

}

}
```

Signed Certificate Rule

The signed-certificate rule selects those S/MIME messages where the X.509 certificate issuer or message signer matches the given regular expression. This rule only supports X.509 certificates.

The rule's syntax is signed-certificate (<field> [<operator> <regular expression>]) , where:

- <field> is either the quoted string "issuer" or "signer" ,
- <operator> is either == or != ,
- and <regular expression> is the value for matching the "issuer" or "signer."

If the message is signed using multiple signatures, the rule returns true if any of the issuers or signers match the regular expression. The short form of this rule, signed-certificate("issuer") and signed-certificate("signer") , returns true if the S/MIME message contains an issuer or signer.

Related Topics

- Signer, on page 44
- Issuer, on page 45
- Escaping in Regular Expressions, on page 45
- $CertificateSigners Action Variable, on page 45
- Examples 1, on page 46

Signer

For message signers, the rule extracts the sequence of rfc822Name names from the X.509 certificate's subjectAltName extension. If there is no subjectAltName field in the signing certificate, or this field does not have any rfc822Name names, the signed-certificate("signer") rule evaluates to false. In the rare cases of multiple rfc822Name names, the rule tries to match all of the names to the regular expression and evaluates as true on the first match.

Issuer

The issuer is a non-empty distinguished name in the X.509 certificate. AsyncOS extracts the issuer from the certificate and converts it to an LDAP-UTF8 Unicode string. For example:

- C=US,S=CA,O=IronPort
- C=US,CN=Bob Smith

Since X.509 certificates require the issuer field, signed-certificate("issuer") evaluates whether the S/MIME message contains an X.509 certificate.

Escaping in Regular Expressions

LDAP-UTF8 defines a mechanism for escaping that you can use in your regular expressions. For a detailed discussion on escaping characters in LDAP-UTF8, consult Lightweight Directory Access Protocol (LDAP): String Representation of Distinguished Names, accessible from http://www.ietf.org/rfc/rfc4514.txt .

The escaping rules for the signed-certificate rule's regular expressions differ from the escaping rules defined in LDAP-UTF8 by limiting escaping to only the characters that require escaping. LDAP-UTF8 allows optional escaping for characters that can be represented without escaping. For example, the following two strings are considered correct for "Example, Inc." using the LDAP-UTF8 escaping rules:

- Example\, Inc.
- Example\,\ Inc\.

However, the signed-certificate rule only matches Example\, Inc. The regular expression does not allow escaping the space and period for matching because these characters do not require escaping, even though it is permitted in LDAP-UTF8. When creating a regular expression for the signed-certificate rule, do not escape a character if it can be represented without escaping.

$CertificateSigners Action Variable

The action variable `$CertificateSigners` is a comma separated list of signers obtained from the `subjectAltName` element of the signing certificate. Multiple email addresses of a single signer will be included in the list with duplicates removed.

For example, Alice signs a message with her two certificates. Bob signs the message with his single certificate. All certificates are issued by a single corporate authority. After the message passes the S/MIME scan, the extracted data contain three items:

```
[

{

'issuer': 'CN=Auth,O=Example\, Inc.',

'signer': ['alice@example.com', 'al@private.example.com']

},

{

'issuer': 'CN=Auth,O=Example\, Inc.',

'signer': ['alice@example.com', 'al@private.example.com']

},

{
```

```
'issuer': 'CN=Auth,O=Example\, Inc.',
'signer': ['bob@example.com', 'bob@private.example.com']
}
]
```

The `$CertificateSigners` variable expands to:

```
"alice@example.com, al@private.example.com, bob@example.com, bob@private.example.com"
```

Examples 1

The following example inserts a new header if the certificate issuer is from the US:

```
Issuer: if signed-certificate("issuer") == "(?i)C=US" {
insert-header("X-Test", "US issuer");
}
```

The following example notifies an administrator if the signer is not from example.com:

```
NotOurSigners: if signed-certificate("signer") AND
signed-certificate("signer") != "example\\.com$" {
notify("admin@example.com");
}
```

The following example adds a header if the message has an X.509 certificate:

```
AnyX509: if signed-certificate ("issuer") {
insert-header("X-Test", "X.509 present");
}
```

The following example adds a header if the message's certificate does not have a signer:

```
NoSigner: if not signed-certificate ("signer") {
insert-header("X-Test", "Old X.509?");
}
```

Header Repeats Rule

The Header Repeats rule evaluates to true if at a given point in time, a specified number of messages:

- With same subject are detected in the last one hour.
- From same envelope sender are detected in the last one hour.

You can use this rule to detect high volume emails. For example, political campaigns through certain websites may send out emails to organizations in high volumes. Anti-spam engines treat such emails as clean, and do not stop the delivery of these emails.

The syntax of this rule is header-repeats (<target>, <threshold> [, <direction>]) , where:

- <target> is subject or mail-from . AsyncOS counts the repetition of values of the target.
- <threshold> is the number of messages with identical values for a given target, received in the last one hour, beyond which the rule evaluates to true.
- <direction> is incoming , outgoing , or both. If direction is not specified in this rule, incoming or outgoing messages are counted for rule evaluation.

Every time when a Header Repeats rule evaluates to true , a System Alert is sent. SeeSystem Alerts.

Note If the header field includes comma or semi-colon separated values, the rule considers the complete string for tracking. This rule ignores messages with empty subject header.

The Header Repeats rule maintains a moving sum of messages with up to one minute's precision. As a result, after the set threshold has reached, there can be a delay of one minute before this rule is triggered.

Related Topics

- Using Header Repeats Rule with Other Rules, on page 47
- Examples, on page 47

Using Header Repeats Rule with Other Rules

You can use the Header Repeats rule with other rules using AND or OR operators. For example, you can categorize an allowed list for a subset of messages using the following filter:

```
F1: if (recv_listener == 'Gray')  AND (header-repeats('subject', X, 'incoming') { drop();}
```

When you use a Header Repeats rule with another rule using AND or OR operators, the Header Repeats rule is evaluated last, and only if needed. If a Header Repeats rule is not evaluated for a given message, subject or mail-from is not counted to compare with the supplied threshold.

As Header Repeats rule is evaluated last and only if needed, the behavior of this rule may vary when used with other rules using an OR operator. The following sample filter uses an OR condition of Signed and Header Repeats rule.

```
f1: if signed OR (header-repeats('subject', 10)) { drop();}
```

In this example, if the first nine messages processed by this filter are signed messages with identical subject, the Header Repeats rule will not process these messages. If the tenth message is an unsigned message with identical subject header as the previous nine messages, the filter will not perform the configured action, even though the threshold has reached.

Examples

In the following example, at any given point in time, if the filter detects X or more incoming messages with identical subject in the last one hour, the subsequent messages with identical subject are sent to Policy quarantine.

```
f1 : if header-repeats('subject', X, 'incoming') { quarantine('Policy');}
```

In the following example, at any given point in time, if the filter detects X or more outgoing messages from same envelope sender in the last one hour, the subsequent messages from the same envelope sender are dropped and discarded.

```
f2 :  if header-repeats('mail-from', X, 'outgoing') {drop();}
```

In the following example, at any given point in time, if the filter detects X or more incoming or outgoing messages with identical subject in the last one hour, the administrator is notified for every subsequent message with identical subject.

```
f3: if header-repeats('subject', X) {notify('admin@xyz.com');}
```

URL Reputation Rules

Use a URL reputation rule to define message actions based on the reputation score of any URL in the message. For important details, see Filtering by URL Reputation or URL Category: Conditions and Rules in Protecting Against Malicious or Undesirable URLs

For these rules:

- `msg_filter_name` : is the name of this message filter.
- `allowedlist` is the name of a defined URL list (via the urllistconfig command.) Specifying an allowed list is optional.

To take action when the reputation service provides a score:

Use the `url-reputation` rule.

Filter syntax when using a url-reputation rule is:

```
<msg_filter_name>:

if url-reputation('<min_score'>, <'max_score'>, '<allowedlist>',
'<include_attachments>','<include_message_body_subject>')

{<action>}
```

Where:

- `min_score` and `max_score` are the minimum and maximum scores in the range for which the action should apply. The values that you specify are included in the range.

Minimum and maximum scores must be between `-10.0` and `10.0` .

- `include_attachments` to scan for URLs in the message attachments. A value of '1' indicates that URL scanning for message attachments is enabled and a value of '0' indicates that URL scanning for message attachments is not enabled.
- `include_message_body_subject` to scan for URLs in the message body and subject. A value of '1' indicates that URL scanning for message body and subject is enabled and a value of '0' indicates that URL scanning for message body and subject is not enabled.

To take action when the reputation service does not provide a score:

Use the `url-no-reputation` rule.

Filter syntax when using a `url-no-reputation` rule is:

```
<msg_filter_name>:
```

```
if url_no_reputation('<allowedlist>',
'<include_attachments>','<include_message_body_subject>')

{<action>}
```

URL Category Rule

Use URL categories to define message actions based on the category of URLs in the message. For important details, see Filtering by URL Reputation or URL Category: Conditions and Rules in Protecting Against Malicious or Undesirable URLs.

Filter syntax when using a `url-category` rule is:

```
<msg_filter_name>: if url-category (['<category-name1>','<category-name2>',...,
'<category-name3>'],'<url_allowed_list>','<include_attachments>','<include_message_body_subject>')

<action>
```

Where:

- `msg_filter_name` is the name of this message filter.
- `action` is any message filter action.
- `category-name` is the URL category. Separate multiple categories with commas. To obtain correct category names, look at a URL Category condition or action in a Content Filter. For descriptions and examples of the categories, see About URL Categories.
- `url_allowed_list` is the name of a defined URL list (via the urllistconfig command.)
- `include_attachments` to scan for URLs in message attachments. A value of '1' indicates that URL scanning for message attachments is enabled and a value of '0' indicates that URL scanning for message attachments is not enabled.
- `include_message_body_subject` to scan for URLs in the message body and subject. A value of '1' indicates that URL scanning for the message body and subject is enabled and a value of '0' indicates that URL scanning for the message body and subjects is not enabled.

Corrupt Attachment Rule

The Corrupt Attachment rule evaluates to true if a message contains corrupt attachment. A corrupt attachment is an attachment that the scanning engine cannot scan and identified as corrupt.

Related Topics

- Example, on page 49

Example

In the following example, if the filter detects a corrupt attachment in a message, the message is quarantined to Policy Quarantine.

```
quar_corrupt_attach: if (attachment-corrupt) { quarantine("Policy"); }
```

Message Language Rule

You may want to take different message actions based on the message language. For example, you may want to:

- Add a disclaimer in Russian to the messages that are in Russian

- Drop the messages whose language could not be determined

Use the message-language rule to take message actions depending on the language of the message subject and body.

Note This rule will not check for the language in attachments and headers.

How Does Language Detection Work

The appliance uses the built-in language detection engine to detect the language in a message. The appliance extracts the subject and the message body and passes it to the language detection engine.

The language detection engine determines the probability of each language in the extracted text and passes it back to the appliance . The appliance considers the language with the highest probability as the language of the message. The appliance considers the language of the message as 'undetermined' in one of the following scenarios:

- If the detected language is not supported by the appliance
- If the appliance is unable to detect the language of the message
- If the total size of the extracted text sent to the language detection engine is less than 50 bytes.

Message Filter Syntax

```
<msg_filter_name>: if (message-language <operator> "<language1>, <language2>,..., <language n>") {<action>}
```

Where:

- `msg_filter_name` is the name of this message filter.
- operator is `==` or `!=` .
- `language` is the value of message language that you want to specify in this message filter. Separate multiple entries with commas. For a list of supported message languages and values, look at the Message Language condition in a content filter. Values are enclosed with brackets ([and]).
- `action` is any message filter action.

Examples

The following example shows how to drop the messages whose language could not be determined:

```
DropMessagesWithUndeterminedLanguage: if (message-language == "unknown") { drop(); }
```

The following example shows how to add a disclaimer in Russian to the messages in Russian:

```
ussianDisclaimerRule: if (message-language == "ru") { add-heading("RussianDisclaimer");
```

Macro Detection Rule

You can use the Macro Detection rule to detect macro-enabled attachments in messages for the specified file types.

Note If an archive or embedded file contains macros, the parent file is dropped from the message.

Macro Detection Syntax

```
<msg_filter_name>: if (macro-detection-rule (['file_type-1', 'file_type-2',...
,'file_type-n'])) {<action>}
```

Where:

- `msg_filter_name` is the name of this message filter.
- `file_type` can be any one of the following supported file types:
  - `Adobe Portable Document Format`
  - `Microsoft Office Files`
  - `OLE File types`
- `action` is any message filter action.

Examples

The following example shows how to drop a message that contains a macro-enabled Microsoft Office attachment:

```
Drop_Messages_With_Macro-enabled_Office_Files: if (macro-detection-rule (['Microsoft Office
Files'])) { drop(); }
```

In the following example, if a message containing a macro-enabled attachment in a PDF format is sent to a specific user, the message is dropped:

```
Strip_Macro_enabled_PDF: if (rcpt-to == "joe@example.com") {
drop-macro-enabled-attachments(['Adobe Portable Document Format']); }
```

Forged Email Detection Rule

You may want to detect fraudulent messages with forged sender address (From: header) and perform actions on such messages.

Use the forged-email-detection rule to detect such messages. While configuring this rule, you must specify a content dictionary and the threshold value (1 through 100) for considering a message as potentially forged.

The forged-email-detection rule compares the From: header with the users in the content dictionary. During this process, depending on the similarity, the appliance assigns similarity score to each of the users in the dictionary. The following are some examples:

- If the From: header is <j0hn.sim0ns@example.com> and the content dictionary contains a user ‘John Simons,’ the appliance assigns a similarity score of 82 to the user.
- If the From: header is <john.simons@diff-example.com> and the content dictionary contains a user ‘John Simons,’ the appliance assigns a similarity score of 100 to the user.

The higher the similarity score, the higher the probability that the message is forged. If the similarity score is greater than or equal to the specified threshold value, the filter action is triggered.

For more information, see Forged Email Detection.

Message Filter Syntax

```
<filter_name>: if (forged-email-detection("<content_dictionary>", threshold)) {<action>;}
```

Where:

- filter_name is the name of the message filter
- content_dictionary is the name of content dictionary
- threshold is the threshold value (1 through 100) for considering a message as potentially forged

Example

The following message filter compares the From: header in the message with the terms in dictionary and if the similarity score of a user in the content dictionary is greater than or equal to 70, the message filter strips the From: header and replaces it with the Envelope Sender.

```
FED_CF: if (forged-email-detection("Execs", 70)) { fed("from", ""); }
```

Duplicate Boundaries Verification Rule

You can use the duplicate_boundaries rule to detect messages that contain duplicate MIME boundaries.

Note Attachment-based rules (for example, attachment-contains) or actions (for example, drop-attachments-where-contains) will not work on malformed messages (with duplicate MIME boundaries).

Message Filter Syntax

```
<filter_name>: if (duplicate_boundaries){<action>;}
```

Example

The following message filter will quarantine all the messages that contain duplicate MIME boundaries.

```
DuplicateBoundaries: if (duplicate_boundaries) { quarantine("Policy"); }
```

Malformed MIME Header Detection Rule

You can use the malformed-header rule to detect messages that contain malformed MIME headers.

Message Filter Syntax

```
<filter_name>: if (malformed-header){<action>;}
```

Example

The following example shows how to quarantine all the messages with malformed MIME headers:

```
quarantine_malformed_headers: if (malformed-header)
{
quarantine("Policy");
}
```

Geolocation Rule

You can use the Geolocation rule to handle incoming messages from particular countries that you select.

Geolocation Syntax

```
<msg_filter_name>: if (geolocation-rule (['country_name-1', 'country_name-2',...
,'country_name-n'])) {<action>}
```

Where:

- `msg_filter_name` is the name of this message filter.
- `country_name` can be name of any country that you select.
- `action` is any message filter action.

Example

The following example shows how to quarantine an incoming message from Country1 and Country2:

```
Quarantine_Incoming_Messages_from_Country1_and_Country2: if (geolocation-rule
(['Country1', 'Country2'])) {quarantine("Policy");}
```

Domain Reputation Rule for ETF

As an example, use the following message filter rule syntax to detect malicious domains in messages using the ETF engine, and take appropriate actions on such messages.

Syntax:

```
quarantine_msg_based_on_ETF: if (domain-external-threat-feeds (['etf_source1'],
 ['mail-from', 'from'], <'domain_exception_list'>)) { quarantine("Policy"); }
```

Where

- `'domain-external-threat-feeds'` is the Domain reputation message filter rule.
- `'etf_source1'` is the ETF source(s) used to detect malicious domain(s) in the header(s) of a message.
- `'mail-from','from'` are the required header(s) used to check for the reputation of the domain.
- `'domain_exception_list'` is the name of a domain exception list. If a domain exception list is not present it is displayed as "".

Example

In the following example, if the domain in the ‘Errors To:’ custom header is detected as malicious by the ETF engine, the message is quarantined.

```
Quaranting_Messages_with_Malicious_Domains: if domain-external-threat-feeds
(['threat_feed_source'], ['Errors-To'], "")) {quarantine("Policy");}
```

Domain Reputation Rule for SDR

You can use the Domain Reputation rule to filter messages based on SDR, and take appropriate actions on such messages:

- Sender Domain Verdict
- Sender Domain Age

- Sender Domain Unscannable

Filtering Messages based on Sender Domain Verdict

Note The recommended blocking threshold is "Poor." For more information about SDR, contact Cisco Talos at https://www.talosintelligence.com.

Syntax:

```
drop_msg_based_on_sdr_verdict:
if sdr-reputation (['awful', 'poor'], "<domain_exception_list>")
{drop();}
```

Where:

- `'drop_msg_based_on_sdr_verdict'` is the name of the message filter.
- `'sdr-reputation'`is the Domain Reputation message filter rule.
- `'awful','poor'` is the range of the sender domain verdict used to filter messages based on SDR.
- `'domain_exception_list'` is the name of a domain exception list. If a domain exception list is not present it is displayed as "".
- `'drop'` is the action applied on the message.

Example

In the following message, if the SDR verdict is 'Unknownr', the message is quarantined.

```
quarantine_unknown_sdr_verdicts:
if sdr-reputation (['unknown'], "")
{quarantine("Policy")}
```

Filtering Messages based on Sender Domain Age

Syntax:

```
<msg_filter_name>
if sdr-age (<‘unit'>, <‘operator'> <‘actual value’>)
{<action>}
```

Where:

- `‘sdr-reputation'` is the Domain Reputation message filter rule.
- `‘sdr_age’` is the age of the sender domain used to filter messages based on SDR.
- `‘unit'` is the number of ‘days,’ ‘years,’ ‘months,’ or 'weeks' option used to filter messages based on the sender domain age.
- `‘operator'` are the following comparison operators used to filter messages based on the sender domain age:
  - – > (Greater than)
  - – >= (Greater than or equal to)
  - – < (Lesser than)

- – <= (Lesser than or equal to)
- – == (Equal to)
- – != (Not equal to)
- – Unknown

- `'actual value'` is the number used to filter messages based on the sender domain age.

Examples

In the following message, if the age of the sender domain is unknown, the message is dropped.

```
Drop_Messages_Based_On_SDR_Age: if (sdr-age ("unknown", "")) {drop();}
```

In the following message, if the age of the sender domain is less than one month, the message is dropped.

```
Drop_Messages_Based_On_SDR_Age: if (sdr-age ("months", <, 1, "")) { drop(); }
```

Filtering Messages based on Sender Domain Unscannable

Syntax:

```
<msg_filter_name>
if sdr-unscannable (<'domain_exception_list'>)
{<action>}
```

Where:

- `'sdr-unscannable'` is the Domain Reputation message filter rule.

  `'domain_exception_list'` is the name of a domain exception list. If a domain exception list is not present it is displayed as "".

Example

In the following message, if the message failed the SDR check, the message is quarantined.

```
Quarantine_Messages_Based_On_Sender_Domain_Unscannable: if (sdr-unscannable (""))
{quarantine("Policy");}
```

Message Filter Actions

The purpose of message filters is to perform actions on selected messages.

The two types of actions are:

- *Final* actions — such as deliver , drop , and bounce — end the processing of a message, and permit no further processing through subsequent filters.
- *Non-final* actions perform an action which permits the message to be processed further.

Note Non-final message filter actions are cumulative. If a message matches multiple filters where each filter specifies a different action, then all actions are accumulated and enforced. However, if a message matches multiple filters specifying the same action, the prior actions are overridden and the final filter action is enforced.

Related Topics

- Filter Actions Summary Table, on page 56
- Action Variables, on page 65
- Matched Content Visibility, on page 67
- Description and Examples of Message Filter Actions, on page 68

Filter Actions Summary Table

Message filters can apply the following actions to an email message as shown in the following table:

Table 5: Message Filter Actions

Action	Syntax	Description
Alter source host	alt-src-host	Change the source hostname and IP interface (Virtual Gateway address) to send the message. See Alter Source Host (Virtual Gateway address) Action, on page 77.
Alter recipient	alt-rcpt-to	Change a recipient of the message. See Alter Recipient Action, on page 76.
Alter mailhost	alt-mailhost	Change the destination mail host for the message. See Alter Delivery Host Action, on page 76.
Notify	notify	Report this message to another recipient. See Notify and Notify-Copy Actions, on page 71.
Notify Copy	notify-copy	Perform just like the notify action, but also sends a copy as with the bcc-scan action. See Notify and Notify-Copy Actions, on page 71.
Blind carbon copy	bcc	Copy this message (message replication) anonymously to another recipient. See Blind Carbon Copy Actions, on page 73.
Blind carbon copy with scan	bcc-scan	Copy this message anonymously to another recipient, and process that message through the work queue as if it were a new message. See Blind Carbon Copy Actions, on page 73.
Archive	archive	Archive this message into an mbox-format file. See Archive Action, on page 77.
Quarantine	`quarantine (`*`quarantine_name`*	Flag this message to be sent to the quarantine named *quarantine_name* . See Quarantine and Duplicate Actions, on page 75.

Action	Syntax	Description
Duplicate (Quarantine)	`duplicate-quarantine (quarantine_name`	Send a copy of the message to the specified quarantine. See Quarantine and Duplicate Actions, on page 75.
Remove headers	strip-header	Remove specified headers from the message before delivering. See Strip Header Action, on page 78.
Insert headers	insert-header	Insert a header and value pair into the message before delivering. See Insert Header Action , on page 78.
Edit header text	edit-header-text	Replace specified header text with a text string you specify in the filter condition. See Edit Header Text Action, on page 79.
Edit body text	edit-body-text()	Strip a regular expression from a message body and replaces it with text that you specify. You might want to use this filter if you want to remove and replace specific content, such as a URL within a message body. See Edit Body Text Action, on page 79.
Convert HTML	html-convert()	Strip HTML tags from message bodies and leaves the plain text content of the message. You might want to use this filter if you want to convert all HTML text in a message to plain text. HTML Convert Action, on page 80.
Assign bounce profile	bounce-profile	Assign a specific bounce profile to the message. See Bounce Profile Action, on page 81.
Bypass Anti-Spam System	skip-spamcheck	Ensure that the anti-spam systems in the Cisco system are *not* applied to this message. See Bypass Anti-Spam System Action, on page 81.
Bypass Graymail Actions	skip-marketingcheck	Bypass actions on marketing emails. See Bypassing Graymail Actions, on page 82.
	skip-socialcheck	Bypass actions on social network emails. See Bypassing Graymail Actions, on page 82.
	skip-bulkcheck	Bypass actions on bulk emails. See Bypassing Graymail Actions, on page 82.

Action	Syntax	Description
Bypass Anti-Virus System	`skip-viruscheck`	Ensure that the anti-virus systems in the Cisco system are *not* applied to this message. See Bypass Anti-Virus System Action, on page 82.
Bypass File Reputation Filtering and File Analysis	`skip-ampcheck`	Ensure that File Reputation Filtering and File Analysis are *not* applied to this message. See Bypass File Reputation Filtering and File Analysis System Actions, on page 83.
Skip Outbreak Filter Scanning	`skip-vofcheck`	Ensure that this message is not processed by the Outbreak Filters scanning. See Bypass Anti-Virus System Action, on page 82.
Drop Attachments by Name	`drop-attachments-by-name`	Drop all attachments on messages that have a filename that match the given regular expression. Archive file attachments (zip, tar), Microsoft Office attachments (doc, docx), and Email attachments (winmail.dat) will be dropped if they contain a file that matches. See Examples of Attachment Scanning Message Filters, on page 94.
Drop Attachments by Type	`drop-attachments-by-type`	Drop all attachments on messages that have a MIME type, determined by either the given MIME type or the file extension. Archive file attachments (zip, tar) will be dropped if they contain a file that matches. See Examples of Attachment Scanning Message Filters, on page 94.
Drop Attachments by File Type	`drop-attachments-by-filetype`	Drop all attachments on messages that match the given “fingerprint” of the file. Archive file attachments (zip, tar) will be dropped if they contain a file that matches. For more information, see Examples of Attachment Scanning Message Filters, on page 94.
Drop Attachments by MIME Type	`drop-attachments-by-mimetype`	Drop all attachments on messages that have a given MIME type. This action does not attempt to ascertain the MIME type by file extension and so it also does not examine the contents of archives. See Examples of Attachment Scanning Message Filters, on page 94.

Action	Syntax	Description
Drop Attachments by Size	`drop-attachments-by-size`	Drop all attachments on the message that, in raw encoded form, are equal to or greater than the size (in bytes) given. Note that for archive or compressed files, this action does not examine the uncompressed size, but rather the size of the actual attachment prior to any decoding. See Examples of Attachment Scanning Message Filters, on page 94.
Drop Attachments by Content	`drop-attachments-where-contains`	Drop all attachments on message that contain the regular expression. Does the pattern occur the minimum number of times you specified for the threshold value? Archive files (zip, tar) will be dropped if any of the files they contain match the regular expression pattern. See Examples of Attachment Scanning Message Filters, on page 94. The optional comment serves as the means to modify the text used to replace the attachment that was dropped. Attachment footers simply append to the message.

Action	Syntax	Description

Drop Attachments with Macro	`drop-macro-enabled-attachments`	Drops all macro-enabled attachments of the specified file type.
		Note If an archive or embedded file contains macros, the parent file is dropped from the message.
		Syntax
		`drop-macro-enabled-attachments (['file_type-1', 'file_type-2', …,'file_type-n'], "custom_replacement_message")`
		Where:
		• `file_type` can be any one of the following supported file types:
		  • `Adobe Portable Document Format`
		  • `Microsoft Office Files`
		  • `OLE File types`
		• **`custom replacement`** message is an optional message to replace the default system generated message added to the bottom of the message body when an attachment is dropped.
		See Macro Detection Rule, on page 50

Drop Attachments by Dictionary Matches	`drop-attachments-where-dictionary-match`	Strip attachments based on matches to dictionary terms. If the terms in the MIME parts considered to be an attachment match a dictionary term (and the user-defined threshold is met), the attachment is stripped from the email. See Examples of Attachment Scanning Message Filters, on page 94.
Add Footer	`add-footer(`*`footer-name`*`)`	Add disclaimer text as a footer to the message. See "Message Disclaimer Stamping" in the "Text Resources" chapter for more information.
Add Heading	`add-heading(`*`heading-name`*`)`	Add disclaimer text as a heading to the message. See "Message Disclaimer Stamping" in the "Text Resources" chapter for more information.

Action	Syntax	Description
Encrypt on Delivery	`encrypt-deferred`	Encrypt message on delivery, which means that the message continues to the next stage of processing, and when all processing is complete, the message is encrypted and delivered.
S/MIME Sign/Encrypt on Delivery	`smime-gateway-deferred ("sending_profile")`	Performs an S/MIME signing or encryption of the message using the specified sending profile during the delivery. See S/MIME Sign or Encrypt on Delivery Action, on page 70.
S/MIME Sign/Encrypt	`smime-gateway("sending_profile")`	Performs an S/MIME signing or encryption using the specified sending profile and delivers the message, skipping any further processing. See S/MIME Sign or Encrypt Action, on page 70.
Add Message Tag	`tag-message(tag-name)`	Add a custom term into the message to use with DLP policy filtering. You can configure a DLP policy to limit scanning to messages with the message tag. The message tag is not visible to recipients. See Add Message Tag Action, on page 83 and the "Data Loss Prevention" chapter.
Add Log Entry	`log-entry`	Adds customized text into the Text Mail logs at the INFO level. The text can include action variables. The log entry appears in message tracking. See Add Log Entry Action, on page 84.
Replace URL with text, based on URL reputation	• `url-reputation-replace` • `url-no-reputation-replace`	Modify URLs or their behavior based on the reputation of the URL. Use a separate action to handle the case in which the reputation service does not provide a score for a URL. See URL Reputation Actions , on page 84.
Defang URL based on URL reputation	• `url-reputation-defang` • `url-no-reputation-defang`	
Redirect URL to a Cisco security proxy, based on URL reputation	• `url-reputation-proxy-redirect` • `url-no-reputation-proxy-redirect`	

Action	Syntax	Description
Replace URL with text, based on URL Category	`url-category-replace`	Modify URLs or their behavior based on the category of the URL. See URL Category Actions , on page 86.
Defang URL based on URL category	`url-category-defang`	
Redirect URL to Cisco security proxy, based on URL category	`url-category-proxy-redirect`	
Forged Email Detection	`fed`	Strips the From: header from the forged message and replaces it with the Envelope Sender. See Forged Email Detection Action, on page 87.
No Operation	`no-op`	No action is performed. See No Operation, on page 87.
*Skip Remaining Message Filters	`skip-filters`	Ensure that this message is not processed by any other message filters and continues through the email pipeline. See Skip Remaining Message Filters Action, on page 69.
*Drop message	`drop`	Drop and discard the message. See Drop Action, on page 69.
*Bounce message	`bounce`	Send the message back to the sender. See Bounce Action, on page 70.
*Encrypt and Deliver Now	`encrypt`	Use Cisco Email Encryption to encrypt outgoing messages. See Encrypt Action, on page 70.
* *Final Actions*		

Related Topics

- Attachment Groups, on page 62

Attachment Groups

You can specify a particular file type (“exe” files for example) or common groups of attachments in the `attachment-filetype` and `drop-attachments-by-filetype` rules . AsyncOS divides the attachments into the groups listed in the following table.

If you create a message filter that uses the `!=` operator to match a message that does not contain an attachment with a specific file type, the filter will not perform any action on the message if there is at least one attachment with the file type you want to filter out. For example, the following filter drops any message with an attachment that is not an **`.exe`** file type:

```
exe_check: if (attachment-filetype != "exe") {

drop();

}
```

If a message has multiple attachments, the appliance does not drop the message if at least one of the attachments is an .exe file, even if the other attachments not `.exe` files.

Table 6: Attachment Groups

Attachment Group Name	Scanned File Types
Document	• doc • docx • mdb • mpp • ole • pdf • ppt • pptx • rtf • wps • x-wmf • xls • xlsx
Executable	• exe • java • msi • pif **Note** Filtering the Executable group will also scan .dll and .scr files, but you cannot filter these file types individually.

Attachment Group Name	Scanned File Types
Compressed	• ace (ACE Archiver compressed file) • arc (SQUASH Compressed archive) • arj (Robert Jung ARJ compressed archive) • binhex • bz (Bzip compressed file) • bz2 (Bzip compressed file) • cab (Microsoft cabinet file) • gzip* (Compressed file - UNIX gzip) • lha (Compressed Archive [LHA/LHARC/LZH]) • rar (Compressed archive • sit (Compressed archive - Macintosh file [Stuffit]) • tar* (Compressed archive) • unix (UNIX compress file) • zip* (Compressed archive - Windows) • zoo (ZOO Compressed Archive File) * These file types can be “body-scanned”
Text	• txt • html • xml
Image	• bmp • cur • gif • ico • jpeg • pcx • png • psd • psp • tga • tiff

Attachment Group Name	Scanned File Types
Media	• aac • aiff • asf • avi • flash • midi • mov • mp3 • mpeg • ogg • ram • snd • wav • wma • wmv

Action Variables

The bcc() , bcc-scan() , notify() , notify-copy() , add-footer() , add-heading() , and insert-headers() actions have parameters that may use certain variables that will be automatically replaced with information from the original message when the action is executed. These special variables are called *action variables.* Your appliance supports the following set of action variables:

Table 7: Message Filter Action Variables

Variable	Syntax	Description
All Headers	`$AllHeaders`	Returns the message headers.
Body Size	`$BodySize`	Returns the size, in bytes, of the message.
Certificate Signers	`$CertificateSigners`	Returns the signers from the subjectAltName element of a signing certificate. See $CertificateSigners Action Variable, on page 45 for more information.
Date	`$Date`	Returns the current date, using the format MM/DD/YYYY.
Dropped File Name	`$dropped_filename`	Returns only the most recently dropped filename.
Dropped File Names	`$dropped_filenames`	Displays list of dropped files (similar to $filenames .
Dropped File Types	`$dropped_filetypes`	Displays list of dropped file types (similar to $filetypes) .

Variable	Syntax	Description
Envelope Sender	`$EnvelopeFrom`	Returns the Envelope Sender (Envelope From, <MAIL FROM>) of the message.
Envelope Recipients	`$EnvelopeRecipients`	Returns all Envelope Recipients (Envelope To, <RCPT TO>) of the message.
File Names	`$filenames`	Returns a comma-separated list of the message's attachments' filenames.
File Sizes	`$filesizes`	Returns a comma-separated list of the message's attachments file sizes.
File Types	`$filetypes`	Returns a comma-separated list of the message's attachments' file types.
Filter Name	`$FilterName`	Returns the name of the filter being processed.
GMTimeStamp	`$GMTimeStamp`	Returns the current time and date, as would be found in the Received: line of an email message, using GMT.
HAT Group Name	`$Group`	Returns the name of the sender group the sender matched on when injecting the message. If the sender group had no name, the string " >Unknown< " is inserted.
Matched Content	`$MatchedContent`	Returns the content that triggered a scanning filter rule (including filter rules such as body-contains and content dictionaries).
Mail Flow Policy	`$Policy`	Returns the name of the HAT policy applied to the sender when injecting the message. If no predefined policy name was used, the string " >Unknown< " is inserted.
Header	`$Header['`*`string`*`']`	Returns the value of the quoted header, if the original message contains a matching header. Note that double quotes may also be used.
Hostname	`$Hostname`	Returns the hostname of the appliance .
Internal Message ID	`$MID`	Returns the Message ID, or "MID" used internally to identify the message. Not to be confused with the RFC822 "Message-Id" value (use $Header to retrieve that).
Receiving Listener	`$RecvListener`	Replaced by the nickname of the listener that received the message.
Receiving Interface	`$RecvInt`	Returns the nickname of the interface that received the message.
Remote IP Address	`$RemoteIP`	Returns the IP address of the system that sent the message to the appliance.

Variable	Syntax	Description
Remote Host Address	`$remotehost`	Returns the hostname of the system that sent the message to the appliance.
SenderBase Reputation Score	`$Reputation`	Returns the SenderBase Reputation score of the sender. If there is no reputation score, it is replaced with “ None ”.
Subject	`$Subject`	Returns the subject of the message.
Time	`$Time`	Returns the current time, in the local time zone.
Timestamp	`$Timestamp`	Returns the current time and date, as would be found in the Received: line of an email message, in the local time zone.

Related Topics

- Non-ASCII Character Sets and Message Filter Action Variables, on page 67

Non-ASCII Character Sets and Message Filter Action Variables

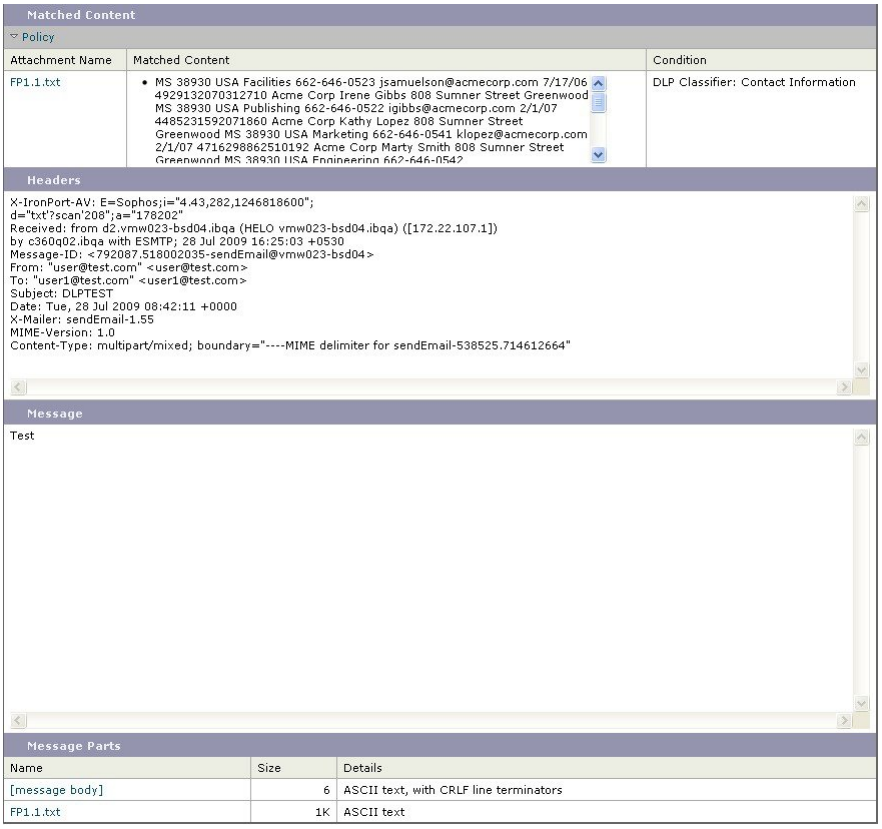
The system supports the expansion of action variables that contain ISO-2022 style character codings (the style of encoding used in header values) and also supports international text in the notification. These will be merged together to generate a notification that will then be sent as a UTF-8, quoted printable message.

Matched Content Visibility

When you configure a quarantine action for messages that match Attachment Content conditions, Message Body or Attachment conditions, Message body conditions, or the Attachment content conditions, you can view the matched content in the quarantined message. When you display the message body, the matched content is highlighted in yellow. You can also use the $MatchedContent action variable to include the matched content in the message subject.

When you view messages in the local quarantine that have triggered message or content filter rules, the GUI may display content that did not actually trigger the filter action (along with content that triggered the filter action). The GUI display should be used as a guideline for locating content matches, but does not necessarily reflect an exact list of content matches. This occurs because the GUI uses less strict content matching logic than is used in the filters. This issue applies only to the highlighting in the message body. The table that lists the matched strings in each part of the message along with the associated filter rule is correct.

Figure 2: Matched Content Viewed in the Policy Quarantine



Description and Examples of Message Filter Actions

The following section describes the various message filter actions in use and their examples.

- Skip Remaining Message Filters Action, on page 69
- Drop Action, on page 69
- Bounce Action, on page 70
- Encrypt Action, on page 70
- Notify and Notify-Copy Actions, on page 71
- Blind Carbon Copy Actions, on page 73
- Quarantine and Duplicate Actions, on page 75
- Alter Recipient Action, on page 76
- Alter Delivery Host Action, on page 76
- Alter Source Host (Virtual Gateway address) Action, on page 77
- Archive Action, on page 77
- Strip Header Action, on page 78
- Insert Header Action , on page 78
- Edit Header Text Action, on page 79
- Edit Body Text Action, on page 79
- HTML Convert Action, on page 80
- Bounce Profile Action, on page 81

- Bypass Anti-Spam System Action, on page 81
- Bypassing Graymail Actions, on page 82
- Bypass Anti-Virus System Action, on page 82
- Bypass File Reputation Filtering and File Analysis System Actions, on page 83
- Bypass Anti-Virus System Action, on page 82
- Add Message Tag Action, on page 83
- Add Log Entry Action, on page 84
- URL Reputation Actions , on page 84
- URL Category Actions , on page 86
- No Operation, on page 87
- Forged Email Detection Action, on page 87

Skip Remaining Message Filters Action

The `skip-filters` action ensures that the message skips any further processing from message filters and continues through the email pipeline. The message that incurs the `skip-filters` action will be subject to anti-spam scanning and anti-virus scanning, if it is available on the appliance . The `skip-filters` action is the default final action for message filters.

The following filter notifies `customercare@example.com` and then immediately delivers any message addressed to `boss@admin` .

```
bossFilter:

if(rcpt-to == 'boss@admin$')

{

notify('customercare@example.com');

skip-filters();

}
```

Drop Action

The `drop` action discards a message without any delivery. The message is not returned to the sender, not sent to the intended recipient, nor processed further in any way.

The following filter first notifies `george@whitehouse.gov` and then discards any message where the subject begins with `SPAM` .

```
spamFilter:

if(subject == '^SPAM.*')

{

notify('george@whitehouse.gov');

drop();

}
```

Bounce Action

The `bounce` action sends the message back to the sender (Envelope Sender) without further processing.

The following filter returns (bounces) any message from an email address that ends in `@yahoo\\.com` .

```
yahooFilter:

if(mail-from == '@yahoo\\.com$')

{

bounce();

}
```

Encrypt Action

The `encrypt` action uses the configured encryption profile to deliver encrypted messages to email recipients.

The following filter encrypts messages if they contain the term `[encrypt]` in the subject:

```
Encrypt_Filter:

if ( subject == '\\[encrypt\\]' )

{

encrypt('My_Encryption_Profile');

}
```

Note You must have a Cisco Encryption Appliance in your network or a hosted key service configured to use this filter action. You must also have configured an encryption profile to use this filter action.

S/MIME Sign or Encrypt on Delivery Action

The smime-gateway-deferred action performs an S/MIME signing or encryption of the message using the specified sending profile during the delivery. This means that the message continues to the next stage of processing, and when all processing is complete, the message is signed or encrypted and delivered.

The following filter performs an S/MIME encryption on all the outgoing messages from a particular sender during the delivery:

```
smime-deferred:if(mail-from == "user@example.com"){smime-gateway-deferred("smime-encrypt");}
```

S/MIME Sign or Encrypt Action

The smime-gateway action performs an S/MIME signing or encryption using the specified sending profile and delivers the message, skipping any further processing.

The following filter performs an S/MIME signing on all the outgoing messages from a particular sender and delivers them immediately:

```
smime-deliver-now:if(mail-from == "user@example.com"){smime-gateway("smime-sign");}
```

Notify and Notify-Copy Actions

The `notify` and `notify-copy` actions send an email summary of the message to the specified email address. The `notify-copy` action also sends a copy of the original message, similar to the bcc-scan action. The notification summary contains:

- The contents of the Envelope Sender and Envelope Recipient (`MAIL FROM` and `RCPT TO`) directives from the mail transfer protocol conversation for the message.
- The message headers of the message.
- The name of the message filter that matched the message.

You can specify the recipient, subject line, from address, and notification template. the following filter selects messages with sizes larger than 4 megabytes, sends a notification email of each matching message to `admin@example.com` , and finally discards the message:

```
bigFilter:

if(body-size >= 4M)

{

notify('admin@example.com');

drop();

}
```

Or

```
bigFilterCopy:

if(body-size >= 4M)

{

notify-copy('admin@example.com');

drop();

}
```

The Envelope Recipient parameter may be any valid email address (for example, `admin@example.com` in the example above), or alternatively, may be the action variable `$EnvelopeRecipients` (see Action Variables, on page 65), which specifies all Envelope Recipients of the message:

```
bigFilter:

if(body-size >= 4M)

{

notify('$EnvelopeRecipients');

drop();
```

```
}
```

The `notify` action also supports up to three additional, optional arguments that allow you to specify the subject header, the Envelope Sender, and a pre-defined text resource to use for the notification message. These parameters must appear in order, so a subject must be provided if the Envelope Sender is to be set or a notification template specified.

The subject parameter may contain action variables (see Action Variables, on page 65) that will be replaced with data from the original message. By default, the subject is set to `Message Notification .`

The Envelope Sender parameter may be any valid email address, or alternatively, may be the action variable `$EnvelopeFrom` , which will set the return path of the message to the same as the original message

The notification template parameter is the name of an existing notification template. For more information, see Notifications, on page 94.

This example extends the previous one, but changes the subject to look like `[bigFilter] Message too large` , sets the return path to be the original sender, and uses the “message.too.large” template:

```
bigFilter:

if (body-size >= 4M)

{

notify('admin@example.com', '[$FilterName] Message too large',

'$EnvelopeFrom', 'message.too.large');

drop();

}
```

You can also use the `$MatchedContent` action variable to notify senders or administrators that a content filter was triggered. The `$MatchedContent` action variable displays the content that triggered the filter. For example, the following filter sends a notification to an administrator if the email contains ABA account information.

```
ABA_filter:

if (body-contains ('*aba')){

notify('admin@example.com','[$MatchedContent]Account Information Displayed');

}
```

Related Topics

- Notification Template, on page 72

Notification Template

You can use the Text Resources page or the `textconfig` CLI command to configure custom notification templates as text resources for use with the `notify()` and `notify-copy()` actions. If you do not create a custom notification template, a default template is used. The default template includes message headers, but the custom notification template does not include message headers by default. To include message headers in the custom notification, include the `$AllHeaders` action variable.

For more information, see the “Text Resources” chapter.

In the following example, when a large message triggers the filter shown below, an email is sent to the intended recipients explaining that the message was too large:

```
bigFilter:

if (body-size >= 4M)

{

notify('$EnvelopeRecipients', '[$FilterName] Message too large',

'$EnvelopeFrom', 'message.too.large');

drop();

}
```

Blind Carbon Copy Actions

The `bcc` action sends an anonymous copy of the message to a specified recipient. This is sometimes referred to as message replication. Because no mention of the copy is made in the original message and the anonymous copy will never successfully bounce back to the recipient, the original sender and recipients of the message will not necessarily know that the copy was sent.

The following filter sends a blind carbon copy to `mom@home.org` for each message addressed to sue from `johnny`:

```
momFilter:

if ((mail-from == '^johnny$') and (rcpt-to == '^sue$'))

{

bcc('mom@home.org');

}
```

The bcc action also supports up to three additional, optional arguments that allow you to specify the subject header and Envelope Sender to use on the copied message, as well as an alt-mailhost. These parameters must appear in order, so a subject must be provided if the Envelope Sender is to be set.

The subject parameter may contain action variables (see Action Variables, on page 65) that will be replaced with data from the original message. By default, this is set to the subject of the original message (the equivalent of `$Subject`).

The Envelope Sender parameter may be any valid email address, or alternatively, may be the action variable `$EnvelopeFrom`, which will set the return path of the message to the same as the original message.

This example expands the previous one by setting the subject to be `[Bcc] <original subject>`, and the return path set to `badbounce@home.org`:

```
momFilter:

if ((mail-from == '^johnny$') and (rcpt-to == '^sue$'))
```

```
{
bcc('mom@home.org', '[Bcc] $Subject', 'badbounce@home.org');
}
```

The alt-mailhost is the fourth parameter:

```
momFilterAltM:
if ((mail-from == '^johnny$') and (rcpt-to == '^sue$'))
{
bcc('mom@home.org', '[Bcc] $Subject', '$EnvelopeFrom',
'momaltmailserver.example.com');
}
```

Caution The `Bcc()`, `notify()`, and `bounce()` filter actions can allow viruses through your network. The blind carbon copy filter action creates a new message which is a full copy of the original message. The notify filter action creates a new message that contains the headers of the original message. While it is rare, headers can contain viruses. The bounce filter action creates a new message which contains the first 10k of the original message. In all three cases, the new message will not be processed by anti-virus or anti-spam scanning.

To send to multiple hosts, you can call the `bcc()` action multiple times:

```
multiplealthosts:
if (recv-listener == "IncomingMail")
{
insert-header('X-ORIGINAL-IP', '$remote_ip');
bcc ('$EnvelopeRecipients', '$Subject', '$EnvelopeFrom', '10.2.3.4');
bcc ('$EnvelopeRecipients', '$Subject', '$EnvelopeFrom', '10.2.3.5');
bcc ('$EnvelopeRecipients', '$Subject', '$EnvelopeFrom', '10.2.3.6');
}
```

Related Topics

- BCC and Scan Mail Sent to Competitors, on page 114

The bcc-scan() Action

The `bcc-scan` action functions similarly to the bcc action, except that the message that is sent is treated as a brand new message and is therefore sent through the entire email pipeline.

```
momFilter:
```

```
if ((mail-from == '^johnny$') and (rcpt-to == '^sue$'))

{

bcc-scan('mom@home.org');

}
```

Quarantine and Duplicate Actions

The `quarantine('quarantine_name')` action flags a message for inclusion into a queue called a quarantine. For more information about quarantines, see the "Quarantines" chapter. The `duplicate-quarantine ( 'quarantine_name')` action immediately places a copy of the message into the specified quarantine and the original message continues through the email pipeline. The quarantine name is case sensitive.

When flagged for quarantine, the message continues through the rest of the email pipeline. When the message reaches the end of the pipeline, if the message has been flagged for one or more quarantines then it enters those queues. Otherwise, it is delivered. Note that if the message does not reach the end of the pipeline, it is not placed in a quarantine.

Accordingly, if a message filter contains a `quarantine()` action followed by a `bounce()` or `drop()` action, the message will not enter the quarantine, since the final action prevents the message from reaching the end of the pipeline. The same is true if a message filter includes a quarantine action, but the message is later dropped by anti-spam or anti-virus scanning, or a content filter. The `skip-filters()` action causes the message to skip any remaining message filters, but content filters may still apply. For example, if a message filter flags a message for quarantine and also includes the `skip-filters()` action, the message skips all remaining message filters and will be quarantined, unless another action in the email pipeline causes the message to be dropped.

In the following example, the message is sent to the Policy quarantine if the message contains any words within the dictionary named "secret_word."

```
quarantine_codenames:

if (dictionary-match ('secret_words'))

{

quarantine('Policy');

}
```

In the following example, suppose a company has an official policy to drop all .mp3 file attachments. If an inbound message has a .mp3 attachment, the attachment is stripped and the remaining message (original body and remaining attachments) is sent to the original recipient. Another copy of the original message with all attachments will be quarantined (sent to the Policy quarantine). If it is necessary to receive the blocked attachment(s), the original recipient would then request that the message be released from the quarantine.

```
strip_all_mp3s:

if (attachment-filename == '(?i)\\.mp3$') {

duplicate-quarantine('Policy');

drop-attachments-by-name('(?i)\\.mp3$');
```

```
}
```

Alter Recipient Action

The `alt-rcpt-to` action changes all recipients of the message to the specified recipient upon delivery.

The following filter sends all messages with an Envelope Recipient address that contain `.freelist.com` and changes all recipients for the message to `system-lists@myhost.com` :

```
freelistFilter:

if(rcpt-to == '\\.freelist\\.com$')

{

alt-rcpt-to('system-lists@myhost.com');

}
```

Alter Delivery Host Action

The `alt-mailhost` action changes the IP address for all recipients of the selected message to the numeric IP address or hostname given.

Note The `alt-mailhost` action prevents a message classified as spam by an anti-spam scanning engine from being quarantined. The `alt-mailhost` action overrides the `quarantine` action and sends it to the specified mail host.

The following filter redirects recipient addresses to the host example.com for all messages.

```
localRedirectFilter:

if(true)

{

alt-mailhost('example.com');

}
```

Thus, a message directed to `joe@anywhere.com` is delivered to the mailhost at example.com with the Envelope To address `joe@anywhere.com` . Note that any additional routing information specified by the `smtproutes` command still affects the routing of the message. (See Routing Email for Local Domains.)

Note The `alt-mailhost` action does not support specifying a port number. To do this, add an SMTP route instead.

The following filter redirects all messages to `192.168.12.5` :

```
local2Filter:
```

```
if(true)

{

alt-mailhost('192.168.12.5');

}
```

Alter Source Host (Virtual Gateway address) Action

The `alt-src-host` action changes the source host for the message to the source specified. The source host consists of the IP interface or group of IP interfaces that the messages should be delivered from. If a group of IP interfaces is selected, the system round-robins through all of the IP interfaces within the group as the source interface when delivering email. In essence, this allows multiple Virtual Gateway addresses to be created on a single appliance . For more information, see Configuring Mail Gateways for all Hosted Domains Using Virtual Gateway™ Technology.

The IP interface may only be changed to an IP interface or interface group currently configured in the system. the following filter creates a Virtual Gateway using the outbound (delivery) IP interface `outbound2` for all messages received from a remote host with the IP address `1.2.3.4` .

```
externalFilter:

if(remote-ip == '1.2.3.4')

{

alt-src-host('outbound2');

}
```

The following filter uses the IP interface group Group1 for all messages received from a remote host with the IP address `1.2.3.4` .

```
groupFilter:

if(remote-ip == '1.2.3.4')

{

alt-src-host('Group1');

}
```

Archive Action

The `archive` action saves a copy of the original message, including all message headers and recipients into an mbox-format file on the appliance. The action takes a parameter that is the name of the log file in which to save the message. The system automatically creates a log subscription with the specified filename when you create the filter, or you can also specify an existing filter log file. After the filter and the filter log file are created, the filter log options may then be edited with the `filters -> logconfig` subcommand.

Note The `logconfig` command is a subcommand of `filters` . See Using the CLI to Manage Message Filters, on page 98 for a full description of how to use this subcommand.

The mbox format is a standard UNIX mailbox format, and there are many utilities available to make viewing the messages easier. Most UNIX systems allow you to type " `mail -f` *mbox.filename* " to view the files. The mbox format is in plain text, so you can use a simple text editor to view the contents of the messages.

In the following example, a copy of the message is saved to a log named `joesmith` if the Envelope Sender matches `joesmith@yourdomain.com` :

```
logJoeSmithFilter:

if(mail-from == '^joesmith@yourdomain\\.com$')

{

archive('joesmith');

}
```

Strip Header Action

The `strip-header` action examines the message for a particular header and removes those lines from the message before delivering it. When there are multiple headers, all instances of the header are removed (for example, the "Received:" header.)

In the following example, all messages have the header `X-DeleteMe` removed before transmission:

```
stripXDeleteMeFilter:

if (true)

{

strip-header('X-DeleteMe');

}
```

When working with headers, remember that the current value of the header includes changes made during processing (such as with filter actions that add, remove, or modify message headings). See Message Header Rules and Evaluation, on page 5 for more information.

Insert Header Action

The `insert-header` action inserts a new header into a message. AsyncOS does not verify the compliance to standards of the header you insert; you are responsible for ensuring that the resulting message complies with Internet standards for email.

The following example inserts a header named `X-Company` with the value set to `My Company Name` if the header is not already found in the message:

```
addXCompanyFilter:
```

```
if (not header('X-Company'))

{

insert-header('X-Company', 'My Company Name');

}
```

The `insert-header()` action allows the use of non-ASCII characters in the text of the header, while restricting the header name to be ASCII (to comply with standards). The transport encoding will be quoted-printable to maximize the readability.

Note The `strip-headers` and `insert-header` actions can be used in combination to rewrite any message headers in the original message. In some case, it is valid to have multiple instances of the same header (for example, `Received:`) where in other cases, multiple instances of the same header could confuse a MUA (for example, multiple Subject: headers.)

When working with headers, remember that the current value of the header includes changes made during processing (such as with filter actions that add, remove, or modify message headings). See Message Header Rules and Evaluation, on page 5 for more information.

Edit Header Text Action

The `edit-header-text` action allows you to rewrite specified header text using the regular expression substitution function. The filter matches the regular expression within the header and replaces it with a regular expression you specify.

For example, an email contains the following subject header:

```
Subject: SCAN Marketing Messages
```

The following filter removes the “SCAN” text, and leaves the text, “Marketing Messages”, in the header:

```
Remove_SCAN: if true

{

edit-header-text (‘Subject’, ‘^SCAN\\s*’,’’);

}
```

After the filter processes the message, it returns the following header:

```
Subject: Marketing Messages
```

Edit Body Text Action

The `edit-body-text()` message filter is similar to the `Edit-Header-Text()` filter, but it operates across the body of the message instead of one of the headers.

The `edit-body-text()` message filter uses the following syntax where the first parameter is the regular expression to search for and the second parameter is the replacement text:

```
Example: if true {
```

```
edit-body-text("parameter 1","parameter 2");

}
```

The `edit-body-text()` message filter only works on the message body parts. For more information about whether a given MIME part is considered a message “body” or a message “attachment”, see Message Bodies vs. Message Attachments, on page 5.

The following example shows a URL removed from a message and replaced with the text, ‘URL REMOVED’:

```
URL_Replaced: if true {

edit-body-text("(?i)(?:https?|ftp)://[^\\s\">]+", "URL REMOVED");

}
```

The following example shows a social security number removed from the body of a message and replaced with the text, “XXX-XX-XXXX’:

```
ssn: if true {

edit-body-text("(?!000)(?:[0-6]\\d{2}|7(?:[0-6]\\d|7[012]))([
-]?)(?!00)\\d\\d\\1(?!0000)\\d{4}",

"XXX-XX-XXXX");

}
```

Note You cannot use smart identifiers with the `edit-body-text()` filter at this time.

HTML Convert Action

While RFC 2822 defines a text format for email messages, there are extensions (such as MIME) to provide the transport of other content within an RFC 2822 message. AsyncOS can now use the `html-convert()` message filter to convert HTML to plain text using the following syntax:

```
Convert_HTML_Filter:

if (true)

{

html-convert();

}
```

The Cisco message filters make a determination on whether a given MIME part is considered a message “body” or a message “attachment”. The `html-convert()` filter only works on the message body parts. For more information about message bodies and attachments, see Message Bodies vs. Message Attachments, on page 5.

Depending on the format, the `html-convert()` filter uses different methods to strip the HTML from within the documents.

If the message is plain text (text/plain), the message passes through the filter unchanged. If the message is a simple HTML message (text/html), all the HTML tags are stripped out of the message and the resulting body replaces the HTML message. The lines are not reformatted, and the HTML is not rendered in plain text. If the structure is MIME (with a multipart/alternative structure) and it contains both a text/plain part and text/html part with the same content, the filter removes the text/html part of the message and leaves the text/plain part of the message. For all other MIME types (such as multipart/mixed), all HTML body parts are stripped of their tags and reinserted into the message.

When encountered in a message filter, the `html-convert()` filter action only tags the message to be processed but does not immediately make a change to the message structure. The changes to the message only take effect after all processing is complete. This allows the other filter actions to process the original message body prior to modification.

Bounce Profile Action

The `bounce-profile` action assigns a previously-configured bounce profile to the message. (See Directing Bounced Email.) If the message is undeliverable, the bounce options configured via the bounce profile are used. Using this feature overrides the bounce profile assigned to the message from the listener's configuration (if one is assigned).

The following filter example assigns the bounce profile "fastbounce" to all email sent with the header `X-Bounce-Profile: fastbounce`:

```
fastbounce:

if (header ('X-Bounce-Profile') == 'fastbounce') {

bounce-profile ('fastbounce');

}
```

Bypass Anti-Spam System Action

The `skip-spamcheck` action instructs the system to allow the message to bypass any content-based anti-spam filtering configured on the system. This action does nothing to the message if no content-based anti-spam filtering is configured, or if the message was never flagged to be scanned for spam in the first place.

The following example allows messages that have a high SenderBase Reputation Score to bypass the content-based anti-spam filtering feature:

```
allowed_list_on_reputation:

if (reputation > 7.5)

{

skip-spamcheck();

}
```

Related Topics

- How Incoming Relays Affect Functionality
- Protecting Appliance -Generated Messages From the Spam Filter

Bypassing Graymail Actions

If you do not want to apply graymail actions on certain messages, you can bypass them using the following message filter actions:

Message Filter Action	Description
`skip-marketingcheck`	Bypass actions on marketing emails
`skip-socialcheck`	Bypass actions on social network emails
`skip-bulkcheck`	Bypass actions on bulk emails

The following example specifies that messages received on the listener "private_listener" must bypass graymail actions on social network emails.

```
internal_mail_is_safe:

if (recv-listener == 'private_listener')

{

skip-socialcheck();

}
```

Bypass Anti-Virus System Action

The `skip-viruscheck` action instructs the system to allow the message to bypass any virus protection system configured on the system. This action does nothing to the message if there is no anti-virus system configured, or if the message was never flagged to be scanned for viruses in the first place.

The following example specifies that messages received on the listener "private_listener" should bypass the anti-spam and the anti-virus systems.

```
internal_mail_is_safe:

if (recv-listener == 'private_listener')

{

skip-spamcheck();

skip-viruscheck();

}
```

Bypass File Reputation Filtering and File Analysis System Actions

The skip-ampcheck action instructs the system to allow message to bypass File Reputation Filtering and File Analysis configured on the system. This action does nothing to the message if File Reputation Filtering and File Analysis is not configured, or if the message was never flagged to be scanned for File Reputation Filtering and File Analysis in the first place.

The following example specifies that messages with PDF attachments should bypass File Reputation Filtering and File Analysis.

```
skip_amp_scan:
if (attachment-filetype == 'pdf')
{
skip-ampcheck();
}
```

Bypass Outbreak Filter Scanning Action

The `skip-vofcheck` action instructs the system to allow the message to bypass the Outbreak Filters scanning. This action does nothing to the message if Outbreak Filters scanning is not enabled.

The following example specifies that messages received on the listener "private_listener" should bypass Outbreak Filter scanning.

```
internal_mail_is_safe:

if (recv-listener == 'private_listener') Outbreak Filters

{

skip-vofcheck();

}
```

Add Message Tag Action

The `tag-message` action inserts a custom term into an outgoing message to use with DLP policy filtering. You can configure a DLP policy to limit scanning to messages with the message tag. The message tag is not visible to recipients. The tag name can contain any combination of characters from the set `[a-zA-Z0-9_-.]` .

For information on configuring a DLP policy to filter messages, see the "Data Loss Prevention" chapter.

The following example inserts a message tag into a message with "`[Encrypt]`" in the subject. You can then create a DLP policy that will encrypt messages with this message tag before delivering them if Cisco Email Encryption is available:

```
Tag_Message:

if (subject == '^\\[Encrypt\\]')

{

tag-message('Encrypt-And-Deliver');

}
```

Add Log Entry Action

The `log-entry` action inserts customized text into the Text Mail logs at the `INFO` level. The text can include action variables. You can use this action to insert useful text for debugging purposes and information on why a message filter performed a certain action. The log entry also appears in message tracking.

The following example inserts a log entry explaining that message was bounced because it possibly contained confidential company information:

```
CompanyConfidential:

if (body-contains('Company Confidential'))

{

log-entry('Message may have contained confidential information.');

bounce();

}
```

URL Reputation Actions

Use the reputation score of URLs in messages to modify the URLs or their behavior. For important details and examples, see Modifying URLs in Messages: Using URL Reputation and URL Category Actions in Filters in Protecting Against Malicious or Undesirable URLs

No rule is needed with these actions.

In URL Reputation actions:

- `msg_filter_name` : is the name of this message filter.
- `min_score` and `max_score` are the minimum and maximum scores in the range for which the action should apply. The applicable range includes the values that you specify.

Minimum and maximum scores must be between `-10.0` and `10.0` .

- To specify an action when the reputation service does not provide a score, use the corresponding "no-reputation" version of the action, as shown in the following subsections.
- `allowedlist` is the name of a defined URL list (via the `urllistconfig` command.) Specifying an allowed list is optional.
- In place of `Preserve_signed` , enter 0 or 1:
  - 1 - Apply this action to unsigned messages only
  - 0 - Apply this action to all messages

If you do not specify a `preserve_signed` value, the action is applied to unsigned messages only.

Related Topics

- Replace URL with Text, Based on URL Reputation, on page 85
- Defang URL, Based on URL Reputation, on page 85
- Redirect URL to Cisco Security Proxy, Based on URL Reputation , on page 85

Replace URL with Text, Based on URL Reputation

To take action when the reputation service provides a score:

Use the `url-reputation-replace` action.

The syntax of a filter using the `url-reputation-replace` action is:

```
<msg_filter_name>:
if <condition>
{url-reputation-replace(<min_score>, <max_score>,'<replace_text>', '< allowedlist>', < Preserve_signed> );}
```

Where `replace_text` is the text with which to replace the URL.

To take action when the reputation service does not provide a score:

Use the `url-no-reputation-replace` action.

The syntax of a filter using the url-no-reputation-replace action is:

```
<msg_filter_name>:
if <condition>
{url-no-reputation-replace ('<replace_text>', '<allowedlist>', <Preserve_signed>);}
```

Where `replace_text` is the text with which to replace the URL.

Defang URL, Based on URL Reputation

To take action when the reputation service provides a score:

Use the `url-reputation-defang` action.

The syntax of a filter using the `url-reputation-defang` action is:

```
<msg_filter_name>:
if <condition>
{url-reputation-defang (<min_score>, <max_score>, '<allowedlist>', <Preserve_signed>);}
```

To take action when the reputation service does not provide a score:

Use the `url-no-reputation-defang` action.

The syntax of a filter using the `url-no-reputation-defang` action is:

```
<msg_filter_name>:
if <condition>
{url-no-reputation-defang ('<allowedlist>', <Preserve_signed>);}
```

Redirect URL to Cisco Security Proxy, Based on URL Reputation

To take action when the reputation service provides a score:

Use the `url-reputation-proxy-redirect` action.

The syntax of a filter using the `url-reputation-proxy-redirect` action is:

```
<msg_filter_name>:

if <condition>

{url-reputation-proxy-redirect (<min_score>, <max_score>, '<allowedlist>',
<Preserve_signed>);}
```

To take action when the reputation service does not provide a score:

Use the `url-no-reputation-proxy-redirect` action.

The syntax of a filter using the `url-no-reputation-proxy-redirect` action is:

```
<msg_filter_name>:

if <condition>

{url-no-reputation-proxy-redirect ('<allowedlist>', <Preserve_signed>);}
```

URL Category Actions

Use the categories of URLs in messages to modify the URLs or their behavior. For important details, see Modifying URLs in Messages: Using URL Reputation and URL Category Actions in Filters in Protecting Against Malicious or Undesirable URLs

No rule is needed with these actions.

In all URL Category actions:

- msg_filter_name : is the name of the message filter.
- category-name is the URL category. Separate multiple categories with commas. To obtain correct category names, look at a URL Category condition or action in a Content Filter. For descriptions and examples of the categories, see About URL Categories.
- url_allowed_list is the name of a defined URL list (via the urllistconfig command.)
- unsigned-only : Enter 0 or 1.
    - 1 - Apply this action to unsigned messages only
    - 0 - Apply this action to all messages

Related Topics

- Replace URL with Text, Based on URL Category , on page 86
- Defang URL, Based on URL Category , on page 87
- Redirect URL to Cisco Security Proxy, Based on URL Category , on page 87

Replace URL with Text, Based on URL Category

The syntax of a filter using the `url-category-replace` action is

```
<msg_filter_name>:

if <condition>

url-category-replace(['<category-name1>','<category-name2>',...,
'<category-name3>'],'<replacement-text>', '<url_allowed_list>', <unsigned-only>);
```

Where `replacement-text` is the text that you want to use to replace the URL.

Defang URL, Based on URL Category

The syntax of a filter using the `url-category-defang` action is:

```
<msg_filter_name>:

if <condition>

url-category-defang(['<category-name1>','<category-name2>',..., '<category-name3>'],
'<url_allowed_list>', <unsigned-only>);
```

Redirect URL to Cisco Security Proxy, Based on URL Category

The syntax of a filter using the `url-category-proxy-redirect` action is:

```
<msg_filter_name>:

if <condition>

url-category-proxy-redirect(['<category-name1>','<category-name2>',..., '<category-name3>'],
'<url_allowed_list>', <unsigned-only>);
```

No Operation

The No Operation action performs a no-op, or no operation. You can use this action in a message filter if you do not want to use any of the other actions such as Notify, Quarantine, or Drop. For example, to understand the behavior of a new message filter that you created, you can use the No Operation action. After the message filter is operational, you can monitor the behavior of the new message filter using the Message Filters report page, and fine-tune the filter to match your requirements.

The following example shows how to use No Operation action in a message filter.

```
new_filter_test: if header-repeats ('subject', X, 'incoming') {no-op();}
```

Forged Email Detection Action

Strips the From: header from the forged message and replaces it with the Envelope Sender.

The following message filter compares the From: header in the message with the terms in dictionary and if the matching score of a term in the content dictionary is greater than or equal to 70, the message filter strips the From: header and replaces it with the Envelope Sender.

```
FED_CF: if (forged-email-detection("Execs", 70)) { fed("from", ""); }
```

Attachment Scanning

The appliance uses Content Scanner to strip attachments from messages that are inconsistent with your corporate policies, while still retaining the ability to deliver the original message.

You can filter attachments based on their specific file type, fingerprint, or based on the content of the attachment. Using the fingerprint to determine the exact type of attachment prevents users from renaming a malicious attachment extension (for example, .exe) to a more commonly used extension (for example, .doc) in the hope that the renamed file would bypass attachment filters.

When you scan attachments for content, the Content Scanner extracts data from attachment files to search for the regular expression. It examines both data and metadata in the attachment file. If you scan an Excel or

Word document, the attachment scanning engine can also detect the following types of embedded files: .exe, .dll, .bmp, .tiff, .pcx, .gif, .jpeg, .png, and Photoshop images.

The Content Scanner in your appliance can perform content scanning on the following archive file formats:

- ACE Archive
- ALZ Archive
- Apple Disk Image
- ARJ Archive
- bzip2 Archive
- EGG Archive
- GNU Zip
- ISO Disk Image
- Java Archive
- LZH
- Microsoft Cabinet Archive
- RAR Multi-Part File
- RedHat Package Manager Archive
- Roshal Archive (RAR)
- Unix AR Archive
- UNIX Compress Archive
- UNIX cpio
- UNIX Tar
- XZ Archive
- Zip Archive
- 7-Zip
- ARC

Note You can view the details of the Content Scanner-related files using the **Security Services** > **Scan Behavior** page in web interface or using the contentscannerstatus command in CLI. These files are automatically updated using update server. If you want to manually update these files, see Configuring Scan Behavior, on page 120.

Related Topics

- Message Filters for Scanning Attachments, on page 89
- Image Analysis, on page 90

• Configuring the Image Analysis Scanning Engine, on page 90
• Configuring the Message Filter to Perform Actions Based on Image Analysis Results, on page 92
• Notifications, on page 94
• Examples of Attachment Scanning Message Filters, on page 94

Message Filters for Scanning Attachments

The message filter actions described in the following table are *non-final* actions. (Attachments are dropped and the message processing continues.)

The optional comment is text that is added to the message, much like a footer, and it can contain Message Filter Action Variables (see Examples of Attachment Scanning Message Filters, on page 94).

Table 8: Message Filter Actions for Attachment Filtering

Action	Syntax	Description
Drop Attachments by Name	`drop-attachments-by-name (<regular expression >[, <optional comment >])`	Drops all attachments on messages that have a filename that matches the given regular expression. Archive file attachments (zip, tar) will be dropped if they contain a file that matches. See Examples of Attachment Scanning Message Filters, on page 94.
Drop Attachments by Type	`drop-attachments-by-type (<MIME type >[, <optional comment >])`	Drops all attachments on messages that have a MIME type, determined by either the given MIME type or the file extension. Archive file attachments (zip, tar) will be dropped if they contain a file that matches.
Drop Attachments by File Type	`drop-attachments-by-filetype (<fingerprint name >[, <optional comment >])`	Drops all attachments on messages that match the given “fingerprint” of the file. Archive file attachments (zip, tar) will be dropped if they contain a file that matches.
Drop Attachments by MIME Type	`drop-attachments-by-mimetype (<MIME type >[, <optional comment >])`	Drops all attachments on messages that have a given MIME type. This action does not attempt to ascertain the MIME type by file extension and so it also does not examine the contents of archives.
Drop Attachments by Size	`drop-attachments-by-size (<number >[, <optional comment >])`	Drops all attachments on the message that, in raw encoded form, are equal to or greater than the size (in bytes) given. Note that for archive or compressed files, this action does not examine the uncompressed size, but rather the size of the actual attachment itself.

Action	Syntax	Description
Attachment Scanning	`drop-attachments-where-contains ("<regular expression>"[, "<optional comment>"])`	Drops all attachments on message that contain the regular expression. Archive files (zip, tar) will be dropped if any of the files they contain match the regular expression pattern.
Drop Attachments by Dictionary Matches	`drop-attachments-where-dictionary-match("<dictionary name>")`	This filter action strips attachments based on matches to dictionary terms. If the terms in the MIME parts considered to be an attachment match a dictionary term (and the user-defined threshold is met), the attachment is stripped from the email. See Examples of Attachment Scanning Message Filters, on page 94.

Image Analysis

Some messages contain images that you may wish to scan for inappropriate content. Use the image analysis engine to search for inappropriate content in email.

The image analyzer uses algorithms that measure image attributes to determine the likelihood of inappropriate content. These algorithms can detect, for example, the shapes and color palette in an image. The analyzer can identify the type of shapes in an image and the percentage of any flesh-tone colors relative to the other colors in the image to help identify inappropriate content. Images with a high percentage of flesh-tone colors are more likely to be inappropriate. The algorithms do not discriminate in any way.

Image analysis is not designed to supplement or replace your Anti-Virus and Anti-Spam scanning engines. Its purpose is to enforce acceptable use by identifying inappropriate content in email. Use the image analysis scanning engine to quarantine and analyze mail and to detect trends.

After you configure your appliance for image analysis, you can use image analysis filter rules to perform actions on suspect or inappropriate emails. Image scanning allows you to scan the following types of attached files: BMP, JPG, TIF, PNG, GIF, TGA, and PCX.

When you scan image attachments, Cisco fingerprinting determines the file type, and the image analyzer uses algorithms to analyze the image content. If the image is embedded in another file, the Content Scanner extracts the file. The image analysis verdict is computed on the message as a whole. If the message does not include any images, the message receives a score of "0" which maps to a "clean" verdict. Therefore, a message without any images will receive a "clean" verdict.

Configuring the Image Analysis Scanning Engine

To enable image analysis from the GUI:

Step 1 Go to **Security Services > IronPort Image Analysis**.

Step 2 Click **Enable**.

A success message displays, and the verdict settings display.

The image analysis filter rule allows you to determine the actions to take based on the following verdicts:

- **Clean:** The image is free of inappropriate content. The image analysis verdict is computed on the message as a whole, so a message without any images will receive a "clean" verdict if scanned.
- **Suspect**: The image may contain inappropriate content.
- **Inappropriate**: The image contains inappropriate content.

These verdicts represent a numeric value assigned by the image analyzer algorithm to determine probability of inappropriate content.

The following values are recommended:

- Clean: 0 to 49
- Suspect: 50 to 74
- Inappropriate: 75 to 100

What to do next

You can fine-tune image scanning by configuring the sensitivity setting, which helps reduce the number of false positives. For example, if you find that you are getting false positives, you can decrease the sensitivity setting. Or, conversely, if you find that the image scanning is missing inappropriate content, you may want to set the sensitivity higher. The sensitivity setting is a value between 0 (no sensitivity) and 100 (highly sensitive). The default sensitivity setting of 65 is recommended.

Related Topics

- Tuning Image Analysis Settings, on page 91

Tuning Image Analysis Settings

Step 1 Go to **Security Services > IronPort Image Analysis**.

Step 2 Click **Edit Settings**.

Step 3 Configure the settings for image analysis sensitivity. The default sensitivity setting of 65 is recommended.

Step 4 Configure the settings for Clean, Suspect, and Inappropriate verdicts.

When you configure the value ranges, ensure that you do not overlap values and that you use whole integers.

Step 5 Optionally, configure AsyncOS to bypass scanning images that do not meet a minimum size requirement (recommended). By default, this setting is configured for 100 pixels. Scanning images that are smaller than 100 pixels can sometimes result in false positives.

You can also enable image analysis settings from the CLI using the imageanalysisconfig command:

What to do next

Related Topics

- Viewing the Verdict Score of a Particular Message, on page 92

Viewing the Verdict Score of a Particular Message

To see the verdict score for a particular message, you can view the mail logs. The mail logs display the image name or file name, the score for a particular message attachment. In addition, the log displays information about whether the images in a file were scannable or unscannable. Note that information in the log describes the result for each message attachment, rather than each image. For example, if the message had a zip attachment that contained a JPEG image, the log entry would contain the name of the zip file rather than the name of the JPEG. Also, if the zip file included multiple images then the log entry would include the maximum score of all the images. The unscannable notation indicates whether any of the images were unscannable.

The log does not contain information about how the scores translate to a particular verdict (clean, suspect or inappropriate). However, because you can use mail logs to track the delivery of specific messages, you can determine by the actions performed on the messages whether the mail contained inappropriate or suspect images.

For example, the following mail log shows attachments dropped by message filter rules as a result of Image Analysis scanning:

```
Thu Apr 3 08:17:56 2009 Debug: MID 154 IronPort Image Analysis: image 'Unscannable.jpg'
is unscannable.

Thu Apr 3 08:17:56 2009 Info: MID 154 IronPort Image Analysis: attachment
'Unscannable.jpg' score 0 unscannable

Thu Apr 3 08:17:56 2009 Info: MID 6 rewritten to MID 7 by
drop-attachments-where-image-verdict filter 'f-001'

Thu Apr 3 08:17:56 2009 Info: Message finished MID 6 done
```

Configuring the Message Filter to Perform Actions Based on Image Analysis Results

Once you enable image analysis, you must create a message filter to perform different actions for different message verdicts. For example, you may wish to deliver messages with a clean verdict, but quarantine messages that are determined to have inappropriate content.

Note Cisco recommends you do not drop or bounce messages with inappropriate or suspect verdicts. Instead, send copies of violations to a quarantine for later review and better understanding of trend analysis.

The following filter shows messages tagged if the content is inappropriate or suspect:

```
image_analysis: if image-verdict == "inappropriate" {

strip-header("Subject");

insert-header("Subject", "[inappropriate image] $Subject");

}

else {

if image-verdict == "suspect" {
```

```
strip-header("Subject");

insert-header("Subject", "[suspect image] $Subject");
}
}
```

Related Topics

- Creating Content Filters to Strip Attachments Based on Image Analysis Verdicts , on page 93

Creating Content Filters to Strip Attachments Based on Image Analysis Verdicts

After you enable image analysis, you can create a content filter to strip attachments based on image analysis verdicts, or you can configure a filter to perform different actions for different message verdicts. For example, you might decide to quarantine messages that contain inappropriate content.

To strip attachments based on image analysis verdicts:

Step 1 Click Mail Policies > Incoming Content Filters.
Step 2 Click Add Filter.
Step 3 Enter a name for the content filter.
Step 4 Under Actions, click **Add Action.**
Step 5 Under Strip Attachment by File Info, click **Image Analysis Verdict is**:
Step 6 Select from the following image analysis verdicts:

- Suspect
- Inappropriate
- Suspect or Inappropriate
- Unscannable
- Clean

Configuring an Action Based on Image Analysis Verdicts

To configure an action based on image analysis verdicts:

Step 1 Click Mail Policies > Incoming Content Filters.
Step 2 Click Add Filter.
Step 3 Enter a name for the content filter.
Step 4 Under Conditions, click **Add Condition**.
Step 5 Under Attachment File Info, click **Image Analysis Verdict**.
Step 6 Choose from one of the following verdicts:

- Suspect
- Inappropriate

- Suspect or Inappropriate
- Unscannable
- Clean

Step 7 Click **Add Action**.

Step 8 Select an action to perform on messages based on the image analysis verdict.

Step 9 Submit and commit your changes.

Notifications

Using the Text Resources page in the GUI or the `textconfig` CLI command to configure custom notification templates as text resources is another useful tool when used in conjunction with attachment filtering rules. The notification template supports non-ASCII characters (you are prompted to choose an encoding while creating the template).

In the following example, the `textconfig` command was first used to create a notification template named `strip.mp3` that will be inserted into to the body of the notification message. Then, an attachment filtering rule is created so that when an .mp3 file has been stripped from a message, a notification email is sent to the intended recipients explaining that the .mp3 file has been deleted.

```
drop-mp3s:

if (attachment-type == '*/mp3')

{ drop-attachments-by-filetype('Media');

notify ('$EnvelopeRecipients', 'Your mp3 has been removed', '$EnvelopeFrom',

'strip.mp3');

}
```

For more information, see Notify and Notify-Copy Actions, on page 71.

Examples of Attachment Scanning Message Filters

The following examples shows actions performed on attachments:

- Inserting Headers, on page 94
- Dropping Attachments by File Type, on page 95
- Dropping Attachments by Dictionary Matches, on page 96
- Quarantining Protected Attachments, on page 97
- Detecting Unprotected Attachments, on page 97

Inserting Headers

In these examples, AsyncOS inserts headers when the attachments contain specified content.

In the following example, all of the attachments on the message are scanned for a keyword. If the keyword is present in all of the attachments, a custom `X-Header` is inserted:

```
attach_disclaim:

if (every-attachment-contains('[dD]isclaimer') ) {

insert-header("X-Example-Approval", "AttachOK");

}
```

In the following example, the attachment is scanned for a pattern in the binary data. The filter uses the `attachment-binary-contains` filter rule to search for a pattern that indicates that the PDF document is encrypted. If the pattern is present in the binary data, a custom header is inserted:

```
match_PDF_Encrypt:

if (attachment-filetype == 'pdf' AND

attachment-binary-contains('/Encrypt')){

strip-header ('Subject');

insert-header ('Subject', '[Encrypted] $Subject');

}
```

Dropping Attachments by File Type

In the following example, the "executable" group of attachments (`.exe` , `.dll` , and `.scr`) is stripped from messages and text is added to the message, listing the filenames of the dropped files (using the `$dropped_filename` action variable). Note that the `drop-attachments-by-filetype` action examines attachments and strips them based on the fingerprint of the file, and not just the three-letter filename extension. Note also that you can specify a single file type ("mpeg") or you can refer to all of the members of the file type ("Media"):

```
strip_all_exes: if (true) {

drop-attachments-by-filetype ('Executable', "Removed attachment:

$dropped_filename");

}
```

In the following example, the same "executable" group of attachments (`.exe` , `.dll` , and `.scr`) are stripped from messages whose Envelope Sender is not within the domain `example.com` .

```
strip_inbound_exes: if (mail-from != "@example\\.com$") {

drop-attachments-by-filetype ('Executable');

}
```

In the following example, a specific member of a file type ("wmf") as well as a the same "executable" group of attachments (`.exe` , `.dll` , and `.scr`) are stripped from messages whose Envelope Sender is not within the domain `example.com` .

```
strip_inbound_exes_and_wmf: if (mail-from != "@example\\.com$") {

drop-attachments-by-filetype ('Executable');

drop-attachments-by-filetype ('x-wmf');

}
```

In the following example, the “executable” pre-defined group of attachments is extended to include more attachment names. (Note that this action will *not* examine the attachments’ file type.)

```
strip_all_dangerous: if (true) {

drop-attachments-by-filetype ('Executable');

drop-attachments-by-name('(?i)\\.(cmd|pif|bat)$');

}
```

The `drop-attachments-by-name` action supports non-ASCII characters.

Note The `drop-attachments-by-name` action matches the regular expression against the filename captured from the MIME header. The filename captured from the MIME header may contain trailing spaces.

In the following example, a message is dropped if the attachment is not an .exe executable file type. However, the filter will not perform any action on the message if there is at least one attachment with the file type you want to filter out. For example, the following filter drops any message with an attachment that is not an **`.exe`** file type:

```
exe_check: if (attachment-filetype != "exe") {

drop();

}
```

If a message has multiple attachments, the appliance does not drop the message if at least one of the attachments is an `.exe` file, even if the other attachments not `.exe` files.

Dropping Attachments by Dictionary Matches

This `drop-attachments-where-dictionary-match` action strips attachments based on matches to dictionary terms. If the terms in the MIME parts considered to be an attachment match a dictionary term (and the user-defined threshold is met), the attachment is stripped from the email. The following example shows attachment drops if words in the “secret_words” dictionary are detected in the attachment. Note that the threshold for the matches is set to one:

```
Data_Loss_Prevention: if (true) {

drop-attachments-where-dictionary-match("secret_words", 1);

}
```

Quarantining Protected Attachments

The `attachment-protected` filter tests whether any attachment in the message is password protected. You might use this filter on incoming mail to ensure that the attachments are scannable. According to this definition, a zip file containing one encrypted member along with unencrypted members will be considered protected. Similarly, PDF file that has no open password will not be considered protected, even though it may restrict copying or printing with a password. The following example shows protected attachments sent to a policy quarantine:

```
quarantine_protected:

if attachment-protected

{

quarantine("Policy");

}
```

Detecting Unprotected Attachments

The `attachment-unprotected` filter tests whether any attachment in the message is *not* password protected. This message filter complements the `attachment-protected` filter. You might use this filter on outgoing mail to detect outgoing mail that is unprotected. The following example shows AsyncOS detecting unprotected attachments on an outgoing listener and quarantining the messages:

```
quarantine_unprotected:

if attachment-unprotected

{

quarantine("Policy");

}
```

Detecting Malicious Files in Messages Attachments Using Message Filter

As an example, use the following message filter rule syntax to detect files in message attachments categorized as malicious by the ETF engine, and take appropriate actions on such messages.

Syntax:

```
Strip_malicious_files: if (file-hash-etf-rule (['etf_source1'], <'file_hash_exception_list'>))

{ file-hash-etf-strip-attachment-action (['etf_source1'], <'file_hash_exception_list>,
 "file stripped from message attachment"); }
```

Where:

- `'file-hash-etf-rule'` is the Attachment File Info message filter rule
- `'etf_source1'` is the ETF source(s) used to detect malicious files in the messages based on the file hash.

- `'file_hash_exception_list'` is the name of a file hash exception list. If a file hash exception list is not present, it is displayed as “”.
- `'file-hash-etf-strip-attachment-action'` is the name of the action that you want to apply on messages that contain malicious files.

In the following example, if a message contains a message attachment detected as malicious by the ETF engine, the attachment is stripped.

```
Strip_Malicious_Attachment: if (true) {file-hash-etf-strip-attachment-action
(['threat_feed_source'], "", "Malicious message attachment has been stripped from
the message.");}
```

Using the CLI to Manage Message Filters

You can use the CLI to add, delete, activate and de-activate, import and export, and set logging options for message filters. The table below shows a summary of the commands and subcommands. The table below shows a summary of the commands and subcommands.

Table 9: Message Filters Subcommands

Syntax	Description
`filters`	The main command. This command is interactive; it asks you for more information (for example, new , delete , import).
`new`	Creates a new filter. If no location is given, it is appended to the current sequence. Otherwise, the filter will be inserted into the specific place in the sequence. For more information, see Creating a New Message Filter, on page 99.
`delete`	Deletes a filter by name or by sequence number. For more information, see Deleting a Message Filter, on page 100.
`move`	Rearranges the existing filters. For more information, see Creating a New Message Filter, on page 99.
`set`	Sets filter to active or inactive state. For more information, see Creating a New Message Filter, on page 99.
`import`	Replaces the current set of filters with a new set stored in a file (in the /configuration directory of the appliance). For more information, see Creating a New Message Filter, on page 99.
`export`	Exports the current set of filters to a file (in the /configuration directory of the appliance). For more information, see Exporting Message Filters, on page 104 .
`list`	Lists information about a filter or filters. For more information, see Displaying a Message Filter List, on page 104.
`detail`	Prints detailed information about a specific filter, including the body of the filter rule itself. For more information, see Displaying Message Filter Details, on page 104 .

Syntax	Description
`logconfig`	Enters the logconfig submenu of filters, allowing you to edit the log subscriptions from archive() filter actions. For more information, see Configuring Filter Log Subscriptions, on page 105.

Note You must issue the commit command for filters to take effect.

Three types of parameters are:

Table 10: Filter Management Parameters

seqnum	An integer representing a filter based on its position in the list of filters. A *seqnum* of 2 represents the second filter in the list, for example.
filtname	The colloquial name of a filter.
range	A range may be used to represent more than one filter, and appears in the form of *X Y* , where X and Y are the first and last *seqnums* that identify the extent. For example, `2-4` represents filters in the second, third, and fourth positions. Either X or Y may be left off to represent an open-ended list. For example, `-4` represents the first four filters, and `2-` represents all filters except the first. You can also use the keyword all to represents all the filters in the filter list.

Related Topics

- Creating a New Message Filter, on page 99
- Deleting a Message Filter, on page 100
- Moving a Message Filter, on page 100
- Activating and Deactivating a Message Filter, on page 100
- Importing Message Filters, on page 103
- Exporting Message Filters, on page 104
- Viewing Non-ASCII Character Sets, on page 104
- Displaying a Message Filter List, on page 104
- Displaying Message Filter Details, on page 104
- Configuring Filter Log Subscriptions, on page 105
- Changing Message Encoding, on page 106
- Sample Message Filters, on page 107

Creating a New Message Filter

`new [seqnum|filtname|last]`

Specifies the position at which to insert the new filter(s). If omitted, or given the keyword last , the filters entered in are appended to the list of filters. No gaps in the sequence numbers are allowed; you are not allowed to enter a *seqnum* outside the boundaries of the current list. If you enter an unknown *filtname* , you are prompted to enter a valid *filtname* , *seqnum* , or last .

After a filter has been entered, you may manually enter the filter script. When you are finished typing, end the entry by typing a period (.) on a line by itself.

The following conditions can cause errors:

- Sequence number beyond the current range of sequence numbers.
- Filter with a non-unique *filtname* .
- Filter with a *filtname* that is a reserved word.
- Filter with a syntax error.
- Filter with actions referring to non-existent system resources such as interfaces.

Deleting a Message Filter

`delete [seqnum|filtname|range]`

Deletes the filter(s) identified.

The following conditions can cause errors:

- No filter with a given filter name.
- No filter with a given sequence number.

Moving a Message Filter

`move [seqnum|filtname|rangeseqnum|last]`

Moves the filters identified by the first parameter to the position identified by the second parameter. If the second parameter is the keyword last , the filters are moved to the end of the list of filters. If more than one filter is being moved, their ordering remains the same in relation to one another.

The following conditions can cause errors:

- No filter with a given filter name.
- No filter with a given sequence number.
- Sequence number beyond the current range of sequence numbers.
- Movement would result in no change of sequence.

Activating and Deactivating a Message Filter

A given message filter is either *active* or *inactive* and it is also either *valid* or *invalid* . A message filter is only used for processing if it is both *active* and *valid* . You change an existing filter from active to inactive (and back again) using the CLI. A filter is invalid if it refers to a listener or interface which does not exist (or has been removed).

Note You can determine if a filter is inactive by its syntax; AsyncOS changes the colon after the filter name to an exclamation point for inactive filters. If you use this syntax when entering or importing a filter, AsyncOS marks the filter as inactive.

For example, the following benign filter named “filterstatus” is entered. It is then made inactive using the `filter -> set` subcommand. Note that when the details of the filter are shown, the colon has been changed to an exclamation point (and is bold in the following example).

```
mail3.example.com> filters

Choose the operation you want to perform:

- NEW - Create a new filter.

- IMPORT - Import a filter script from a file.

[]> new

Enter filter script. Enter '.' on its own line to end.

filterstatus: if true{skip-filters();}

.

1 filters added.

Choose the operation you want to perform:

- NEW - Create a new filter.

- DELETE - Remove a filter.

- IMPORT - Import a filter script from a file.

- EXPORT - Export filters to a file

- MOVE - Move a filter to a different position.

- SET - Set a filter attribute.

- LIST - List the filters.

- DETAIL - Get detailed information on the filters.

- LOGCONFIG - Configure log subscriptions used by filters.

- ROLLOVERNOW - Roll over a filter log file.

[]> list

Num Active Valid Name

1 Y Y filterstatus

Choose the operation you want to perform:

- NEW - Create a new filter.

- DELETE - Remove a filter.

- IMPORT - Import a filter script from a file.

- EXPORT - Export filters to a file

- MOVE - Move a filter to a different position.

- SET - Set a filter attribute.
```

```
- LIST - List the filters.
- DETAIL - Get detailed information on the filters.
- LOGCONFIG - Configure log subscriptions used by filters.
- ROLLOVERNOW - Roll over a filter log file.
[]> set
Enter the filter name, number, or range:
[all]> all
Enter the attribute to set:
[active]> inactive
1 filters updated.
Choose the operation you want to perform:
- NEW - Create a new filter.
- DELETE - Remove a filter.
- IMPORT - Import a filter script from a file.
- EXPORT - Export filters to a file
- MOVE - Move a filter to a different position.
- SET - Set a filter attribute.
- LIST - List the filters.
- DETAIL - Get detailed information on the filters.
- LOGCONFIG - Configure log subscriptions used by filters.
- ROLLOVERNOW - Roll over a filter log file.
[]> detail
Enter the filter name, number, or range:
[]> all
Num Active Valid Name
1 N Y filterstatus
filterstatus! if (true) {
skip-filters();
}
Choose the operation you want to perform:
- NEW - Create a new filter.
- DELETE - Remove a filter.
```

```
- IMPORT - Import a filter script from a file.

- EXPORT - Export filters to a file

- MOVE - Move a filter to a different position.

- SET - Set a filter attribute.

- LIST - List the filters.

- DETAIL - Get detailed information on the filters.

- LOGCONFIG - Configure log subscriptions used by filters.

- ROLLOVERNOW - Roll over a filter log file.

[]>
```

Related Topics

- Activating or Deactivating a Message Filter, on page 103

Activating or Deactivating a Message Filter

`set [seqnum|filtname|range] active|inactive`

Sets the filters identified to have the given state. Legal states are:

- active: Set the state of the selected filters to be active.
- inactive: Set the state of the selected filters to be inactive.

The following conditions can cause errors:

- No filter with a given *filtname* .
- No filter with a given sequence number.

Note A filter which is inactive may also be noted in its syntax; the colon after the label (name of the filter) is changed to an exclamation point (!). A filter entered manually from the CLI, or imported, that contains this syntax, will automatically be marked inactive. For example, mailfrompm! instead of mailfrompm: is displayed.

Importing Message Filters

`import filename`

The name of the file containing filters to be processed. This file must reside in the configuration directory of the FTP/SCP root directory on the appliance , if you enabled FTP/SCP access for the interface with the `interfaceconfig` command. It is ingested and parsed, and any errors are reported. The filters imported replace all filters existing in the current filter set. See FTP, SSH, and SCP Access for more information. Consider exporting the current filter list (see Exporting Message Filters, on page 104) and then editing that file before importing.

When importing message filters, you are prompted to select the encoding used.

The following conditions can cause errors:

- File does not exist.
- Filter with a non-unique filter name.
- Filter with a *filtname* that is a reserved word.
- Filter with a syntax error.
- Filter with actions referring to non-existent system resources such as interfaces.

Exporting Message Filters

```
export filename[seqnum|filtname|range]
```

Output a formatted version of the existing filter set to a file in the configuration directory of the FTP/SCP root directory on the appliance . See FTP, SSH, and SCP Access for more information.

When exporting message filters, you are prompted to select the encoding used.

The following conditions can cause errors:

- No filter with a given filter name.
- No filter with a given sequence number.

Viewing Non-ASCII Character Sets

The system displays filters containing non-ASCII characters in the CLI in UTF-8. If your terminal/display does not support UTF-8, the filter will unreadable.

The best way to manage non-ASCII characters in filters is to edit the filter in a text file and then import that text file (see Importing Message Filters, on page 103) into the appliance .

Displaying a Message Filter List

```
list [seqnum|filtname|range]
```

Shows summarized information about the identified filters in a tabular form without printing the filter body. The information displayed includes:

- Filter name
- Filter sequence number
- Filter's active/inactive state
- Filter’s valid/invalid state

The following conditions can cause errors:

- Illegal range format.

Displaying Message Filter Details

```
detail [seqnum|filtname|range]
```

Provides full information about the identified filters, including the body of the filter and any additional state information.

Configuring Filter Log Subscriptions

`logconfig`

Enters a submenu that allows you to configure the filter log options for the mailbox files generated by the `archive()` action. These options are very similar to those used by the regular `logconfig` command, but the logs may only be created or deleted by adding or removing filters that reference them.

Each filter log subscription has the following default values, which can be modified using the logconfig subcommand:

- Retrieval method - FTP Poll
- File size - 10MB
- Max number of files - 10

For more information, see the "Logging" chapter.

```
mail3.example.com> filters

Choose the operation you want to perform:

- NEW - Create a new filter.

- DELETE - Remove a filter.

- IMPORT - Import a filter script from a file.

- EXPORT - Export filters to a file

- MOVE - Move a filter to a different position.

- SET - Set a filter attribute.

- LIST - List the filters.

- DETAIL - Get detailed information on the filters.

- LOGCONFIG - Configure log subscriptions used by filters.

- ROLLOVERNOW - Roll over a filter log file.

[]> logconfig

Currently configured logs:

1. "joesmith" Type: "Filter Logs" Retrieval: FTP Poll

Choose the operation you want to perform:

- EDIT - Modify a log setting.


[]> edit

Enter the number of the log you wish to edit.

[]> 1

Choose the method to retrieve the logs.

1. FTP Poll
```

```
2. FTP Push

3. SCP Push

[1]> 1

Please enter the filename for the log:

[joesmith.mbox]>

Please enter the maximum file size:

[10485760]>

Please enter the maximum number of files:

[10]>

Currently configured logs:

1. "joesmith" Type: "Filter Logs" Retrieval: FTP Poll

Enter "EDIT" to modify or press Enter to go back.

[]>
```

Changing Message Encoding

You can use the `localeconfig` command to set the behavior of AsyncOS regarding modifying the encoding of message headings and footers during message processing:

```
example.com> localeconfig

Behavior when modifying headers: Use encoding of message body
Behavior for untagged non-ASCII headers: Impose encoding of message body
Behavior for mismatched footer or heading encoding: Try both body and footer or heading
encodings
Behavior when decoding errors found: Disclaimer is displayed as inline content and the
message body is added as an attachment.

Choose the operation you want to perform:
- SETUP - Configure multi-lingual settings.
[]> setup

If a header is modified, encode the new header in the same encoding as the message body?
(Some MUAs incorrectly handle headers encoded in a different encoding than the body.
However, encoding a modified header in the same encoding as the message body may cause
certain
characters in the modified header to be lost.) [Y]>

If a non-ASCII header is not properly tagged with a character set and is being used or
modified,
impose the encoding of the body on the header during processing and final representation
of the message?
(Many MUAs create non-RFC-compliant headers that are then handled in an undefined way.
Some MUAs handle headers encoded in character sets that differ from that of the main body
in an incorrect way.
Imposing the encoding of the body on the header may encode the header more precisely.
This will be used to interpret the content of headers for processing, it will not modify
or rewrite the
header unless that is done explicitly as part of the processing.) [Y]>
```

```
Disclaimers (as either footers or headings) are added in-line with the message body whenever
 possible.
However, if the disclaimer is encoded differently than the message body, and if imposing a
 single encoding
will cause loss of characters, it will be added as an attachment. The system will always
try to use the
message body's encoding for the disclaimer. If that fails, the system can try to edit the
message body to
use an encoding that is compatible with the message body as well as the disclaimer. Should
 the system try to
re-encode the message body in such a case? [Y]>

If the disclaimer that is added to the footer or header of the message generates an error
when decoding the message body,
it is added at the top of the message body. This prevents you to rewrite a new message
content that must merge with
the original message content and the header/footer-stamp. The disclaimer is now added as
an additional MIME part
that displays only the header disclaimer as an inline content, and the rest of the message
 content is split into
separate email attachments. Should the system try to ignore such errors when decoding the
message body? [N]>

Behavior when modifying headers: Use encoding of message body
Behavior for untagged non-ASCII headers: Impose encoding of message body
Behavior for mismatched footer or heading encoding: Try both body and footer or heading
encodings
Behavior when decoding errors found: Disclaimer is displayed as inline content and the
message body
is added as an attachment.

Choose the operation you want to perform:
- SETUP - Configure multi-lingual settings.
[]>
```

The first prompt determines whether or not a message header's encoding should be changed to match that of the message body if the header is changed (via a filter, for example).

The second prompt controls whether or not the appliance should impose the encoding of the message body on the header if the header is not properly tagged with a character set.

The third prompt is used to configure how disclaimer stamping (and multiple encodings) in the message body works. Please see "Disclaimer Stamping and Multiple Encodings" in the "Text Resources" chapter for more information.

The fourth prompt is used to configure the behaviour of disclaimer stamping, if an error is generated during the decoding of the message body. If you select 'Yes', the decoding errors are ignored and the disclaimer is stamped. If you select 'No', the disclaimer text is added as an attachment to the message.

Sample Message Filters

In the following example, the filter command is used to create three new filters:

- The first filter is named **`big_messages`**. It uses the `body-size` rule to drop messages larger than 10 megabytes.
- The second filter is named **`no_mp3s`**. It uses the `attachment-filename` rule to drop messages that contain attachments with the filename extension of `.mp3` .
- The third filter is named **`mailfrompm`**. It uses mail-from rule examines all mail from `postmaster@example.com` and blind-carbon copies `administrator@example.com` .

Using the `filter -> list` subcommand, the filters are listed to confirm that they are active and valid, and then the first and last filters are switched in position using the `move` subcommand. Finally, the changes are committed so that the filters take effect.

```
mail3.example.com> filters

Choose the operation you want to perform:

- NEW - Create a new filter.

- IMPORT - Import a filter script from a file.

[]> new

Enter filter script. Enter '.' on its own line to end.

big_messages:

if (body-size >= 10M) {

drop();

}

.

1 filters added.

Choose the operation you want to perform:

- NEW - Create a new filter.

- DELETE - Remove a filter.

- IMPORT - Import a filter script from a file.

- EXPORT - Export filters to a file

- MOVE - Move a filter to a different position.

- SET - Set a filter attribute.

- LIST - List the filters.

- DETAIL - Get detailed information on the filters.

- LOGCONFIG - Configure log subscriptions used by filters.

- ROLLOVERNOW - Roll over a filter log file.

[]> new

Enter filter script. Enter '.' on its own line to end.

no_mp3s:

if (attachment-filename == '(?i)\\.mp3$') {

drop();

}

.
```

```
1 filters added.

Choose the operation you want to perform:

- NEW - Create a new filter.

- DELETE - Remove a filter.

- IMPORT - Import a filter script from a file.

- EXPORT - Export filters to a file

- MOVE - Move a filter to a different position.

- SET - Set a filter attribute.

- LIST - List the filters.

- DETAIL - Get detailed information on the filters.

- LOGCONFIG - Configure log subscriptions used by filters.

- ROLLOVERNOW - Roll over a filter log file.

[]> new

Enter filter script. Enter '.' on its own line to end.

mailfrompm:

if (mail-from == "^postmaster$")

{ bcc ("administrator@example.com");}

.

1 filters added.

Choose the operation you want to perform:

- NEW - Create a new filter.

- DELETE - Remove a filter.

- IMPORT - Import a filter script from a file.

- EXPORT - Export filters to a file

- MOVE - Move a filter to a different position.

- SET - Set a filter attribute.

- LIST - List the filters.

- DETAIL - Get detailed information on the filters.

- LOGCONFIG - Configure log subscriptions used by filters.

- ROLLOVERNOW - Roll over a filter log file.

[]> list

Num Active Valid Name
```

```
1 Y Y big_messages

2 Y Y no_mp3s

3 Y Y mailfrompm

Choose the operation you want to perform:

- NEW - Create a new filter.

- DELETE - Remove a filter.

- IMPORT - Import a filter script from a file.

- EXPORT - Export filters to a file

- MOVE - Move a filter to a different position.

- SET - Set a filter attribute.

- LIST - List the filters.

- DETAIL - Get detailed information on the filters.

- LOGCONFIG - Configure log subscriptions used by filters.

- ROLLOVERNOW - Roll over a filter log file.

[]> move

Enter the filter name, number, or range to move:

[]> 1

Enter the target filter position number or name:

[]> last

1 filters moved.

Choose the operation you want to perform:

- NEW - Create a new filter.

- DELETE - Remove a filter.

- IMPORT - Import a filter script from a file.

- EXPORT - Export filters to a file

- MOVE - Move a filter to a different position.

- SET - Set a filter attribute.

- LIST - List the filters.

- DETAIL - Get detailed information on the filters.

- LOGCONFIG - Configure log subscriptions used by filters.

- ROLLOVERNOW - Roll over a filter log file.

[]> list
```

```
Num Active Valid Name

1 Y Y no_mp3s

2 Y Y mailfrompm

3 Y Y big_messages

Choose the operation you want to perform:

- NEW - Create a new filter.

- DELETE - Remove a filter.


- IMPORT - Import a filter script from a file.

- EXPORT - Export filters to a file

- MOVE - Move a filter to a different position.

- SET - Set a filter attribute.

- LIST - List the filters.

- DETAIL - Get detailed information on the filters.

- LOGCONFIG - Configure log subscriptions used by filters.

- ROLLOVERNOW - Roll over a filter log file.

[]> move

Enter the filter name, number, or range to move:

[]> 2

Enter the target filter position number or name:

[]> 1

1 filters moved.

Choose the operation you want to perform:

- NEW - Create a new filter.

- DELETE - Remove a filter.

- IMPORT - Import a filter script from a file.

- EXPORT - Export filters to a file

- MOVE - Move a filter to a different position.

- SET - Set a filter attribute.

- LIST - List the filters.

- DETAIL - Get detailed information on the filters.

- LOGCONFIG - Configure log subscriptions used by filters.

- ROLLOVERNOW - Roll over a filter log file.
```

```
[]> list

Num Active Valid Name

1 Y Y mailfrompm

2 Y Y no_mp3s

3 Y Y big_messages

Choose the operation you want to perform:

- NEW - Create a new filter.

- DELETE - Remove a filter.

- IMPORT - Import a filter script from a file.

- EXPORT - Export filters to a file


- MOVE - Move a filter to a different position.

- SET - Set a filter attribute.

- LIST - List the filters.


- DETAIL - Get detailed information on the filters.

- LOGCONFIG - Configure log subscriptions used by filters.

- ROLLOVERNOW - Roll over a filter log file.

[]>

mail3.example.com> commit

Please enter some comments describing your changes:

[]> entered and enabled 3 filters: no_mp3s, mailfrompm, big_messages

Do you want to save the current configuration for rollback? [Y]> n

Changes committed: Fri May 23 11:42:12 2014 GMT
```

Message Filter Examples

This section contains some real world examples of filters with a brief discussion of each.

Related Topics

- Open-Relay Prevention Filter, on page 113
- Policy Enforcement Filters, on page 113
- Routing and Domain Spoofing, on page 117

Open-Relay Prevention Filter

This filter bounces messages with addresses using % , extra @ , and ! characters in email addresses:

- `user%otherdomain@validdomain`
- `user@otherdomain@validdomain:`
- `domain!user@validdomain`

```
sourceRouted:

if (rcpt-to == "(%|@|!)(.*)@") {

bounce();

}
```

The appliances are not susceptible to these third party relay hacks that are often used to exploit traditional Sendmail/Qmail systems. As many of these symbols (for example %) can be part of a perfectly legal email address, appliances will accept these as valid addresses, verify them against the configured recipient lists, and pass them on to the next internal server. The appliances do not relay these messages to the world.

These filters are put in place to protect users who may have open-source MTAs that are misconfigured to allow relay of these types of messages.

Note You can also configure a listener to handle these types of addresses. See Listening for Connection Requests by Creating a Listener Using Web Interface for more information.

Policy Enforcement Filters

- Notify Based on Subject Filter, on page 113
- BCC and Scan Mail Sent to Competitors, on page 114
- Block Specific User Filter, on page 114
- Archive and Drop Messages Filter, on page 114
- Large “To:” Header Filter, on page 115
- Blank “From:” Filter, on page 115
- SBRS Filter, on page 115
- Alter SBRS Filter, on page 116
- Filename Regex Filter, on page 116
- Show SenderBase Reputation Score in Header Filter, on page 116
- Insert Policy into Header Filter, on page 116
- Too Many Recipients Bounce Filter, on page 116

Notify Based on Subject Filter

This filter sends notification based on whether the subject contains specific words:

```
search_for_sensitive_content:


if (Subject == "(?i)plaintiff|lawsuit|judge" ) {
```

```
notify ("admin@company.com");

}
```

BCC and Scan Mail Sent to Competitors

This filter scans and blind copies messages that are sent to competitors. Note that you could use a dictionary and the `header-dictionary-match()` rule to specify a more flexible list of competitors (see Dictionary Rules, on page 37):

```
competitorFilter:

if (rcpt-to == '@competitor1.com|@competitor2.com') {

bcc-scan('legal@example.com');

}
```

Block Specific User Filter

Use this filter to block email from a specific address:

```
block_harrasing_user:

if (mail-from == "ex-employee@hotmail\\.com") {

notify ("admin@company.com");

drop ();

}
```

Archive and Drop Messages Filter

Log and drop only the messages that have matching filetypes:

```
drop_attachments:

if (mail-from != "user@example.com") AND (attachment-filename ==

'(?i)\\.(asp|bas|bat|cmd|cpl|exe|hta|ins|isp|js)$')

{

archive("Drop_Attachments");

insert-header("X-Filter", "Dropped by: $FilterName MID: $MID");

drop-attachments-by-name("\\.(asp|bas|bat|cmd|cpl|exe|hta|ins|isp|js)$");

}
```

Large “To:” Header Filter

Find messages with very large “To” headers.

Use the `archive()` line for verification of proper action, with drop() enabled or disabled for extra safety:

```
toTooBig:

if(header('To') == "^.{500,}") {

archive('tooTooBigdropped');

drop();

}
```

Blank “From:” Filter

Identify blank “From” headers,

This filter can alleviate various forms of blank “from” addresses:

```
blank_mail_from_stop:

if (recv-listener == "InboundMail" AND header("From") == "^$|<\\s*>") {

drop ();

}
```

If you also want to drop messages with a blank envelope from, use this filter:

```
blank_mail_from_stop:

if (recv-listener == "InboundMail" AND (mail-from == "^$|<\\s*>" OR header ("From") ==
"^$|<\\s*>"))

{

drop ();
}
```

SBRS Filter

SenderBase Reputation filter:

```
note_bad_reps:
if (reputation < -2) {
strip-header ('Subject');

insert-header ('Subject', '***BadRep $Reputation *** $Subject');
}
```

Alter SBRS Filter

Alter the (SenderBase Reputation Score) SBRS threshold for certain domains:

```
mod_ipr:
if ( (rcpt-count == 1) AND (rcpt-to == "@domain\\.com$") AND (reputation < -2) ) {
drop ();
}
```

Filename Regex Filter

This filter specifies a range of size for the body of the message, and looks for an attachment that matches the regular expression (this matches files named “readme.zip”, “readme.exe”, “attach.exe”, and so forth.):

```
filename_filter:
if ((body-size >= 9k) AND (body-size <= 20k)) {
if (body-contains ("(?i)(readme|attach|information)\\.(zip|exe)$")) {
drop ();

}

}
```

Show SenderBase Reputation Score in Header Filter

Remember to log the headers (see the “Logging” chapter) so they appear in the mail log:

```
Check_ipr:

if (true) {

insert-header('X-ipr', '$Reputation');

}
```

Insert Policy into Header Filter

Show which mail flow policy accepted the connection:

```
Policy_Tracker:

if (true) {

insert-header ('X-HAT', 'Sender Group $Group, Policy $Policy applied.');

}
```

Too Many Recipients Bounce Filter

Bounce all outbound email messages with more than 50 recipients from more than two unique domains:

```
bounce_high_rcpt_count:
```

```
if ( (rcpt-count > 49) AND (rcpt-to != "@example\\.com$") ) {

bounce-profile ("too_many_rcpt_bounce"); bounce ();

}
```

Routing and Domain Spoofing

- Using Virtual Gateways Filter, on page 117
- Same Listener for Deliver and Listener Filter, on page 117
- Single Listener Filter, on page 117
- Drop Spoofed Domain Filter (Single Listener), on page 118
- Drop Spoofed Domain Filter (Multiple Listeners), on page 118
- Another Drop Spoofed Domain Filter, on page 118
- Detect Looping Filter, on page 119

Using Virtual Gateways Filter

Segment traffic using virtual gateways. Assuming you have two Interfaces on the system, 'public1' and 'public2', and the default delivery interface is 'public1'. This would force all of your outbound traffic over the second interface; since bounces and other similar types of mail do not go through filters, they will be delivered from public1:

```
virtual_gateways:

if (recv-listener == "OutboundMail") {

alt-src-host ("public2");

}
```

Same Listener for Deliver and Listener Filter

Use the same listener for delivery and receiving. This filter will allow you to send any messages received on the public listener “listener1” out the interface “listener1” (you will have to set up a unique filter for each public listener configured):

```
same_listener:

if (recv-inj == 'listener1') {

alt-src-host('listener1');

}
```

Single Listener Filter

Make the filter work on a single listener. For example, specify a specific listener for message filter processing instead of being performed system wide.

```
textfilter-new:
```

```
if (recv-inj == 'inbound' and body-contains("some spammy message")) {
alt-rcpt-to ("spam.quarantine@spam.example.com");
}
```

Drop Spoofed Domain Filter (Single Listener)

Drop email with a spoofed domain (pretending to be from an internal address; works with a single listener). IP addresses below represent fictional domain for `mycompany.com` :

```
DomainSpoofed:
if (mail-from == "mycompany\\.com$") {
if ((remote-ip != "1.2.") AND (remote-ip != "3.4.")) {
drop();
}
}
```

Drop Spoofed Domain Filter (Multiple Listeners)

As above, but works with multiple listeners:

```
domain_spoof:
if ((recv-listener == "Inbound") and (mail-from == "@mycompany\\.com")) {
archive('domain_spoof');
drop ();
}
```

Another Drop Spoofed Domain Filter

Summary: Anti domain spoof filter:

```
reject_domain_spoof:
if (recv-listener == "MailListener") {
insert-header("X-Group", "$Group");
if ((mail-from == "@test\\.mycompany\\.com") AND (header("X-Group") != "RELAYLIST")) {
notify("me@here.com");
drop();
strip-header("X-Group");
```

```
}
```

Detect Looping Filter

This filter is used to detect, stop, and determine what is causing, a mail loop. This filter can help determine a configuration issue on the Exchange server or elsewhere.

```
External_Loop_Count:

if (header("X-ExtLoop1")) {


if (header("X-ExtLoopCount2")) {

if (header("X-ExtLoopCount3")) {

if (header("X-ExtLoopCount4")) {

if (header("X-ExtLoopCount5")) {

if (header("X-ExtLoopCount6")) {

if (header("X-ExtLoopCount7")) {

if (header("X-ExtLoopCount8")) {

if (header("X-ExtLoopCount9")) {

notify ('joe@example.com');

drop();

}

else {insert-header("X-ExtLoopCount9", "from
$RemoteIP");}}

else {insert-header("X-ExtLoopCount8", "from $RemoteIP");}}

else {insert-header("X-ExtLoopCount7", "from $RemoteIP");}}

else {insert-header("X-ExtLoopCount6", "from $RemoteIP");}}

else {insert-header("X-ExtLoopCount5", "from $RemoteIP");}}

else {insert-header("X-ExtLoopCount4", "from $RemoteIP");}}

else {insert-header("X-ExtLoopCount3", "from $RemoteIP");}}

else {insert-header("X-ExtLoopCount2", "from $RemoteIP");}}

else {insert-header("X-ExtLoop1", "1");

}
```

Note By default, AsyncOS automatically detects mail loops and will drop messages after 100 loops.

Configuring Scan Behavior

You can control the behavior of body and attachment scanning, such as the attachment types to be skipped during a scan by configuring the scanning parameters. Use the Scan Behavior page or the `scanconfig` command to configure these parameters. Scan behavior settings are global settings, meaning that they affect the behavior of all the scans.

Note If you want to scan a MIME type that may be included in a zip or compressed file, you must include list 'compressed' or 'zip' or 'application/zip' in the scan list.

Step 1 Click **Security Services** > **Scan Behavior**.

Step 2 Define the attachment type mapping. Do one of the following:

- Add a new attachment type mapping. Click **Add Mapping**.
- Import a list of attachment type mappings using a configuration file. Click **Import List**, and import the desired configuration file from the configuration directory.

  Note To perform this step, the configuration file must be present in the configuration directory of your appliance. See Managing the Configuration File.

- Click **Edit** to modify an existing attachment type mapping.

Step 3 Configure the global settings. Do the following:

a) Under Global Settings, click **Edit Global Settings**.

b) Edit the required fields:

Field	Description
Action for attachments with MIME types / fingerprints in table above	Choose whether to scan or skip attachments types defined in the attachment type mapping.
Maximum depth of attachment recursion to scan	Specify the level up to which the recursive attachments are to be scanned.
Maximum attachment size to scan	Specify the maximum size of attachments to scan.
Attachment Metadata scan	Specify whether to scan or skip metadata of the attachments.
Attachment scanning timeout	Specify the scanning time-out period.
Assume attachment matches pattern if not scanned for any reason	Specify whether to consider unscanned attachments as match to the search pattern.
Action when message cannot be deconstructed to remove specified attachments	Specify the action to be taken when a message could not be deconstructed to remove specified attachments.
Bypass all filters in case of a content or message filter error	Specify whether to bypass all filters in case of a content or message filter error.

Field	Description
Encoding to use when none is specified	Specify the encoding to be used if no encoding is specified.
Convert opaque-signed messages to clear-signed (S/MIME unpacking)	Specify whether to convert opaque-signed messages to clear-signed (S/MIME unpacking).
Scanning of Password-protected Attachments	
Actions for Unscannable Messages due to decoding errors found during URL Filtering Actions	Specify the actions to take when a message cannot be scanned by the Content Scanner due to decoding errors found during URL filtering actions.
Action for unscannable messages due to extraction failures	Specify the actions to take when a message cannot be scanned by the Content Scanner because of an attachment extraction failure.
Action for unscannable messages due to RFC violations	Specify the actions to take when a message cannot be scanned by the Content Scanner because of an RFC violation.

c) Click **Submit**.

Step 4 (Optional) Manually update the Content Scanner files. Under **Current Content Scanner files**, click **Update Now**.

Usually, these files are automatically updated using update server.

Note You can also use the `contentscannerupdate` in CLI to manually update these files.

Step 5 Commit the changes.

Configuring Message Handling Actions for Unscannable Messages

The Content Scanner in your appliance can now handle messages that are not scanned due to the following reasons:

- File extraction failure
- RFC violation
- Decoding errors found during URL Filtering actions

You can configure any one of the following message handling actions on messages that are not scanned by the Content Scanner:

- Drop the message
- Deliver the message as it is
- Send the message to the policy quarantine

You can click on the **Edit Global Settings** button in the Security Services > Scan Behavior page of the web interface, to enable and configure message handling actions on messages that are not scanned by the Content Scanner.

Delivering the Message

You can perform the following additional actions, if you choose to deliver the message:

- Modify the message subject
- Add a custom header to the message
- Modify the message recipient
- Send message to alternate destination host

Note These actions are not mutually exclusive; you can combine some or all of them differently within different incoming or outgoing policies for different processing needs for groups of users.

Modifying Message Subject

You can alter the text of messages that are not scanned by the Content Scanner by prepending or appending certain text strings to help users easily identify and sort identified messages.

Note White space is not ignored in the “Modify message subject” field. Add spaces after (if prepending) or before (if appending) the text you enter in this field to separate your added text from the original subject of the message. For example, add the text `[WARNING: UNSCANNABLE EXTRACTION FAILURE]` with a few trailing spaces if you are prepending.

The default text that is added to the subject of the message that is not scanned by the Content Scanner:

Reason	Default Text Added to Subject
Extraction failure	[WARNING: UNSCANNABLE EXTRACTION FAILED]
RFC Violation	[WARNING: UNSCANNABLE RFC NON-COMPLIANT]
Decoding errors found during URL Filtering actions	[WARNING: DECODING ERRORS WHEN APPLYING URL FILTERING ACTIONS]

Adding Custom Header to Message

You can define an additional, custom header to add to all messages that are not scanned by the Content Scanner. Click **Yes** and define the header name and text.

Modifying Message Recipient

You can modify the message recipient, causing the message that is not scanned by the Content Scanner to be delivered to a different address. Click **Yes** and enter the new recipient address.

Sending Message to Alternate Destination Host

You can choose to send the notification to a different recipient or destination host for messages that are not scanned by the Content Scanner. Click **Yes** and enter an alternate address or host.

For example, you can route messages that are not scanned by the Content Scanner to an administrator's mailbox or a special mail server for subsequent examination. In the case of a multi-recipient message, only a single copy is sent to the alternative recipient.

Sending Message to Policy Quarantine

When flagged for quarantine, the message that is not scanned by the Content Scanner continues through the rest of the email pipeline. When the message reaches the end of the pipeline, if the message has been flagged for one or more quarantines then it enters those queues. Note that if the message does not reach the end of the pipeline, it is not placed in a quarantine.

For example, a content filter can cause a message to be dropped or bounced, in which case the message will not be quarantined.

Note If a policy quarantine is not defined in your appliance , you cannot sent the message to the quarantine.

You can perform the following additional actions, if you choose to send the message to the policy quarantine:

- Modify the message subject
- Add a custom header to the message

Modifying Message Subject Header

You can alter the text of messages that are sent to the policy quarantine by prepending or appending certain text strings to help users easily identify and sort identified messages.

Note White space is not ignored in the "Modify message subject" field. Add spaces after (if prepending) or before (if appending) the text you enter in this field to separate your added text from the original subject of the message. For example, add the text `[WARNING: UNSCANNABLE EXTRACTION FAILURE]` with a few trailing spaces if you are prepending.

The default text that is added to the subject of the message that is sent to the policy quarantine:

Reason	Default Text Added to Subject
Extraction failure	[WARNING: UNSCANNABLE EXTRACTION FAILED]
RFC Violation	[WARNING: UNSCANNABLE RFC NON-COMPLIANT]
Decoding errors found during URL Filtering actions	[WARNING: DECODING ERRORS WHEN APPLYING URL FILTERING ACTIONS]

Adding Custom Header to Message

You can define an additional, custom header to add to all messages that are sent to the policy quarantine. Click **Yes** and define the header name and text.