



NetWitness Log Parser Tool User Guide

for Version 1.1



Contact Information

RSA Link at <https://community.rsa.com> contains a knowledgebase that answers common questions and provides solutions to known problems, product documentation, community discussions, and case management.

Trademarks

For a list of RSA trademarks, go to www.emc.com/legal/emc-corporation-trademarks.htm#rsa.

License Agreement

This software and the associated documentation are proprietary and confidential to EMC, are furnished under license, and may be used and copied only in accordance with the terms of such license and with the inclusion of the copyright notice below. This software and the documentation, and any copies thereof, may not be provided or otherwise made available to any other person.

No title to or ownership of the software or documentation or any intellectual property rights thereto is hereby transferred. Any unauthorized use or reproduction of this software and the documentation may be subject to civil and/or criminal liability.

This software is subject to change without notice and should not be construed as a commitment by EMC.

Third-Party Licenses

This product may include software developed by parties other than RSA. The text of the license agreements applicable to third-party software in this product may be viewed on the product documentation page on RSA Link. By using this product, a user of this product agrees to be fully bound by terms of the license agreements.

Note on Encryption Technologies

This product may contain encryption technology. Many countries prohibit or restrict the use, import, or export of encryption technologies, and current use, import, and export regulations should be followed when using, importing or exporting this product.

Distribution

EMC believes the information in this publication is accurate as of its publication date. The information is subject to change without notice.

Contents

- NetWitness Log Parser Tool Overview 5**
 - Parser Structure 5
 - Obtaining a Log File 6
 - Understanding Events 7
 - How Logs are Parsed in NetWitness 9
- Getting Started with NetWitness Log Parser Tool 10**
 - Getting a Log File from NetWitness 11
 - Creating a Log Parser File 12
 - Selecting an Event from the Log File 12
 - Defining a Header 12
 - Header Order 12
 - Defining a Message 13
 - Message Order 14
 - Editing a Log Parser File 15
 - Validating the Precedence of Pattern Definitions 15
 - Viewing Parsed Events and Associated Definitions 15
 - Custom Table Map Files 16
 - GitHub Community Link 16
- NetWitness Log Parser Tool Workflow 18**
 - Installing the NetWitness Log Parser Tool 18
 - Setting Preferences 18
 - Creating a Log Parser 20
 - (For 1.1 version) Creating a Custom Parser 23
 - Editing an Existing Parser 23
 - Parser Version 24
 - Deploying Parsers on a Log Decoder 25
 - Opening a Log File 26
 - Auto Splitting Log Files 26
 - Generating Parsing Summary Report 27

Status Bar	29
Understanding the Log Data Section Workflow	29
Determining the Parser Definition Method	29
Example: Extract Generic Information	29
Defining the Header Pattern	30
Defining the Message Pattern	33
Defining a Throw-away Variable	34
Adding a Constant Function	34
Adding an Event Time Function	34
Event Time Function Formatting Characters	35
Adding a Custom Table Map	36
Using Delete for a header or message	37
Using Move Up or Move down	37
Using Duplicate for a header or message	37
Using Undo and Redo	38
Log Filter Functionality	39
Parser Header and Message Search Functionality	41
Header Search Functionality	41
Message Search Functionality	42
Advanced Search Options	44
Advanced Log Filter Options	44
Advanced Header Search	45
Advanced Message Search	46
TAGVALMAP Feature	47
Using the TAGVALMAP Feature	47
Setting Up a Header and Creating a Message	48
(For 1.1 version) VALUMAPS	50

NetWitness Log Parser Tool Overview

NetWitness Log Parser Tool (NwLPT) is a graphical tool that enables you to create and edit log parsers that run on the NetWitness Log Decoder. Using the NetWitness Log Parser Tool, you can define how a NetWitness Log Decoder identifies, parses, and extracts information from the events of a specific event source. These parser definitions are stored as an XML file, called a log parser XML file, which is deployed on the NetWitness platform.

You can create a new log parser for an event source that is not currently supported by NetWitness. You can also edit an existing log parser to add or edit definitions for events, or to correct errors. You may need to edit an log parser in one of the following situations:

- Upgrade to a new version of an event source that contains new, updated, or deprecated log messages.
- Include additional definitions in existing events.
- Update the definition for an existing event in a log parser.

Parser Structure

The NetWitness Log Parser Tool uses the device type of the log parser to create the structure for the parser. The NetWitness Log Parser Tool also appends the device type to the directory that you specify for your parser.

When you create a log parser, you select a device type for it. The device type must start with a letter. The RSA naming convention is to make the device type all lowercase and remove the spaces. For example, Cisco ASA would have the device type **ciscoasa**. It is not necessary to follow the RSA naming convention to use this tool.

In your log parser directory, the NetWitness Log Parser Tool creates two files with the correct name for NetWitness:

- **INI file.** This is the parser configuration file. (Example: ciscoasa.ini)
- **XML file.** This is the log parser XML file that contains the parser definitions. (Example: ciscoasamsg.xml). The device type is appended with **msg**.

Both of these files are required to deploy your parser in NetWitness.

When you finish creating or updating your parser, you have the option of retrieving the completed parser in four formats:

- **Individual Files: Parser (.XML) and Configuration (.INI)** (In the main menu, select **File > Save** or **Save As**). This option creates a device type folder containing an XML file and a configuration INI file. These individual files are viewable as the raw parser and configuration files, but they cannot be imported directly in the NetWitness Log Decoder.

- **Parser Package: .envision** (In the main menu, select **Actions > Export Parser**). This option creates an event source package that consists of the log parser XML and configuration INI file. This format is used to import the event source to a Log Decoder directly.
- **Live Resource: .zip** (In the main menu, select **Actions > Export Resource**). This option creates an event source package in a .zip format that consists of all the log parser XMLs and configuration INI files. The .zip format can be used to deploy parsers through RSA Live. It enables deployment of parsers to multiple Log Decoders simultaneously.
- **To deploy a parser on a Log Decoder** (In the main menu, select **Actions > Deploy Parser**). This option enables you to deploy the parser directly to the Log Decoder. It also supports deployment of parsers to multiple Log Decoders simultaneously. For more information, see [Deploying Parsers on a Log Decoder](#).

For more information, see the "Download Log Parsers from Live and Deploy from Local Network" topic in the *RSA Content and Resources* on how to upload the event source log parsers from your local network to the NetWitness Log Decoder.

For more information, see the "Resource Package Deployment Wizard" topic in the *Live Services Management Guide* for Version 11.1 for information on how to upload the event source log parsers from your local network to the NetWitness Log Decoder at the following location: <https://community.rsa.com/docs/DOC-79989>

For more information, see the "Enable and Disable Parsers and Log Parsers" topic in the *11.0 Decoder and Log Decoder Configuration Guide* for Version 11.1 at the following location: <https://community.rsa.com/docs/DOC-80190>.

Obtaining a Log File

To create a log parser that a NetWitness Log Decoder can use to identify, parse, and extract information from a specific event source, you must obtain a log file from the event source that you want to integrate with NetWitness. After you obtain the log file, you can use NetWitness Log Parser Tool to create a log parser.

Before getting started with NetWitness Log Parser Tool, you must know the log collection protocol that was configured when the event source was deployed with NetWitness.

If the log collection protocol that you configured when you set up the event source in NetWitness is Syslog, you can use a log file generated by the event source to create or edit a log parser.

If you configured any other log collection protocol, you must export a log file from NetWitness in text format.

RSA recommends that you compile a log file that contains all the unique events generated by the event source that you want to integrate with NetWitness. While compiling the log file, ensure that:

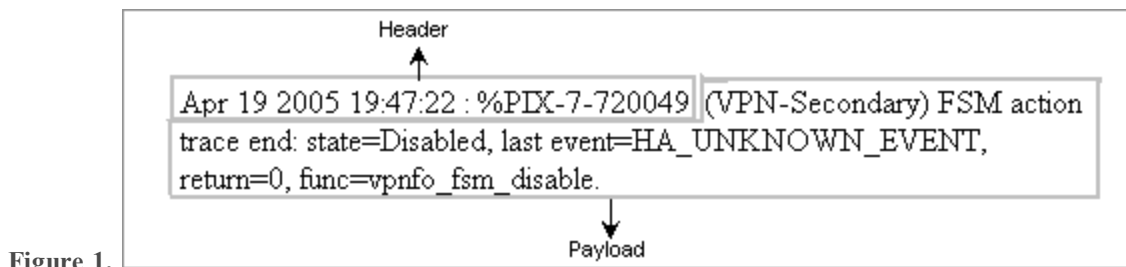
- All the events are from a single event source.
- Each event is listed in a single line, without any line breaks.
- The maximum size of any event is 64 KB.
- The log file contains one or two instances of each unique event.

Note: The recommended maximum size of the log file is 25 MB. A larger file can be used, but it will take more time to load and parse.

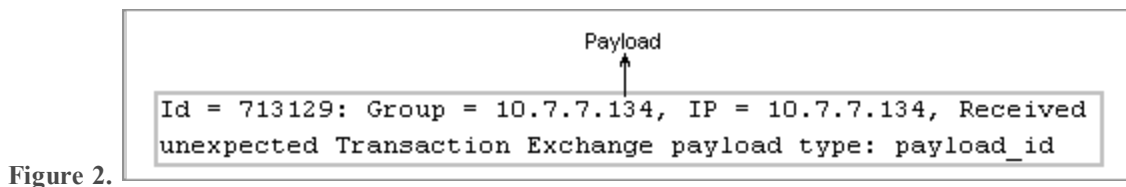
For events transmitted by Syslog, you can put the raw logs directly in the NetWitness Log Parser Tool. For all other event formats, you need to get the log data from NetWitness. To get log data from NetWitness, see [Understanding Events](#).

Understanding Events

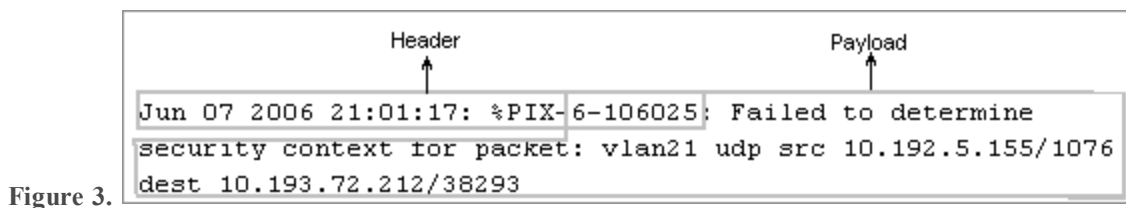
Typically an event consists of two main elements, a header and a payload. The following figure shows an example of an event with a header and a payload.



In some events, you may define the entire event as payload as shown in the following figure.



In some events, you may define the payload to begin from the header, and the header and payload may overlap.



Header

The header consists of the following elements, which are common across multiple events:

MessageID . Indicates a unique identifier for the message in the event. In the examples in the following figures, the MessageID is unique to the event.

Note: Make sure to define a single MessageID for each header.

Message ID
↑

Apr 19 2005 19:47:22 : %PIX-7-720049 (VPN-Secondary) FSM action trace end: state=Disabled, last event=HA_UNKNOWN_EVENT, return=0, func=vpnfo_fsm_disable.

Message ID
↑

Jan 01 11:06:39 [10.5.92.51] Id - 713129 Group = 10.7.7.134, IP = 10.7.7.134, Received unexpected Transaction Exchange payload type: payload_id

Caution: If you create a header that is too generic and can be used to identify a wide variety of logs, it could match logs that are currently parsed through other parsers.

. (Optional) Consists of the date and time when the event was generated by the event source. Some events may not contain an event source time stamp.

Event source time stamp
↑

Apr 19 2005 19:47:22 %PIX-7-720049: (VPN-Secondary) FSM action trace end: state=Disabled, last event=HA_UNKNOWN_EVENT, return=0, func=vpnfo_fsm_disable.

Header Variable. (Optional) Contains a value in the event header that varies across similar types of events. In the examples in the following figures, 4874 and 4921 are header variables that indicate the session ID in the events.

Header variable
↑

Feb 11 04:20:16 [10.10.1.1] Socks5[4874]: TCP Connection Request: Connect (172.30.21.43:37444 to 172.30.33.23:80) for user root

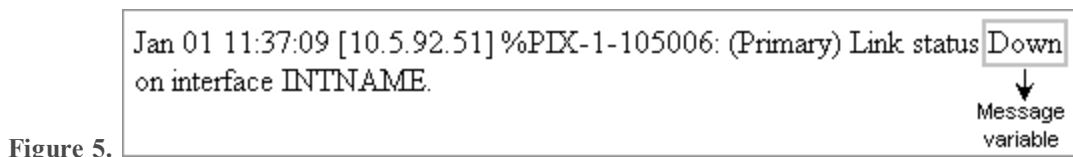
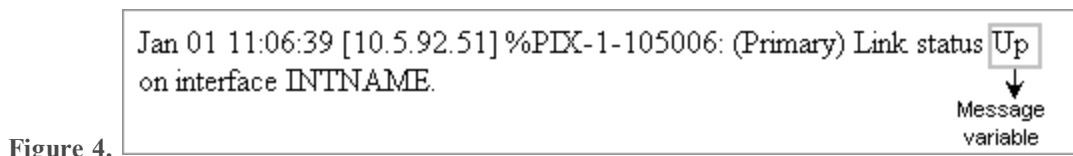
Header variable
↑

Feb 11 04:20:16 [10.10.1.1] Socks5[4921]: TCP Connection Request: Connect (172.30.21.43:37445 to 172.30.32.92:80) for user root

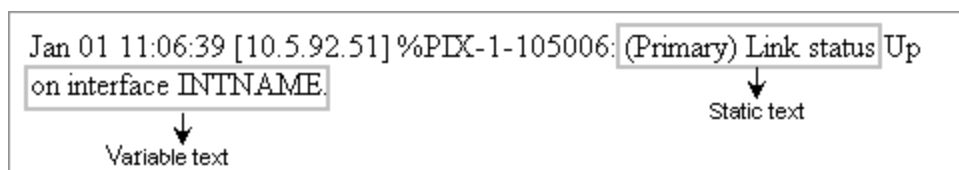
Payload

The payload is everything in the event that is not the header. It contains detailed information about the event. The payload is the message in the event. NetWitness uses this information for analysis and reporting. The payload consists of message variables and static text.

A message variable is a value in the payload that varies across similar types of events. In the examples in the following figures, Up and Down are message variables that indicate the link status of the INTNAME interface.

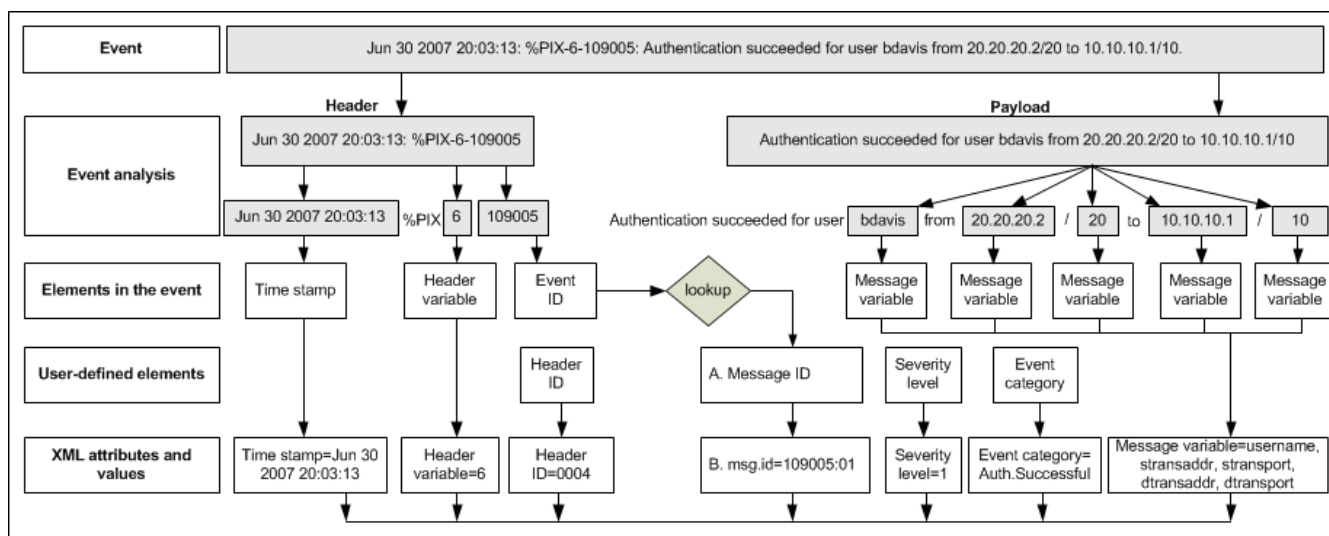


NetWitness Log Parser Tool classifies all the values in the payload that are not message variables as static text. The following figure shows an example of values that NetWitness Log Parser Tool classifies as static text.



How Logs are Parsed in NetWitness

The following figure shows an example of how you can create an XML definition that makes the event data available for analysis and reporting in NetWitness. It also shows the various elements in an XML definition.



Getting Started with NetWitness Log Parser Tool

The following table provides a high-level overview of the tasks that you can perform using NetWitness Log Parser Tool.

Goal	Task	Reference
Integrate an event source that is not supported by NetWitness.	1. Create a parser file that contains definitions for the events generated by the event source.	Creating a Log Parser File
	2. (Optional) View events that are parsed by a header or message definition.	Viewing Parsed Events and Associated Definitions
	3. (Optional) View the header and message definition that parse a selected event.	Viewing Parsed Events and Associated Definitions
	4. Create an event source package for deployment to a NetWitness Log Decoder.	Parser Structure
	5. In NetWitness, deploy the event source package.	Deploying Parsers on a Log Decoder See the "Add or Update Supported Event Source Log Parsers" topic in the <i>RSA Content and Resources</i> documentation. The "Download Log Parsers from Live and Deploy from Local Network" section provides information on how to upload the event source log parsers from your local network to the NetWitness Log Decoder.

Goal	Task	Reference
Upgrade an event source that is already supported by NetWitness.	1. Edit the existing log parser file.	Editing a Log Parser File
	2. (Optional) View events that are parsed by a header or message definition.	Viewing Parsed Events and Associated Definitions
	3. (Optional) View the header and message definition that parse a selected event.	Viewing Parsed Events and Associated Definitions
	4. In NetWitness, deploy the event source package.	Deploying Parsers on a Log Decoder See the "Add or Update Supported Event Source Log Parsers" topic in the <i>RSA Content and Resources</i> documentation. The "Download Log Parsers from Live and Deploy from Local Network" section provides information on how to upload the event source log parsers from your local network to the NetWitness Log Decoder.

Getting a Log File from NetWitness

1. In the **NetWitness** menu, go to **Investigate > Events**.
2. In the **Investigate** dialog, select a Log Decoder, Archiver, Concentrator, or Broker service and click **Events**. In the **Events** view, select the events and in the Actions menu, select **Export > Export All Logs**.
3. In the **Enter file name for extraction** dialog, enter a name for your log file and click **OK**.
4. In the **Export Log Format** dialog, select **Text** and click **Export**.
You will receive a Scheduled Job notice.
5. Check the Job Notifications tray to view the status of the log file. Click the **View** link to go to the Jobs panel in the Profile view to download the log file.

Creating a Log Parser File

Creating a log parser file involves creating a definition for each type of event in the log file generated by an event source. Creating an event definition involves the following tasks:

- [Selecting an Event from the Log File](#)
- [Defining a Header](#)
- [Defining a Message](#)

Selecting an Event from the Log File

Select an event from the log file to define the various elements of the header and message in the event.

Defining a Header

Define the header by assigning the values in the event to header elements. The purpose of defining a header is to identify the event source from which the event is generated. When you define a header with all its elements, the definition can parse similar types of events in the log file.

RSA recommends that you define a generic header definition that will parse multiple events that follow similar formats. The NetWitness Log Parser Tool generates a unique identifier, the HeaderID, for each header definition to identify the header definitions available in the log parser XML file. However, you can change the generated identifier to provide a unique HeaderID of your choice.



You can include the following elements when defining how to locate the MessageID in the header:

- MessageID, used in the Event ID lookup, which enables you to specify one of the following options:
 - MessageID variable
 - Variable suffix
 - Concatenation

Header Order

Header order determines the precedence of the headers. In general, headers should be ordered from specific to generic (see [Validating the Precedence of Pattern Definitions](#)).

A header's position can be changed by selecting the header and using **Move Up/Down** menu items on the **Edit** menu (or using the keyboard shortcuts for the up and down arrow keys

(**Ctrl+Up**  and **Ctrl+Down** ). You can also right click on a header or message to bring up a popup Context menu that has options to move up and down.

For more information on these elements, see [Understanding Events](#).

Defining a Message

Define the message by assigning the values in the payload to message variables and defining message elements. A single message definition may parse one or more similar events in your log file.

The following table lists the message elements that you must define.



Message Element	Description
MessageID	<p>Indicates the identifier by which NetWitness identifies the event uniquely. The MessageID can be defined in one of the following ways:</p> <ul style="list-style-type: none">• Same as the event ID.• A combination of the event ID defined in the header definition and a unique variant. For example, if the event ID is 109801, the MessageID can be defined as 109801:02.• A brief description that identifies the event. For example, in the following event, the event ID is 187698, and the MessageID can be defined as CableFailover. <pre>Jan 01 11:06:39 [10.5.92.51] %PIX-1-101001: (PRIORITY) Error reading failover cable status.</pre> <div>Note: Make sure to define a single MessageID for each header.</div>
Event category	<p>Indicates the category to which the event belongs, based on the NetWitness taxonomy.</p>

Message Element	Description
Functions	<p>(Optional) Define actions to be performed on variables in an event to generate user-defined values. The NetWitness Log Parser Tool supports the following functions:</p> <ul style="list-style-type: none"> • Assign Constant assigns user-defined values to variables. • Assign Message Variable assigns the value of a message variable to another variable. • Calculation performs a calculation on values and variables in the event. • Convert Domain converts domain names to IP addresses. • Event Time assigns the date and time information in the event to a message variable. • Remove Quotes removes quotes from a variable. • URL Part extracts parts of a URL string.

Message Order

All messages are displayed in order by message group. Messages with differing message group values cannot be re-ordered. However, messages with the same group can be re-ordered, as order determines the precedence within the group. In general, messages within a group should be ordered from specific to generic (see [Validating the Precedence of Pattern Definitions](#)).

When applicable, a message's position within its message group can be changed by selecting the message and using the **Move Up/Move Down** menu items on the **Edit** menu (or using the

keyboard shortcuts for the up and down arrow keys (**Ctrl+Up**  and **Ctrl+Down** ). You can also right-click to open a context menu with the **Move Up/Move Down** options.

When creating a new message or editing the group of an existing message, the message is (re)positioned based on the new message group value. If there are existing messages with that group value, the message is positioned at the end of the message group.

Editing a Log Parser File

Before you edit a log parser file, identify and analyze the events in the corresponding log file. You can edit a log parser file that was created by the NetWitness Log Parser Tool or any other source.

Editing a log parser file involves the same header and message definition tasks as creating a log parser file. However, you may not need to define the header pattern if the headers are all defined. For example, you may want to edit a parser to add new messages to Windows or other platforms. You may also want to adjust an existing parsed message. For example, you may need to change IP source to IP destination in a particular event.

Validating the Precedence of Pattern Definitions

It is important to consider the precedence of pattern definitions when defining headers and messages with the same event ID.

You must arrange the definitions in the log parser file in order from specific to generic so that events are parsed against the specific header or message definition.

For example, suppose that a log parser file contains the following message definitions:

- Message 10123 < A B C >, where A, B, and C are elements in the message
- Message 10124 < A B C D >, where A, B, C, and D are elements in the message

If the order of the message definitions is as shown, NetWitness parses an A B C event from the event source using the Message 10123 definition. NetWitness also parses an A B C D event from the event source using the Message 10123 definition. The A B C D event must be parsed against the Message 10124 definition, which is specific. Therefore, you must ensure that the specific definition, Message 10124, appears before the generic definition, Message 10123, in the log parser file, as follows:

- Message 10124 <A B C D>
- Message 10123 < A B C >

After validating the log parser for data pattern warnings, you must validate the precedence of the header and message definitions in the log parser file.

The NetWitness Log Parser Tool displays the errors that occur in the header and message definition order. For better analysis and reporting, you must resolve all the precedence errors.

Viewing Parsed Events and Associated Definitions

After defining the log parser, you can view the highlighted header and message definitions in the parsed logs within the NetWitness Log Parser Tool. You can also view the header and message definition for a selected event in the log file.

While defining the log parser, you can view parsed events and the associated header and message definitions. For example, if you have defined two header definitions and one message definition in the log parser file, you can view parsed events for these definitions, and then continue to define more header and message definitions depending on the parsed events already viewed.

Custom Table Map Files

If you changed the table map file (**table-map.xml**) or created a custom table map file (**table-map-custom.xml**), get the changed table mapping files from NetWitness. You can add these custom table map files to the NetWitness Log Parser Tool.

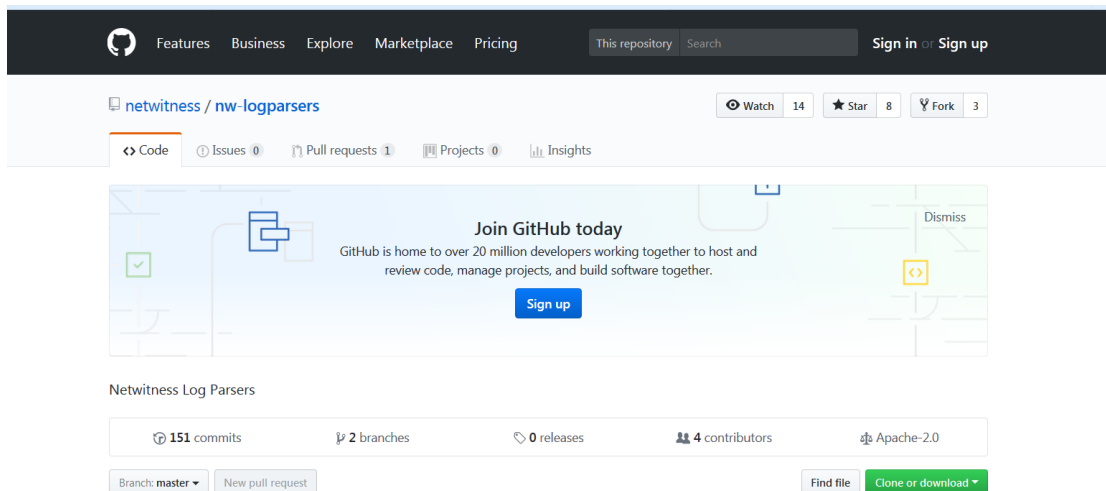
GitHub Community Link

The GitHub Community link provides access to a repository where you can share and contribute event source log parsers for the NetWitness Log Decoder.

To access the GitHub Community Link:

1. From the GitHub Welcome screen, go to **Help > Parser Community**. You can also use the keyboard shortcut **F3** to access the GitHub Community Link.

The GitHub Welcome screen is displayed.



GitHub members can contribute to the repository by adding or editing a log parser by raising a Pull Request that is reviewed by NetWitness Engineers. As a member of the GitHub community, you can create a new log parser for an event source that is not currently supported by NetWitness and share it with the NetWitness community. You can also edit an existing log parser to add or edit definitions for events, or to correct errors.

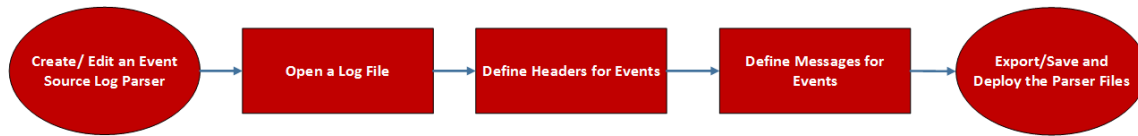
You may need to edit an log parser in one of the following situations:

- You upgrade to a new version of an event source that contains new, updated, or deprecated event messages.

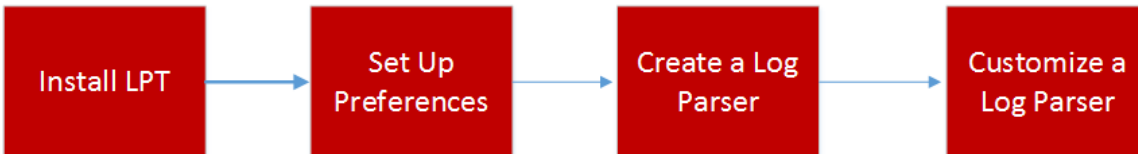
- You want to include additional definitions for existing events.
- You want to update the definition for an existing event in a log parser.

NetWitness Log Parser Tool Workflow

The following figure shows the workflow of the NetWitness Log Parser Tool 1.1 user interface.



This workflow shows the procedures to install LPT and create a log parser.



What do you want to do?

User Role	I want to ...	LPT Documentation
Administrator	Install LPT	Installing the NetWitness Log Parser Tool
Administrator	Set up Preferences	Setting Preferences
Content Expert	Create a Log Parser	Creating a Log Parser
Content Expert	Customize a Log Parser	(For 1.1 version) Creating a Custom Parser

Installing the NetWitness Log Parser Tool

You can download the Windows and MacOS versions of the NetWitness Log Parser Tool from the following location:

<https://community.rsa.com/docs/DOC-85202>

If you are using a Beta version of the Event Source Integrator Tool, you need to uninstall and download the new NetWitness Log Parser Tool installer from the following location:

<https://community.rsa.com/docs/DOC-85202>

Note: The Recent Parsers and Open Recent sections are empty, since this is a new installer.

Setting Preferences

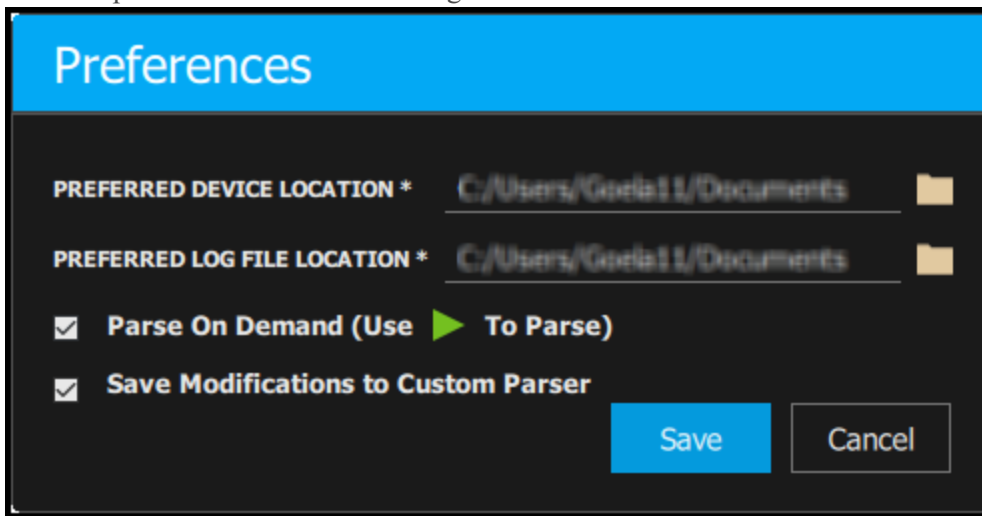
The Preferences dialog allows you to provide paths for logs and parsers, and to select a parsing mode for the application.

Note: The default path is your Documents folder and the default parsing mode is Auto.

To access the Preferences settings:


- For Windows systems: From the main menu, go to **File > Preferences**.
- For MacOS systems: From the NetWitness Log Parser Tool menu, select **Preferences**.

An example of the Preferences settings is shown below.



The default paths and modes are explained in the following table.

Field	Description
Preferred Device Location	<ul style="list-style-type: none">• Default directory that is opened when you open an existing parser.• The default directory is the location where a new parser is created.• Imported parsers are saved in this directory by default.
Preferred Log File Location	Directory that the NetWitness Log Parser Tool opens when you want to load a log file.

Field	Description
Parsing Mode	<p>By default, the NetWitness Log Parser Tool uses Continuous Parse mode, which means the log file is reparsed whenever there is a change to the parser.</p> <p>Note: You can change the mode to Parse On Demand if you notice that it is taking a while to parse the log file. A Play icon () is displayed in the middle divider that can be used to parse the log when a sufficient number of changes is complete. You can also use the F5 keyboard shortcut to facilitate log file parsing.</p>
(For 1.1 version) Save Modifications to Custom Parser	<p>When this check box is enabled, all the changes in the log parser file will be saved in custom parser file. This option will be ignored if the custom parser already exists and all the changes are saved in the custom file.</p>

Note: When a custom parser file is opened, the tool displays the merged entries of the log parser and the custom parser.

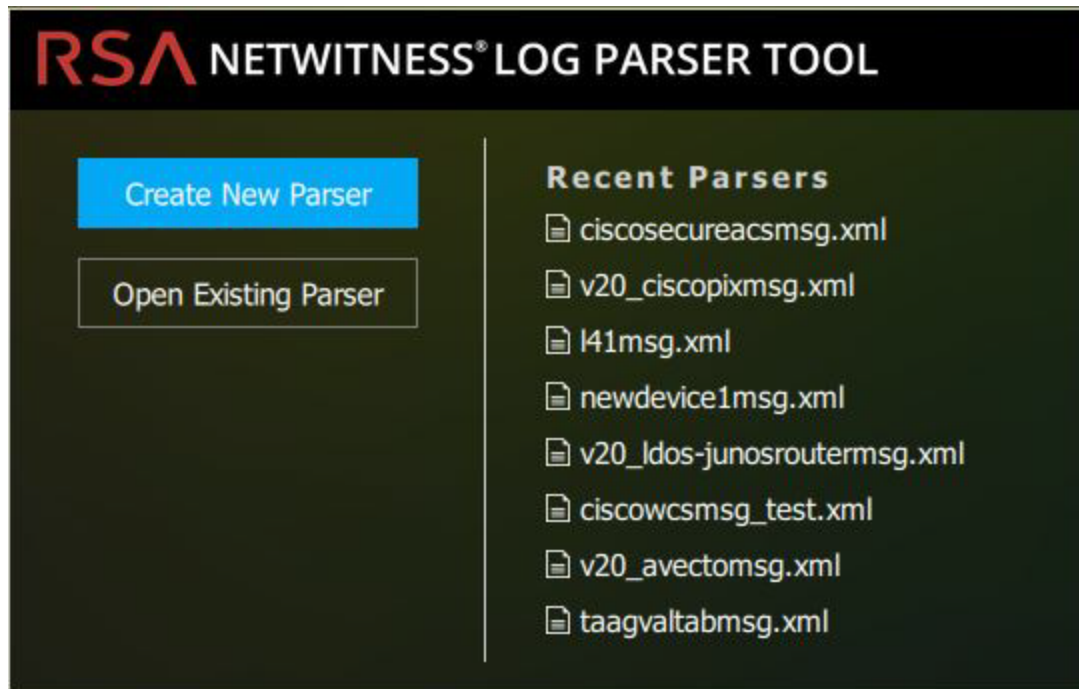
Caution: A user cannot open a custom parser file when log parser file is not available.

Creating a Log Parser

Note: All parsers opened in the tool are automatically saved at 30 seconds intervals as serves as a backup. You can view the last saved time in the Status Bar.

To create a new parser, follow these steps.

1. Select **Create New Parser**.



The **Create New Parser** dialog is displayed.

2. In the **Create New Parser** dialog:
 - a. Enter the **Device Type**. The **Device Type** must start with a letter. The RSA naming

convention is to make the device type all lowercase and remove the spaces. For example, Cisco ASA would have the device type **ciscoasa** and Actiance Advantage would be **actianceadvantage**. It is not necessary to follow the RSA naming convention to use this tool. The device type provides additional information about the event.

Note: Special characters are not allowed for the Device Type.

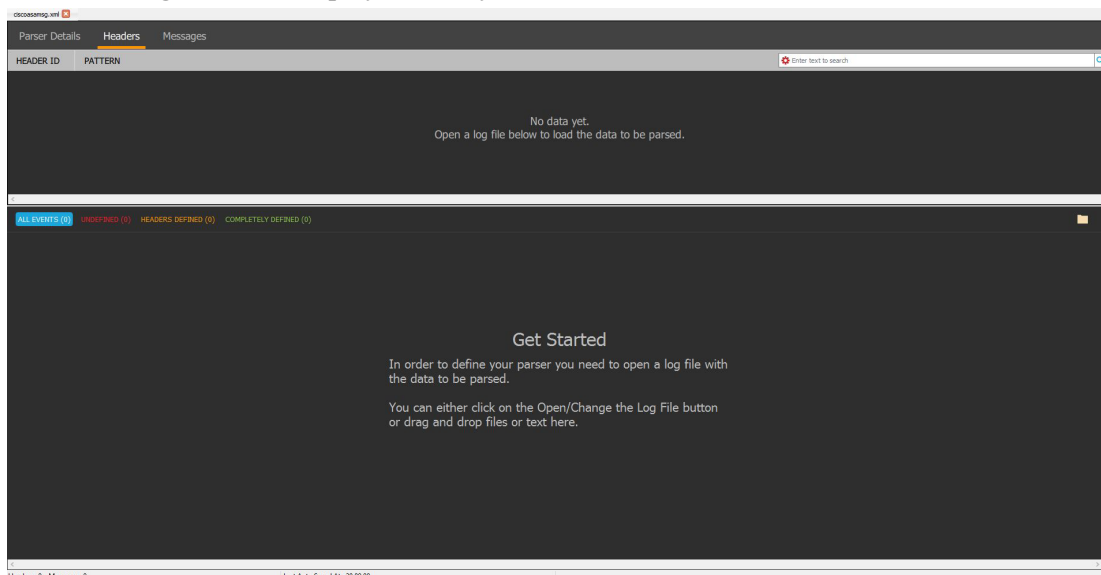
- b. Enter the **Device Display Name**. For example, Cisco ASA.
- c. Select a **Device Class** from the drop-down menu. For example, Firewall.
- d. In the **Device Location** field, specify the directory where you want to create the parser. In the directory that you specify, the NetWitness Log Parser Tool creates two files with the correct name for NetWitness:

INI file (Example: ciscoasa.ini)

XML file (Example: ciscoasamsg.xml) The device type is appended with **msg**.

Note: The parser path that is set in the **Preference** page is auto-populated here.

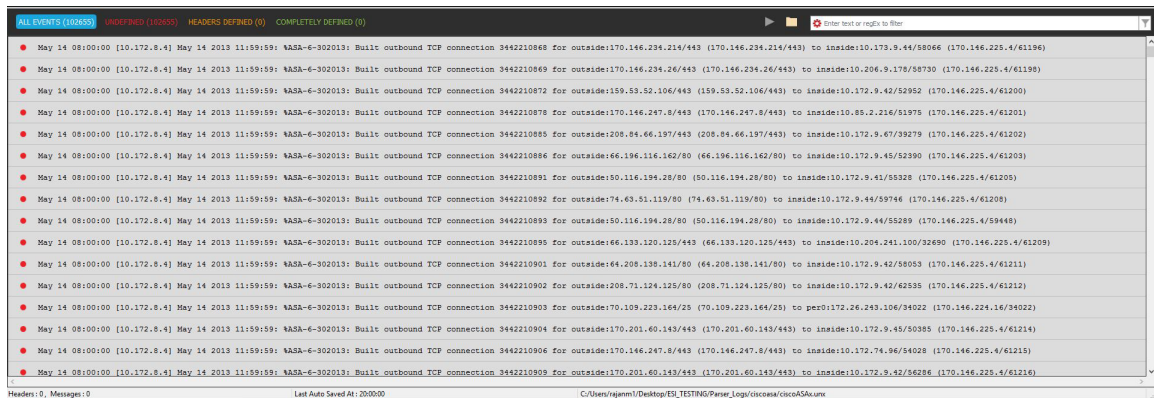
- e. In the **Log File** field, select a log file. This field is optional and can be selected at a later time.
3. Click **Create** to create a new parser, or click **Cancel** to return to the **Welcome** page. The following screen is displayed after you select **Create New Parser**.



4. Click the **Open/Change Log File** icon (📁). Browse to find the log file that you want to open and click **Open**.

For more information, see [Opening a Log File](#)

All the events in the selected log file are displayed, as shown in the following example.



(For 1.1 version) Creating a Custom Parser

Log Parsers can be customized by adding new parser elements or modifying existing ones. On customization, you can save it as a separate custom parser file, such that the base parser can be updated independently and customizations are applied on top of it.

Note: The custom parser is not deleted or overwritten during Log Decoder upgrades or RSA Live Content updates.

By default, log parser files of Log Decoder are located at
`/etc/netwitness/ng/envision/etc/devices`

The custom parser files are saved in the respective folder where the log parser files exists. For example, **ciscoasa** parser file will be saved in following folder.

`/etc/netwitness/ng/envision/etc/devices/ciscoasa`

The custom parser file is an XML file and should be saved with a device name followed by "-custom". For example **ciscoasams-g-custom.xml**

Note: Custom Parser is not supported in 11.0 version and is available in 10.6.5 or later and 11.1 and later versions.

Make sure you have enabled "Save Modifications to Custom Parser". For more information see *Preferences* section

For more information, see "Log Parser Customization" at <https://community.rsa.com/docs/DOC-83425>

Editing an Existing Parser

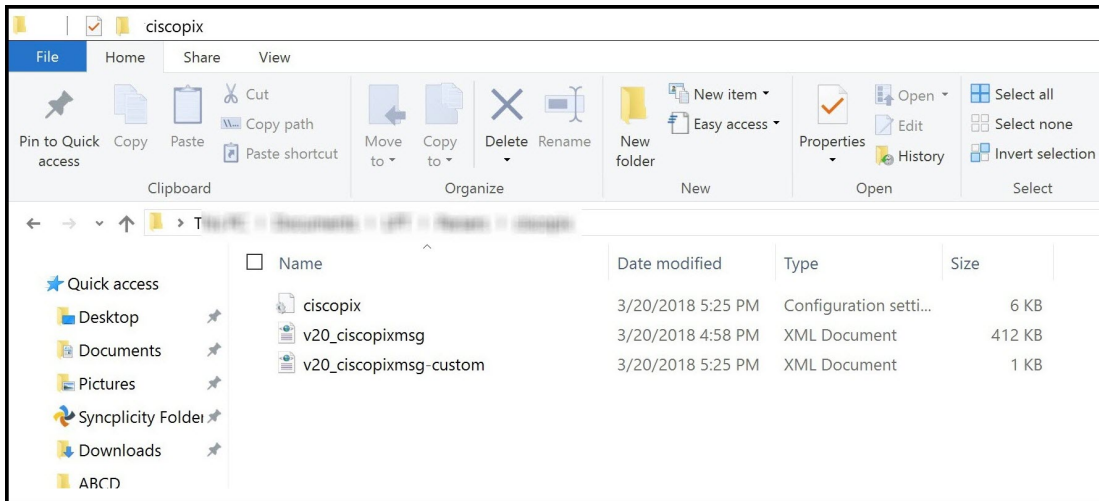
To edit an existing parser, follow these steps.

Note: If you opened your parser previously in the NetWitness Log Parser Tool, you can open the parser from the **Recent Parsers** section of the Welcome screen.

1. Select **Open Existing Parser** and click **Open**.

The directory which is set as the default device location under **Preferences** is opened when **Open Existing Parser** is selected.

The directory for NetWitness Log Parser Tool parsers is displayed.

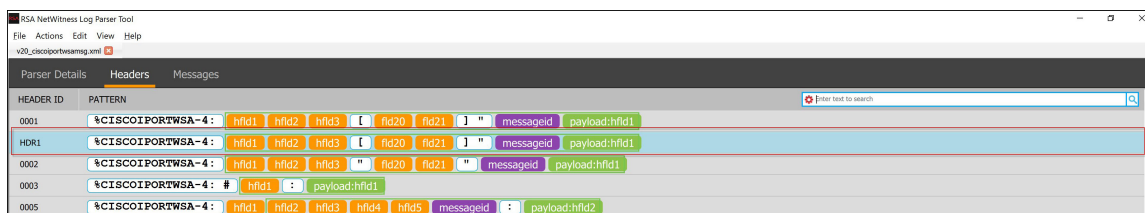


2. Type or select the name of the parser that you want to edit and click **Open**. For example, ciscoasamsg.xml.

3. Edit the parser entries such as headers, messages or tagval for customization.

4. Go to **File** and click **Save**.

Note: All the custom parsers entries are highlighted in blue color as displayed in the image



Parser Version

In the Parser Details section there is a Parser Version field. If there is a specific version associated with the parser, that version is displayed in the Parser Version field. You can also manually change the Parser Version in the Parser Version field.

Parser Details
Headers
Messages

DEVICE
TAGVAL
VALUEMAPS

DEVICE TYPE:
The name of the device. This name is fixed to the filename structure and cannot be changed.

DEVICE CLASS:
The type of device from which the log is extracted.

DISPLAY NAME:
The name that will be displayed for this device type.

DEVICE VERSION:
The version that will be displayed for this device type.

Deploying Parsers on a Log Decoder

Using the NetWitness Log Parser Tool, you can deploy parsers on a Log Decoder by following these steps:

1. Open the parser that you want to deploy on the Log Decoder.
2. Go to **Actions > Deploy Parser**.

The following pop-up dialog is displayed.

NwLPT

Deploy Parser

Deploy Parser in Log Decoder providing IP and access credential

LOG DECODER IP *

USERNAME *

PASSWORD *

Deploy
Cancel

3. Enter the IP address of the Log Decoder where you want to add the parser, along with the credentials of the Log Decoder.

- Click **Deploy** to add the parser to the Log Decoder, or click **Cancel** to return to the previous screen. After you click **Deploy**, the parser is added to the Log Decoder. And subsequently, all log parsers are reloaded.

Note: These entries are not retained, all the fields must be re-entered every time this dialog is opened.

Opening a Log File

Note: The first time you open the log file for a parser, the location that is set in **Preferences** is the default location.

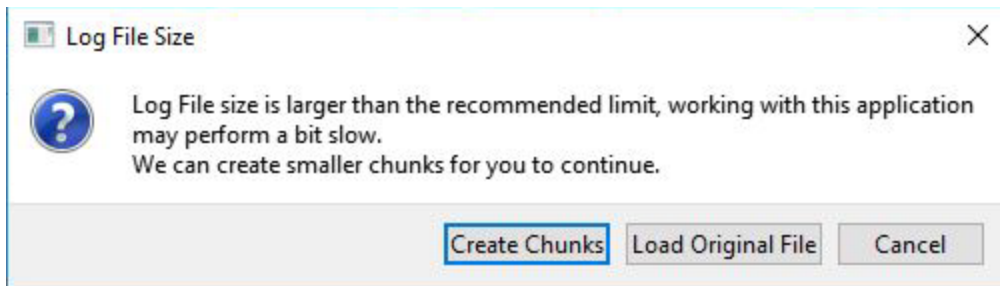
In the **Log Data** section, do one of the following to add a log file for parsing:

- Click **Open/Change Log File** icon.
- Drag and drop the log file.
- Drag and drop the text of a plain log.

Note: When a log file is open, you can change to a different log file using the **Open/Change Log File** icon.

Auto Splitting Log Files

When you upload a log file, the following dialog is displayed if your log file exceeds 25 MB.



The Log File Auto Splitting option allows you to split the size of a log file that exceeds 25 MB.

- If you select **Create Chunks**, the log file is split into smaller sized files without affecting the original log file.
- If you select **Load Original File**, the log file may load slowly, because it exceeds the recommended maximum size of 25 MB.
- If you select **Cancel**, you can select a different log file to load.

Note: If a file larger than 25 MB is used, it is recommended that you use On-Demand Parsing so that the parser does not attempt to parse a large log file after every change.

Generating Parsing Summary Report

You can generate a report on the parsing details of the parser and log file that you loaded. This report provides detailed information about the parsed messages and headers, as well as information about unused headers and messages and top 10 meta values. The report opens automatically in your default browser and is saved your Documents folder on your computer.

Note: Each time you generate a Parsing Summary Report, the existing report is overwritten. If you want to retain the existing report, it is recommended that you re-name the generated report.

To generate a report, follow these steps:

1. Open the parser associated with the report that you want to generate.
2. Open the log file that is associated with your selected parser. Note that if your log file is extremely large, a dialog is displayed that asks if you want the log file broken into smaller chunks.
3. Go to **Actions > Generate Parsing Summary**.

An example of the Generate Parsing Summary Report is shown below.

NWLPT Parsing Summary Report

Device Parser : C:/
Device Log File : C:/

Overall Parsing Summary

This section gives the summary of the overall parsing for the selected log file against the device parser

Total Log(s)	Parsed Log(s)	UnParsed Log(s)	Only Header Parsed Log(s)
20	20	0	0

Header Parsing Summary

This section gives the summary for the Header(s) in the device parser. Primarily it summarizes the Top 10 used HeaderId(s) and the list of HeaderId(s) that were unused for this selected log file

Header Utilization = 18.18%

#	HeaderId	Matching Log Count	Matching Log Percentage
1	0001	19	95.00
2	0004	1	5.00

Unused HeaderId(s)

0002, 0003, 0005, 0006, 0007, 0008, 0009, 0010, 0033

Message Parsing Summary

This section gives the summary for the Message(s) in the device parser. Primarily it summarizes the Top 10 used MessageId(s) and the list of MessageId(s) that were unused for this selected log file

Message Utilization = 1.73%

#	MessageId	Matching Log Count	Matching Log Percentage
1	104002:01	2	10.00
2	101001	1	5.00
3	101002	1	5.00
4	101003	1	5.00
5	101004	1	5.00
6	101005	1	5.00
7	102001	1	5.00
8	103001	1	5.00
9	103002	1	5.00
10	103003	1	5.00

Unused MessageId(s)

105001, 105002, 105003, 105004, 105005, 105006, 105007, 105008, 105009, 105010, 105011, 105020, 105021, 105031, 105032, 105034,

Metas Parsing Summary

This section gives the summary for the metas in the device parser. Primarily it summarizes the Top 10 used metas

#	Meta Name	Meta Count
1	context	20
2	event_description	20
3	level	20
4	messageid	20
5	result	5
6	day	1
7	month	1
8	resultcode	1
9	time	1
10	year	1

Status Bar

Within the NetWitness Log Parser Tool user interface, there is a status bar located at the bottom of the page that displays the following information:

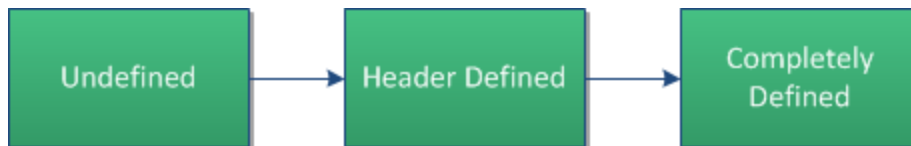
- Header and Message count
- Latest auto-save time
- Progress Bar that displays that Custom Parser is enabled or disabled.

The following example Status Bar shows the most recent time that a log file was auto-saved.



Understanding the Log Data Section Workflow

The following figure shows the workflow of the Log Data section.



After you have defined a new parser or opened a parser to edit, work from the Log Data section to define the headers and messages. Events move from **Undefined** (Header and Message not defined) to **Header Defined** (Message not defined) and then to **Completely Defined** (Header and Message defined).

Note: All defined Headers and Messages can be duplicated. This allows a simplified parser development where a similar pattern is needed for a Message or Header.

Determining the Parser Definition Method

There are two main methods to specify a parser definition depending upon the type of information that you need to collect in the logs:

- Identify events with a specific type and extract generic information.
- Extract as much detailed information as possible from an event.

Example: Extract Generic Information

The following example shows how to create a parser that extracts generic information from a Cisco ASA log. It uses the following event from the Cisco ASA log:

%ASA-1-101001: (PRIORITY) Failover cable OK.

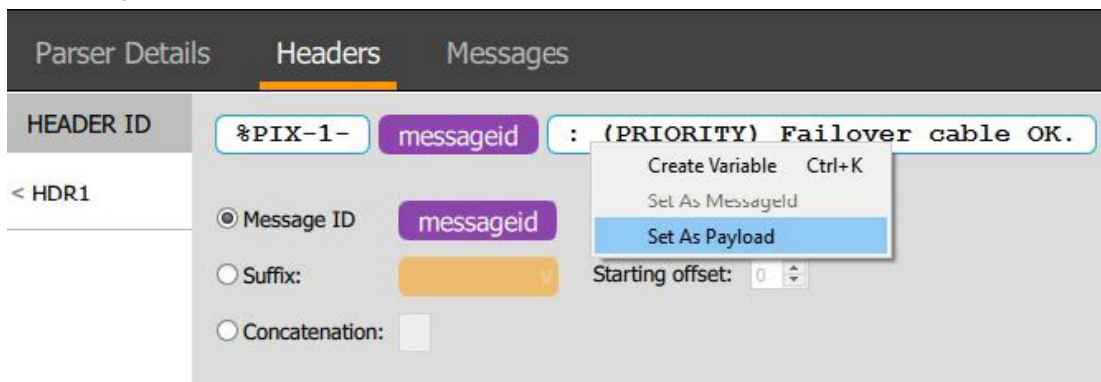
Defining the Header Pattern

To define the header pattern, follow these steps:

1. Select an undefined event.
2. Identify the text as the MessageID and highlight it.
3. Click **Create Header** to create the header.
4. If you want to change the log file, then click the **Open/Change Log File** icon.



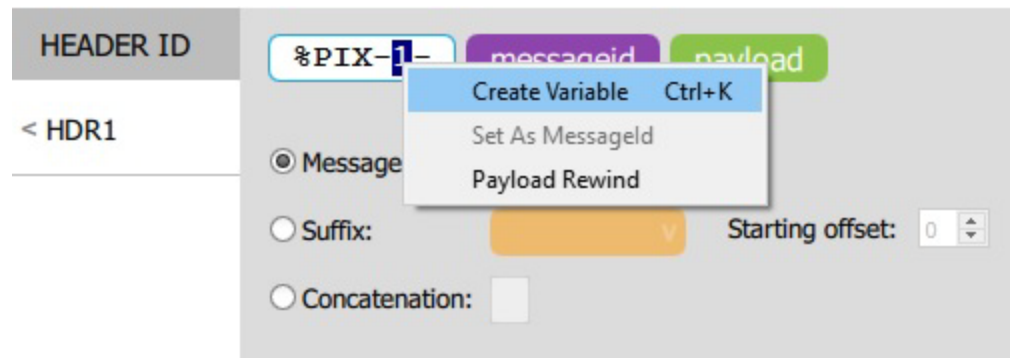
5. To start the payload, select the text to be marked as payload, right-click, then click **Set as Payload**. Alternatively, place your cursor at the start of the payload, right-click, and select **Set as Payload**.



6. Define a variable for anything that can change. To define a variable:
 - a. Highlight the text that you want to change to a variable.
 - b. Press **CTRL+K** (**COMMAND+K** for MacOS). You can also right-click to get a context menu that provides an option to create a variable. The background changes to orange,

which indicates a variable.

This example shows changing 1 to a variable.



- c. Start typing the name of the variable in the variable field, use the down and up arrow keys to select the variable, and press **ENTER** or you can also double-click the variable to select it.



7. To change where the payload starts, or to start at the header, right-click a variable and select **Payload Rewind**. For example, right-click the **Level** variable and select **Payload Rewind**.

The red box indicates a complete payload with a Message Header.

HEADER ID	Message ID	Suffix	Concatenation	Starting offset
& HDR1	messageid			0

8. To change the MessageID to a generic value:

- a. Select **Concatenation**, type a generic text string in the text field, and press **ENTER**.

HEADER ID	Message ID	Suffix	Concatenation	Starting offset
%PIX-level-id			generic-header	0

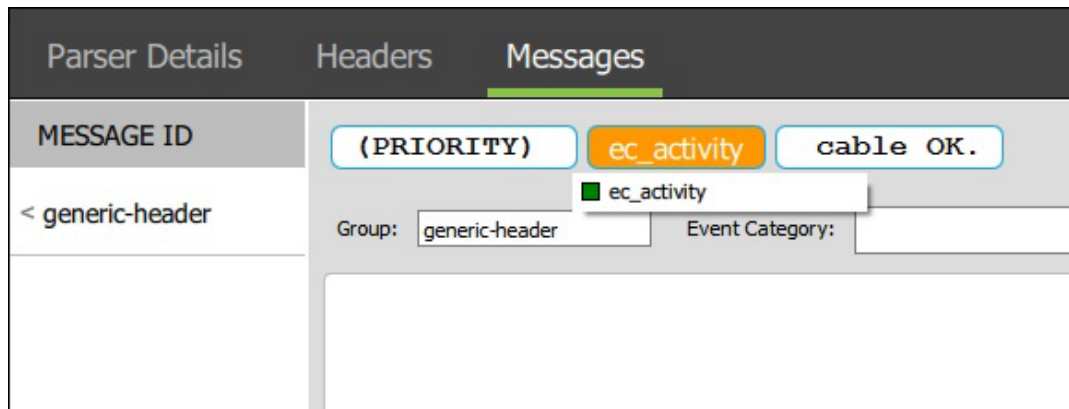
Note: Each Header needs a MessageID and a payload. Make sure to define a single MessageID for each header.

Note: The **Create Message** button in the log section is only enabled when the MessageID and Payload are defined.

Note: If you want to change your MessageID, you may need to delete the header and recreate it.

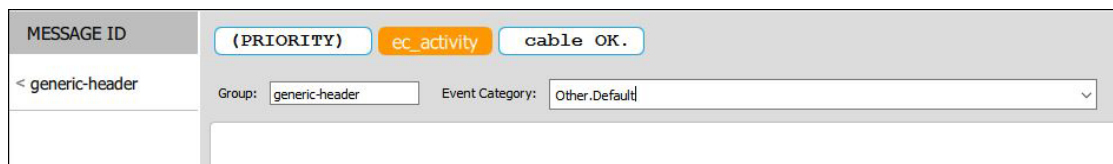
Defining the Message Pattern

1. After the MessageID and payload are defined in the event, the **Create Message** button is enabled.
Click on the **Create Message** button. This takes you to the **Messages** tab with the payload populated as the message.
2. In the message pattern, define variables for the values that you want to extract as meta. To define a variable:
 - a. Highlight the text that you want to change to a variable and press **CTRL+K** (**COMMAND+K** for MacOS). Or you can select the **Create Variable** option from the context menu. The background changes to orange, which indicates a variable.
 - b. Start typing the name of variable in the variable field, use the down and up arrow keys to select the variable, and press **ENTER**.



Note: The variables that you define create meta in the Log Decoder.

3. In the **Event Category** field, select a generic category for the message. For example, Other.Default.



The **Group** field populates from the header. The event shows as completely defined in the **Select an Event** section.

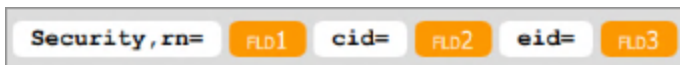
4. To save your changes, select **File > Save** or **File > Save As**, or press **Ctrl+S** (**COMMAND+S** for MacOS).
5. After you complete and save your changes, retrieve the completed parser.
You have a choice of three formats:

- **.envision** (In the menu, select **Actions > Export Parser**) This option creates an event source package that consists of the event source XML and configuration (INI) file.
 - **.zip** (In the menu, select **Actions > Export Resource**) This option creates an event source package in a .zip format that consists all the event source XMLs and configuration INI files.
 - Deploy the parser directly deployed from the Log Parser Tool to the Log Decoder.
From the main menu, select **Actions > Deploy Parser**.
6. Deploy the event source package in the NetWitness platform to integrate the event source. RSA recommends that you first deploy the parser to a test system to verify that it parses log traffic correctly.

Defining a Throw-away Variable

To add a throw-away field variable for information that you do not care about, create a variable and give it a name that is not defined in the variable list, such as **fld1**, **fld2**, or **fld3**.

1. Highlight the text that you want to change to a variable and press **CTRL+K** (**COMMAND+K** for Mac). You can also right-click to get a context menu that provides an option to create a variable. The background changes to orange, which indicates a variable.
2. Type the name of the throw-away field. For example, **fld1**
Since throw-away fields are not in the mapping, the information contained will be removed when parsing.



Adding a Constant Function

1. Right-click the box below the **Group** field and select **Add Function > Assign Constant**.
2. In the **value** field, type the value that you want to assign to the variable.
3. In the **Set Variable** field, type the name of a variable that was not previously selected.



Adding an Event Time Function

Use the Event Time function to change the format of an event source timestamp.

1. Right-click the box below the **Group** field and select **Add Function > Event Time**.
2. In the **Set Value** field, type the format that you want to assign to the time variable. For example, **%B %F %W %N:%U:%O**, which appears in the format **Jan 27 2015 23:55:29**.
3. In the **from** field, select whether to parse event time in this format from the message (MSG) or from the header (HDR).
4. Right-click the **Select Variable** fields and select a variable from the list. To add additional variables as required, right-click a variable and select **Add**.

Event time example:

Parse event time as	%B %F %W %N:%U:%O	from	HDR	EVENT_TIME_STRING	and assign to	EVENT_TIME
---------------------	--------------------------	------	-----	--------------------------	---------------	-------------------

Event Time Function Formatting Characters

The following table shows the format characters that Log Decoder supports for the Event time function.

Format Character	Description
%C	Dates of this format: 04/20/05 14:01:57
%R	Full Month Name, fixed width field: January, February, March, April, May, June, July, August, September, October, November, December
%B	Abbreviated Month Name, fixed width field: Jan, Feb, Mar, Apr, May, Jun, Jul, Aug, Sep, Oct, Nov, Dec
%M	Numeric Month, fixed width field: 01, 02, 03, 04, 05, 06, 07, 08, 09, 10, 11, 12
%G	Numeric Month Variable width field: 01, 02, 03, 04, 05, 06, 07, 08, 09, 10, 11, 12
%D	Numeric Month Day, fixed width field: 01, 02, 03, 04, 05, 06, 07, 08, 09, 10, , 23, 24, 25, 26, 27, 28, 29, 31
%F	Numeric Month Day Variable width field: 01, 02, 03, 04, 05, 06, 07, 08, 09, 10, , 23, 24, 25, 26, 27, 28, 29, 31
%H	Hour, fixed width field: 00-23
%I	Hour, fixed width field: 00-12
%N	Hour: Variable width field: 00-12 , 00-24

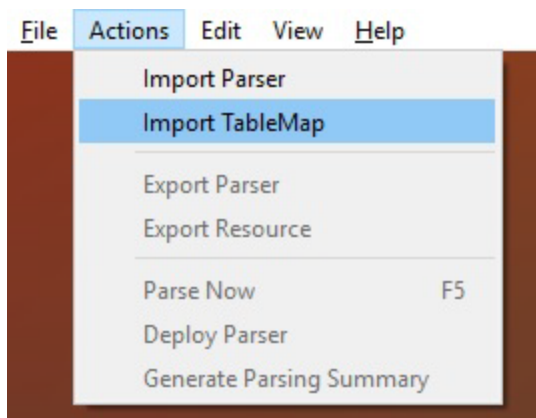
%T	Minute, fixed width field: 00-59
%U	Minute: Variable width field: 00-59
%J	Julian day, fixed width field: 001-365
%P	Alpha, fixed width field: AM or PM
%Q	A.M./P.M.
%S	Seconds, fixed width field: 00-59
%O	Variable width field: Seconds: 00-59
%Y	Year: 00-99
%W	Year, fixed width field: 0000-9999
%Z	Hours:Min:Sec
%A	Days
%X	Unix Time-Stamp (for example: 1424849941)

Adding a Custom Table Map

Note: If you changed the table mapping file (**table-map.xml**), or created a custom table mapping file (**table-map-custom.xml**), you can upload this updated file to the NetWitness Log Parser Tool.

To create a custom table map:

From the Welcome screen, select **Actions > Import TableMap** and choose the **table-map** that you want to upload.



Note: Before your file is overwritten, a confirmation message is displayed in the dialog box if you have already created a custom table map with the same name. This message does not display if you are uploading a **table-map-custom.xml** file for the first time.

Using Delete for a header or message

The Delete option allows you to delete a single header or message.

- To delete a header or message, press **Ctrl+Del** for Windows and **Fn+Ctrl+Del** for Mac OS or from the main menu, select **Edit>Delete**.
- Select a log you want to delete, right click and select **Delete**

Using Move Up or Move down

The Move up and Move down options allow you to move a header or a message up and down as per requirement. Using a Move Up option allows a message or a header to jump a position. Using a Move Down option allows a message or a header to step down.

- To Move Up a header or a message, press **Ctrl+Up** for Windows and **Fn+Ctrl+Up** for Mac OS or from the main menu, select **Edit>Move Up**.
- To Move Down a header or a message, press **Ctrl+Down** for Windows and **Fn+Ctrl+Down** for Mac OS or from the main menu, select **Edit>Move Down**.
- Select a message definition or header definition, you want to move up or move down, right click and select **Move Up** or **Move Down**

Using Duplicate for a header or message

The Duplicate options allows you to create a new message or header with identical information to the original record.

- To duplicate a header or message, press **Ctrl+D** for Windows and **Fn+Ctrl+D** for Mac OS or from the main menu, select **Edit>Duplicate**.
- Select a log you want to duplicate, right click and select **Duplicate**

Using Undo and Redo

The Undo and Redo options allow you to undo and redo any number of commands while the parser is being built or updated. Using the Undo option reverts to the last change that you made to a single message or header. Using the Redo option allows you to make changes to a single message or header.

- To Undo your changes, press **CTRL + Z**, or from the main menu, select **Edit > Undo**.
- To Redo your changes, press **CTRL + Y**, or from the main menu, select **Edit > Redo**.

Log Filter Functionality

The following example shows the filter options that are available from the Log Filter drop-down menu.



Log filter options are described in the following table.

Log Filter Option	Description
Logs	Searches for the selected value for log search. Searches for any log that contains the selected search text. For example, you can search for Bangalore . Search results include any logs that contain Bangalore as part of a log.
Meta	Searches for any meta in the logs section. For example, you can search for the term username . Search results include logs that contain username as part of a log.
Regex	Searches for the selected regex pattern in the logs section. For example, you can search for <code>\d+\.\d+.*</code> and the search results include logs that contain IP addresses as part of the logs.
Header ID	Searches for the selected HeaderID. For example, you can search for a HeaderID that is listed as 0008 . The search results include the number of headers that contain 008* defined as part of the header.

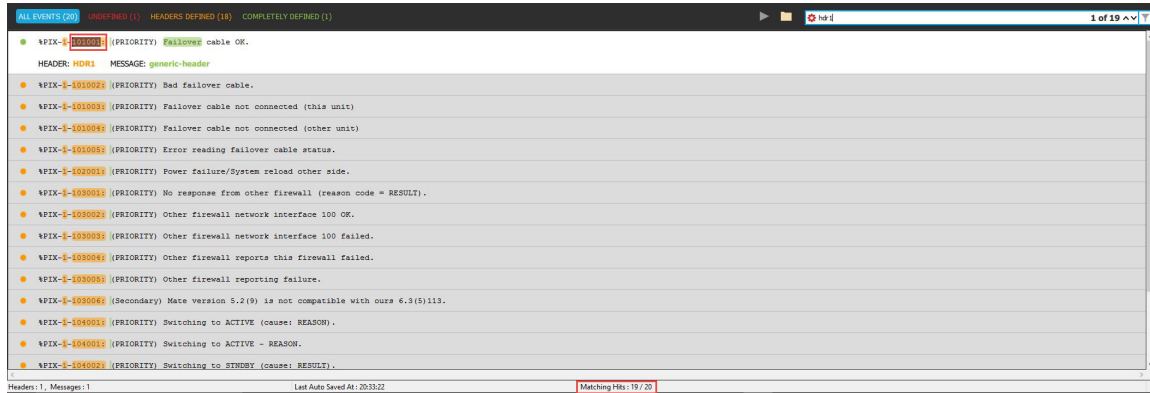
Message ID

Searches for the selected MessageID. For example, you can search for a MessageID such as **04_TACACSAcc**. The search results include the number of messages that contain **04_TACACSAcc*** defined as part of the message.

All

Searches for any logs, HeaderIDs, MessageIDs, variables, meta, or regex that matches the selected search text. For example, you can search for the text **syslog**, and the search results include any logs, HeaderIDs, MessageIDs, variables, meta, or regex that matches **syslog**.

The following example shows the search results for logs. The Search field and the Status Bar Message Count fields are highlighted.



Parser Header and Message Search Functionality

Within the Parser section, you can search on a HeaderID or Message Group by entering the value in the search field.

Header Search Functionality

The following example shows the search options that are available from the Header Search drop-down menu.

The image shows a user interface for searching. At the top is a search bar with a gear icon on the left and a magnifying glass icon on the right. The text "Enter text to search" is inside the bar. Below the search bar, a dropdown menu is open, showing the text "Search For:" followed by four options: "All", "Literal (Default)" (which is selected with a checkmark), "Variable", and "Header ID".

Header Search options are described in the following table.

Header Search Option	Description
Literal	Default search option that matches literals. This option searches for any literal within the Header section. For example, you can search for Bangalore with this option selected. The search results include the number of headers that contain Bangalore as part of a literal.
Variable	Searches for any variable or meta defined in the Header section. For example, you can search for saddr with this option selected. The search results include the number of headers that contain saddr defined as part of the header.
HeaderID	Searches for the selected HeaderID. For example, you can search for a HeaderID that is listed as 0008 . The search results include the number of headers that contain 008* defined as part of the header.
All	Searches for any selected text. For example, you can search for the text syslog , and a search is performed on any literal, HeaderID, or variable that matches the selected search text.

The following example shows the header search results.

HEADER ID	PATTERN
0004	months day year time : &PFX level messageid payload
0005	months day year time paddr : &PWISM level messageid payload
0006	months day year time : &PWISM level messageid payload
0007	paddr : &PFX level messageid payload
> 0008	paddr : &PWISM level messageid payload

Message Search Functionality

The following example shows the message search drop-down menu with the Literal option selected.

Message Search options are described in the following table.

Message Search Option	Description
Literal	Default search option that matches literals. Default selected value for Message searches. This option searches for any literal within the Message section. For example, you can search for Bangalore with this option selected. The search results include the number of messages that contain Bangalore as part of a literal.
Variable	Searches for any variable or meta defined in the Message section. For example, you can search for saddr with this option selected. The search results include the number of messages that contain saddr defined as part of the message.

Category	Searches for a particular event category that is part of the selected message. the search results include the number of messages that contain Auth.Successful.* defined as part of the selected message.
MessageID	Searches for the selected MessageID. For example, you can search for a MessageID such as 04_TACACSAcc . The search results include the number of messages that contain 04_TACACSAcc* defined as part of the message.
Message Group	Searches for the selected Message Group. For example, you can search for the text syslog . The search results include any literal, MessageID, variable, category, or Message Group that contain syslog .
All	Searches for any selected text. For example, you can search for the text syslog , and a search is performed on any literal, MessageID, variable, category, or Message Group that matches the selected search text.

The following example shows message search results.

Parser Details	Headers	Messages
MESSAGE ID	MESSAGE GROUP	PATTERN
> 101001	101001	<input type="text" value="context"/> <input type="text" value="event_description"/>
101002	101002	<input type="text" value="context"/> <input type="text" value="event_description"/>
101003	101003	<input type="text" value="context"/> <input type="text" value="event_description"/>
101004	101004	<input type="text" value="context"/> <input type="text" value="event_description"/>
101005	101005	<input type="text" value="context"/> <input type="text" value="event_description"/>

Advanced Search Options

Note: To perform an advanced search, use **&&**, which is an **AND** condition that helps drill down to your exact search pattern. This applies to all three advanced header and message search options.

Note: When you are performing a combination search, it is an **OR** condition for the search. The search begins after you press **ENTER**.

Note: The context menu for Headers and Messages has a new option called **Parsed Logs** that displays all the logs parsing from the selected Header or Message.

Note: All search options listed below can be used individually, as well as with other search options. The advanced search options can be used also be used in the logs section.

Advanced Log Filter Options

Note: When using the Advanced Search, you must select **All** from the Log Filter drop-down menu.

The following table shows the advanced log filter options.

Message Search Option	Description
Header Search	If you need to filter logs that parse with a specific header, use the @hid option, followed by the HeaderID (for example, @hid:0001).
Message Search	If you need to filter logs that parse with a specific header, use the @hid option, followed by the HeaderID (for example, @hid:0001).
Variable Name Search	If you want to filter logs that contain a specific variable or meta item, use the @<Variable-name> @saddr option. This option lists all logs that contain saddr meta keys.
Variable Value Search	If you want to filter logs that have a specific variable value, use the @<variable-name>:<variable-value> option (for example, @dport:10). This option lists all logs with a value of meta key dport as 10 .

Message Search Option	Description
Regex Search	If you want to search the logs using a regex, use the @regex option (for example, @regex:\d+\.\d+.*). This option displays all logs that contain an IP address.
Free Text Search	If you use this option, any text provided will be searched in the logs. If two words are provided, both of them are searched separately (similar to the way that Google performs text searches). As an example, if you enter the words rsa bangalore , rsa and bangalore are searched separately and all logs containing either word are displayed.

Note: If you want to search on terms as though they are in a sentence, you need to enclose your search query in quotation marks (for example, "**rsa bangalore**").

If the search query is not properly enclosed within quotation marks, the error is not automatically corrected, and no error message is displayed.

Note that when you search on a header, message, variable name or variable value, by default all searches only perform a contains match. For example, if you enter the search text **@hid:0001**, all headers containing **id 0001**, **0001:01**, **0001:02**, and so on will be displayed. If you want an exact match, the query should be entered as **@hid:"0001"** so that only logs that match **header-id 0001** are displayed.

Advanced Header Search

The following table shows the advanced header search options.

Advanced Header Search Option	Description
@id:	Searches for the selected HeaderID. Syntax: @id:<header>id> , where header_id is the HeaderID that you are searching for. For example, @id:0008 displays the number of headers that contain 0008* defined as headers that you can search.

Advanced Header Search Option	Description
@var:	<p>Searches for any variable or meta defined in the header section.</p> <p>Syntax: @var:<variable>, where variable is the variable or meta to search.</p> <p>For example: @var:saddr displays the number of headers that contain saddr defined as a part of the header that you can search.</p>
@literal:	<p>Searches for any literal content in the header section.</p> <p>Syntax: @literal:<text>, where text is the literal to search.</p> <p>For example, Bangalore displays the number of headers that contain Bangalore as part of the literal that you can search.</p>

Advanced Message Search

Advanced message search options are explained in the following table.

Advanced Message Search Option	Description
@id:	<p>Searches for the selected MessageID.</p> <p>Syntax: @id:<message_id>, where message_id is the MessageID that you are searching for.</p> <p>For example, @id:04_TACACSAcc displays the number of messages that contain 04_TACACSAcc* defined as messages that you can search.</p>
@var:	<p>Searches for any variable or meta defined in the header section.</p> <p>Syntax: @var:<variable>, where variable is the variable or meta to search.</p> <p>For example: @var:saddr displays the number of headers that contain saddr defined as a part of the header that you can search.</p>
@literal:	<p>Searches for any literal content in the message section.</p> <p>Syntax: @literal:<text>, where text is the literal to search.</p> <p>For example, Bangalore displays the number of messages that contain Bangalore as part of the literal that you can search.</p>

TAGVALMAP Feature

The <TAGVALMAP> feature is an advanced feature that enables easy parsing of event logs using the <TAGVAL> format. The parsing of <TAGVAL> logs is different from other logs, as the Log Decoder allows listing all **Tag=value** in a single MessageID where the tags can display in any order in the log.

Using the TAGVALMAP Feature

Using the <TAGVALMAP> feature saves you from writing different messages for the same MessageID if your log messages only differ in the order of variables within the payload, or if some parameters do not show up in the payload.

When you define your delimiters, the parser does not work the same way by matching the payload character by character, but the payload string is split around the delimiters into ><key-value> pairs. For every <key-value> pair, the static text is matched against the appropriate key and the value goes to the corresponding parameter in the table.

There are types of event logs where the part logs have a pattern of <tag-value> (<name-value>), which means the the tag parts remain the same, but the value is changed.

The Entry Separator has a maximum limit of three characters, while Entry Escape and Name Value Escape allow only one character.

The following example shows a TAGVAL parameter in the Parser Details section.

The screenshot shows the 'Parser Details' tab of the NetWitness Log Parser Tool. The 'TAGVAL' sub-tab is selected. The main configuration area displays the example payload 'name1=value1^name2=value2'. Below this, there are five configuration fields with their respective values:

Configuration Field	Value
Entry Separator:	^
Name/Value Delimiter:	=
Value Encapsulator:	
Entry Escape:	
Name/Value Escape:	

After you save the data, the corresponding XML file content looks similar to the following example:

```
<TAGVALMAP
```

```

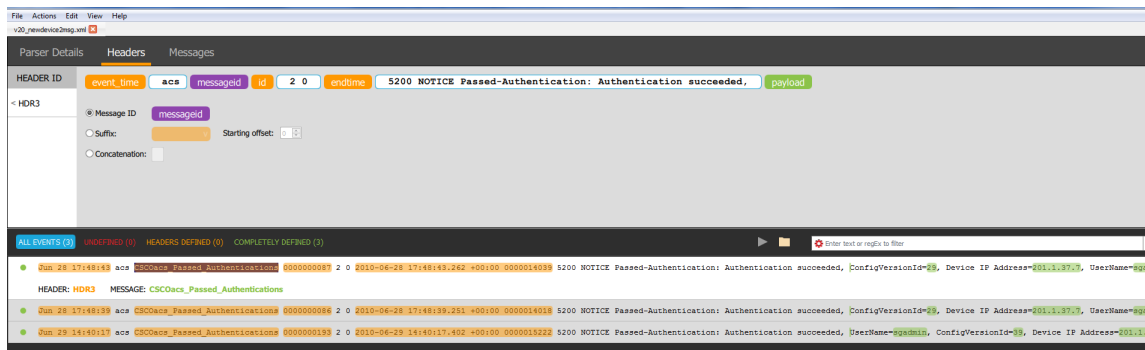
pairedlimiter=", "
encapsulator="'"
valuedelimiter="="
escapeValueDelim=" \"
escapePairDelim=" \">

```

Note: Press **ENTER** after you change each delimiter.

Setting Up a Header and Creating a Message

The following example shows a header and message that is created.



The corresponding Header XML definition is shown below:

```

<HEADER
  id1="HDR3"
  id2="HDR3"
  content="<event_time> acs<messageid><id> 2 0
  <endtime> 5200 NOTICE Passed-Authentication: Authentication
  succeeded, <!payload>"/>

```

Create Message

When you create a message, the **Name Value Pair** check box should be checked in order for the messages to be parsed. Note that if you do not select the **Name Value Pair** check box, the parser does not parse **<TAGVAL>** messages with a different order. The **Name Value Pair** is disabled by default and it is enabled for user input only if the message definitions satisfy the **<TAGVAL>** format, as shown in the following examples.

The **TAGVAL** format is either:

```

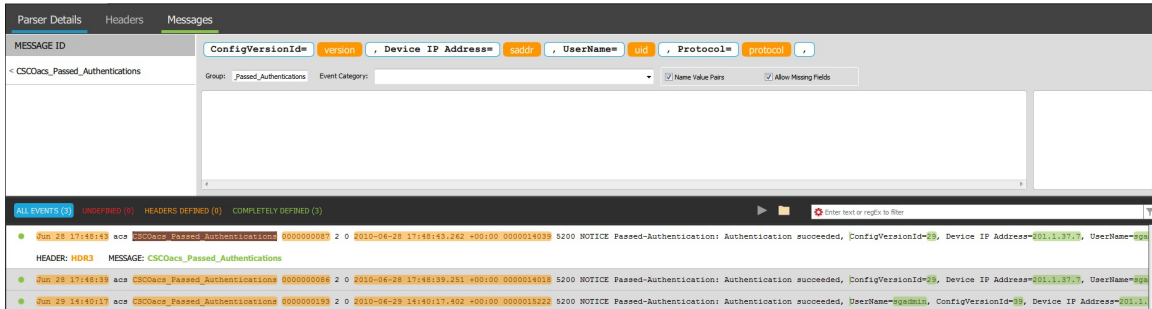
<literal><valuedelimiter><variable><pairedlimiter>...<literal><valuedeli
  miter><variable> format

```

Or

<literal><valuedelimiter><variable><paireddelimiter>...<literal><valuedelimiter><variable><paireddelimiter> format

The **Allow Missing Fields** option is used to parse event logs that missed some <TAGVALUE> pairs defined in the message definition. This means not all <TAGVALUE> pairs defined in the message definition need to be parsed.



The corresponding Message XML definition is shown below:

```
<MESSAGE
id1="CSCOacs_Passed_Authentications"
id2="CSCOacs_Passed_Authentications"
tagval="true"
missField="true"
content="ConfigVersionId=&lt;version&gt;; Device IP
Address=&lt;saddr&gt;; UserName=&lt;uid&gt;;
Protocol=&lt;protocol&gt;;"/>
```

Listed below are sample payload parts of event logs that are parsed:

ConfigVersionId=29, Device IP Address=201.1.37.7, UserName=sgadmin,
Protocol=Radius,

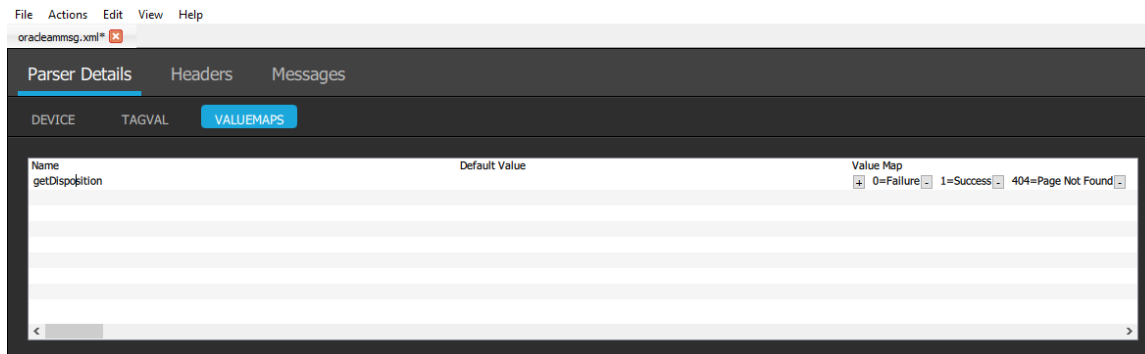
ConfigVersionId=29, Device IP Address=201.1.37.7, UserName=sgadmin,

Authentication succeeded, UserName=sgadmin, ConfigVersionId=39, Device
IP Address=201.1.37.7, Protocol=Radius

(For 1.1 version) VALUMAPS

VALUEMAPS functionality allows mapping of a value that is parsed in a meta to another corresponding meta.

For example, if you have a new event saved with a code “404” and a relevant information defined is “page not found error.” With VALUMAPS, you can map 404 to "page not found error" so that whenever 404 is seen, it also shows another mapped meta key.





VALUEMAPS consists of three columns.

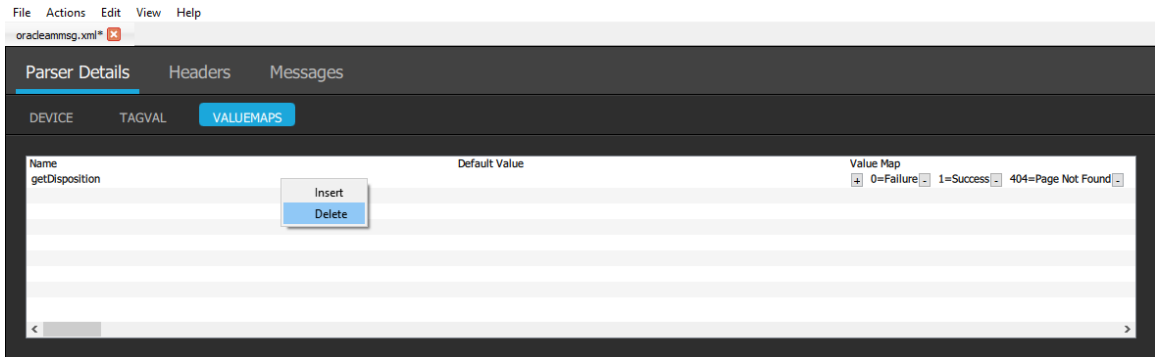
Field	Description
Name	It is a unique identity of a VALUMAPS.
Default Value	When there is no value relevant to a defined specific key, it will use default value
Value Map	It is a set of key value pair where a meta is found in the parser it is replaced by the value.

VALUEMAPS can be edited as per the requirements, you can insert new VALUMAPS containing the default values and delete any unwanted VALUMAPS.

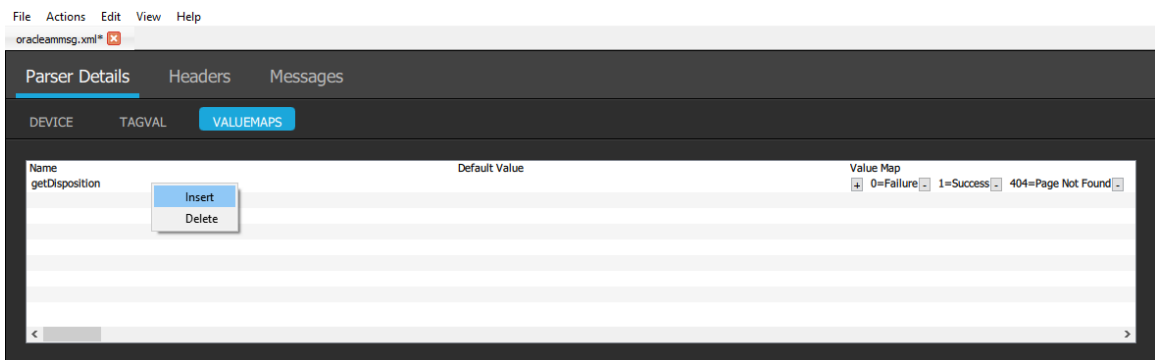
To edit an existing VALUMAPS click on it to make changes.

- Add - Click on  button to add a new key value pair
- Delete- click on  to delete an existing key value pair

To delete an unwanted VALUMAPS right click on the name column and select the “delete” option you want to delete.



To Insert a new VALUEMAPS right click on the name column and select the Insert option you want to insert.



New VALUEMAPS will be created with new name VALUEMAPS 1, VALUEMAPS 2 and so on which will contain default values.