Hewlett Packard
Enterprise

HPE Security ArcSight Quick Flex Parser Tool

Software Version: 1.0

Online Help

December 2, 2016

Legal Notices

Warranty

The only warranties for Hewlett Packard Enterprise products and services are set forth in the express warranty statements accompanying such products and services. Nothing herein should be construed as constituting an additional warranty. Hewlett Packard Enterprise shall not be liable for technical or editorial errors or omissions contained herein.

The information contained herein is subject to change without notice.

The network information used in the examples in this document (including IP addresses and hostnames) is for illustration purposes only.

HPE Security ArcSight products are highly flexible and function as you configure them. The accessibility, integrity, and confidentiality of your data is your responsibility. Implement a comprehensive security strategy and follow good security practices.

This document is confidential.

Restricted Rights Legend

Confidential computer software. Valid license from Hewlett Packard Enterprise required for possession, use or copying. Consistent with FAR 12.211 and 12.212, Commercial Computer Software, Computer Software Documentation, and Technical Data for Commercial Items are licensed to the U.S. Government under vendor's standard commercial license.

Copyright Notice

Support

Contact Information

Phone	A list of phone numbers is available on the HPE Security ArcSight Technical Support Page: https://softwaresupport.hpe.com/documents/10180/14684/esp-support-contact-list
Support Web Site	https://softwaresupport.hpe.com
Protect 724 Community	https://www.protect724.hpe.com

Contents

About the Quick Flex Parser Tool

Quick Flex Parser Tool helps you create a parser file quickly and efficiently. You can use the tool to create a base regex, tokens, and token filters for every message type. It validates and tests log file parsing and mapping, and generates a parser properties file that can be used in the FlexConnector framework.

Audience

The Quick Flex Parser Tool is intended for users who will be developing parser properties files that can be used with ArcSight products. It is expected that users will have expertise in regex expressions, parser development, and the FlexConnector framework.

Features and benefits

Quick Flex Parser Tool allows you to generate a parser file suitable for use in the FlexConnector framework by giving you the ability to do the following:

- load a log file up to 200MB in size
- replace the current log file in the project with a different log file
- search and filter messages in the Log View
- create and reuse tokens
- build a token repository
- construct token filters from tokens
- override the token regex or use the original token regex depending on the token filter
- change a token or token filter property in one place and having it applied globally
- switch to token filter edit mode from different places in the tool
- export token filter test results to files for further analysis

Quick Flex Parser Tool provides the following features to help you analyze the log file and track your progress:

- message highlighting in the Base Regex and Token Filter Managers to indicate whether tokens are parsing the log lines successfully
- message highlighting in the Log View to indicate if lines are parsed successfully and whether a particular message is being parsed by multiple token filters

- graphical statistics in the Log View to track your progress in analyzing the log file
- tests you can run to detect whether the parsing you defined makes sense; you can drill down into the test results to determine why a test might have failed

Workflow summary

The following tasks provide a high-level description of how to use Quick Flex Parser Tool to create a parser file, suitable for the FlexConnector framework.

1. Create a project

Quick Flex Parser Tool creates parser files within the context of a project. The project contains the definitions of your tokens, base regex, token filters and mappings based on the content of a log file. When you create a project you load the log file and identify the folder to store your results. See "Using the Quick Flex Parser Tool Landing Page".

2. Create the base regex

The base regex (also known as a preparser) is used to process headers from all messages in a file or stream. The base regex is a regular expression which corresponds to the regex in the connector parser file. The base regex must match all log lines in a file. Base regex provides the opportunity to further refine message processing defining message token filters. Edit the base regex until all messages are processed. See "Create a base regex".

3. Create tokens and token filters

Create tokens based on the content of the message. A token is a tag that identifies a data field or other useful information in a message. Verify that the tokens work for all of the specified messages. Use the tokens to build a token filter. A token filter is the tokenized form of a message or log record. See "Create a token", "Create a token filter", and "Override token regex".

4. Test the token filters

Test log lines against the token filters. The goal is to see if the parsing makes sense or if the matching against the log lines works. The log lines are parsed by a combination of the base regex and token filters. Ideally, each log line should be matched by only one token filter. If the log line is matched by more than one token filter, then you should resolve this situation. Create token filters as needed by using existing tokens or creating new tokens. The Quick Flex Parser Tool uses highlighting in the Log View to identify portions of the log line that are matched by the base regex and by the token filters. The Log View also identifies lines that match the token filter, do not match the token filter, or are matched by

multiple token filters. See "Quick Flex Parser Tool Log View", "Highlighting patterns in log lines", and "Managing and testing token filters".

5. Generate the parser properties file

Generate the parser properties file based on the tokens, base regex, token filters and mappings you created. This file can be imported into the FlexConnector framework. See "Generate a parser file ".

Using the Quick Flex Parser Tool Landing Page

You can perform these tasks on the Quick Flex Parser Tool Landing Page:

- Create a project
- Open project files
- View a workflow summary

Create a project

Navigation: Landing page>Create New

About:

Quick Flex Parser Tool creates a parser file within the scope of a project. The project contains the definitions of your tokens, token filters, and their respective mappings based on the content of a log file. The result of the project is a `.properties` file that is suitable for parsing the content of a log file within the FlexConnector framework.

Quick Flex Parser Tool saves the project to the folder you choose to store your project artifacts. The project can be reopened for further editing.

Procedure:

Create a new project:

1. Click **Create New** on the Quick Flex Parser Tool Landing Page to open the New Project dialog.
2. (Optional) Enter the following information in the Create New Project page:
   - the name of the vendor who provided the log file
   - the name of the product that produced the log file
   - the version number of the product

   If you do not specify these details at the beginning of a project, you can specify them later by selecting **File>Edit Project Properties** in the Log View.
3. (Optional) Click **Browse** to navigate to the log file.
   If you do not select a log file at the beginning of a project, you can select it later by selecting **File>Open Log File** in the Log View.

Note:
- Quick Flex Parser Tool cannot process comment lines. You must remove comment lines from the log file before loading it into the tool.
- For syslog log files, you must remove the header from the log file before loading it into the tool.

4. Click **Browse** to navigate to the folder that will store your project artifacts.
5. Click **Create**. The log file is loaded into the Quick Flex Parser Tool Log View.

Open project files

Navigation: Landing page>Open Files

Procedure:

Click **Open Files** and navigate to your project files. The log file and any associated project artifacts are loaded into the Quick Flex Parser Tool. Quick Flex Parser Tool project names have the format <*project_file_name*>`.json`.

Note: Each project file contains a path pointing to the location of the log file. If Quick Flex Parser Tool does not find the log file at that path, it notifies you and asks if you want to browse for the log file before opening the project.

View a workflow summary

Navigation: Landing page>Quick Overview

Procedure:

Click **Quick Overview** for a graphic representation of the Quick Flex Parser Tool workflow. For more information on Quick Flex Parser Tool, see

Creating tokens and filters

Open a saved project

Navigation: Log View>File>Open Project

Select **File>Open Project** and select the name of the project. The log file and any associated project artifacts are loaded into the Quick Flex Parser Tool. Quick Flex Parser Tool project names have the format <*project_file_name*>`.json`.

> **Note:** Each project file contains a path pointing to the location of the log file. If Quick Flex Parser Tool does not find the log file at that path, it notifies you and asks if you want to browse for the log file before opening the project.

Quick Flex Parser Tool Log View

The Quick Flex Parser Tool Log View opens when you create a new project or open an existing project. This view displays the log file messages and the token filters applied to the messages. The Log View also displays statistics for the token filters and the number of lines parsed by the filters.

The menu bar contains the following:

- **File**: contains commands to create a new project, open an existing project, open a file in the project, save the project, and edit project properties.
- **Base Regex Editor**: Click to open the Base Regex Editor where you can create and edit the base regex. See "Create a base regex".
- **Token Filter Editor**: Click to open the Token Filter Editor where you can create and edit token filters. See "Create a token filter".
- **Token Manager**: Click to open the Token Manager where you can create, update, and delete tokens.

See "Create a token".

- **Token Filter Manager**: Click to open the Token Filter Manager where you can view the token filter status, enable or disable token filters, and test the selected token filters. See "Managing and testing token filters".
- **Help**: Click to access the online help.

The ribbon above the Log View displays the following status and commands:

- **Total Logs**: Displays the total number of lines in the log file. Click to display the contents of the log file in the Log View Panel.
- **Base Parsed**: Number of log lines that are parsed successfully by the base regex. Click to display the parsed lines in the Log View Panel.
- **Base Unparsed**: Number of log lines that are not parsed by the base regex Click to display the unparsed lines in the Log View Panel.
- **Complete**: Number of log lines that are parsed by the base regex and at least one token filter. Click to display the lines in the Log View.
- **Incomplete**: Number of log lines that are not parsed by the base regex, a token filter, or both
- **Next Unparsed Line**: Click to go to the next line which is not parsed by either the base regex or a token filter.
- **Go to**: Enter a line number or text string to find the information you want.

The Settings drop-down contains these options:

- **Visualize Stats**: Select to graphically display the statistics that appear in the top toolbar.
- **Regex Highlight**: Select to display highlighting in the log lines. See "Highlighting patterns in log lines".

In the Log View Panel, the **Log Message** column displays the messages in the log file. The **Matched Token Filter** column displays the number of token filters that match a log line. Initially, it displays 0. As token filters are constructed and applied, this value changes to represent the number of token filters that are parsing a particular log line.

You can begin defining tokens and token filters for your project by selecting a message in the log file. This action opens the Token Filter Editor. See "Create a token filter".

The Log View displays these statistics to track your progress in parsing the log file:

- **Token Filter Coverage**: indicates the number of log lines that are parsed by a single token filter, by 2 token filters, and by 3 or more token filters
- **Token Filter Stats**: displays the number of matching messages for the top ten filters which are covering most of the log lines.

When you have finished defining your base regex, tokens and token filters, and are satisfied with the test results, click **Generate Parser** to generate the parser (`.properties`) file for the project. See "Generate a parser file ".

Creating token filters for messages

To create a token filter, complete these tasks:

- Create a base regex
- Create a token
- Create a token filter
- Create a mapping
- Override token regex

Create a base regex

Navigation: Log View>Base Regex Editor

About:

Specify a base regex (also known as a preparser). Connectors use a base regex to separate the header from the body of a log message. The header is considered to be those parts of a log line that are common to all messages. It is recommended that you create the base regex that processes all log lines successfully before creating and applying token filters to process the rest of the message.

Quick Flex Parser Tool assumes that you have used an external tool to create the base regex. Copy the base regex that you have written and paste it into the **Base Regex** field. You can also enter a base regex by typing directly in the **Base Regex** field.

Once you have a valid regex expression for the log line, click **Tokenize** to create base regex tokens. Except for the generated regex, you can edit any of the default values that Quick Flex Parser Tool assigns to the tokens.

Procedure:

1. Click **Base Regex Editor** on the **Log View**. The Base Regex Editor opens.
2. Define the base regex for the message in the **Base Regex** field.

   Note:
   - At a minimum, the base regex must be defined as (`.*`).
   - You must create and save at least two base regex tokens before you can create message tokens.
   - Quick Flex Parser Tool expects the base regex to contain at least one capture group.

- Quick Flex Parser Tool uses parentheses to represent captured elements. To represent a parenthesis as a literal character, you must escape it with the backslash character. For example: "\(".

3. Inspect the Log View to ensure that the base regex processes all lines successfully before you create any tokens. If the base regex does not parse all lines, then they will not be parsed correctly by the resulting parser properties file. See "Highlighting in the Log View".
4. Click **Tokenize**. Quick Flex Parser Tool opens a pop-up with a list of suggested tokens for the highlighted regex. You can accept the suggested tokens or dismiss the pop-up. If you accept the suggested tokens, then the tokenized regex selection is highlighted.

   Note:
   - The **Tokenize** button will not become active unless the regex is valid and successfully processes the entire log line in the Base Regex Editor.
   - After you click **Tokenize**,when you place the cursor in the regex, the corresponding piece of the log line is highlighted and information about the token is displayed in Token Details. See "Highlighting in the Base Regex Editor".
   - After you click **Tokenize**, you can examine how the tokens relate to the log line. Click **Matching Details** to display a table that lists each token, the regex defined for the token, and the portion of the log line it represents.

5. (Optional) Edit the information about the token. Select a token from the Base Token List. Provide this information in the Token Details region:
   a. Edit the **Token Name**.
   b. (Optional) Select a **Type** from the drop-down list. The **Format**. See "ArcSight Token Types" for a description of token types.

      Note: If you select **TimeStamp**, a **Format** field opens with a default time format. You can enter a time format or click the search button to select a predefined format. See "Date and time format symbols".

   c. Examine the **Regex** expression for the token. This field is populated by the regex defined for the token in the **Base Regex** field. You can change the value in this field only by editing the regex in the **Base Regex** field.
   d. (Optional) Enter a text **Description** for the token.
   e. (Optional) Select an **Assignment** from the drop-down list. See "ArcSight assignments ".
   f. Click **Save Token**. The name of the token appears in the **Base Token List**.
6. (Optional) Create mappings for the base regex tokens. See "Create a mapping".
7. Select a **Message ID Token** and **Message Token** from the drop-down lists. This will be translated to a sub-message in the parser file.

> **Note:** If you want to process the log file with the base regex only, then this step is not required.

The **Message ID Token** drop-down list contains the tokens created for the project. You associate one of these tokens with the **Message Token** to identify a sub-message.

A **Message ID** is not required, but it is desirable, because parsing performance is improved if the Message Token can be related to a token filter.

8. (Optional) The **Additional Data** switch generates a setting in the parser properties file `additionaldata.enabled = true/false`. The `true` setting tells the SmartConnectors to collect all the unused base regex tokens (that is, tokens which are not mapped to anything). For example, if the **Additional Data** switch is set to `true` for the token `TokenABC`, then the additional mapping `addionaldata.TokenABC = Token ABC` will be created. This value will not appear in the parser properties file, but it will be in the exported data or in the ESM view).

   If you do not want SmartConnectors to automatically collect the unused tokens, you always have option of creating your own `additionaldata` when you do mapping. For example, you can manually enter something similar to the following in the parser properties file: `additonaldata.ANY_CUSTOM_NAME = TokenABC`.

Create a token

Navigation: Log View>Token Manager or right-click a selected portion of the of the raw log message in the Token Filter Editor

About:

A token is a tag that identifies a data field or other useful information in a message. Typically, the name of the tag will be the name of the field it applies to. A token's properties apply to each filter the token is used in. If you change a token's properties, then the change will be reflected in each filter that uses the token.

Procedure:

The Token Manager contains a list of tokens that have been defined and a region where you can create or edit message tokens. You can create only message tokens in the Token Manager. To create base regex tokens, use the Base Regex Editor. See "Create a base regex".

Use one of these methods to open the Token Manager:

- Select **Token Manager** in the Log View.
- Select a message in the Log View Panel. The message appears in the **Raw Log** working area of the Token Filter Editor. Select and right-click a part of the message. The Token Manager opens as a pop-up.

Provide the following information in the Token Manager.

1. Enter a **Token Name** in the Token Properties table.
2. (Optional) Select a **Type** from the drop-down list. See "ArcSight Token Types" for a description of token types.

   Note: If you select **TimeStamp**, a **Format** field opens with a default time format. You can enter a time format or click the search button to select a predefined format. See "Date and time format symbols".

3. Edit the **Regex** expression for the token. When a token is created, its initial value is the default regex: `\\S+`. Verify that the edited token regex processes the selected message segment.

   Note: Quick Flex Parser Tool does not support the use of capture symbols `(...)` or optional symbols `(?...?)` in the regex expression. Use the **Capture** and **Optional** toggle buttons instead.

4. (Optional) Set the value of the **Capture** and/or **Optional** toggle buttons:
   - **Capture**—Set to True to capture the value matching the token regex as a back reference. The default is False.
   - **Optional**—Set to True if the token value must be present in the message. The default is False.
5. (Optional) Enter a text **Description** for the token.
6. (Optional) Select an **Assignment** from the drop-down list. See "ArcSight assignments ".
7. Click **Save** to save the token. The name of the token appears in the **Token List**.
8. Repeat steps 1-7 until you have defined tokens which satisfy all of the log lines.

Create a token filter

Navigation: Log View>Token Filter Editor

About:

A token filter is the tokenized form of a message or log record. It is used to create a parser file and to exercise various properties.

Procedure:

1. Highlight and right-click a log line in the Log View and select **Token Filter Editor**. The log line appears in the **Raw Log** and **Token Filter** fields of the Token Filter Editor.
2. Enter a name for the token filter in the **Filter ID** field.
3. Right-click the log line in the **Token Filter** field to open a pop-up containing the list of available tokens. You can create, edit or delete tokens:
   a. Click **+ New** to create a new token or select a token in the list to edit. See "Create a token".
   b. Click **X Delete** to remove a selected token from the project.

c. Enable **Token Details** to see more information about a selected token. You can edit the details, if necessary.

d. Enable **Override Regex** in the Token Details if the token definition should override the token regex. See "Override token regex".

4. Assign mappings to the tokens. See "Create a mapping".
5. Click **Apply** to add the token to the **Token Filter** field. Quick Flex Parser Tool highlights portions of the log line that match the regex defined in the filter (see "Highlighting patterns in log lines").

   Note:
   - The order in which tokens appear, and any spaces or punctuation you add to the token filter, is important.
   - Quick Flex Parser Tool uses parentheses to represent captured elements. To represent a parenthesis as a literal character, you must escape it with the backslash character. For example: "\ (".

6. When you are satisfied with the results of your token filter, click **Save**.

Create a mapping

Navigation: Log View>Base Regex Editor, Token Filter Editor

About:

A mapping describes the relationship or the process of establishing the relationship between a log message field and an ArcSight schema field. The mappings describe how the token will map to the fields in ArcSight products, such as Logger, ArcMC, Express, and so on. More than one mapping can be associated with a field.

Procedure:

1. Click **+ New** to create a new mapping or select an existing mapping from the list to edit.
2. (Optional) Choose an **Assignment** from the drop-down list. See ArcSight assignments .
3. (Optional) Enter a text **Description** of the mapping.
4. Select an **Operation** from the drop-down list. See "ArcSight Operations".
5. Enter any **Arguments** that are required by the selected operation. See "ArcSight Operations".
6. Click **Save Mapping** to add the mapping to the Mapping List field.

Override token regex

Navigation: Log View>Token Manager

About:

Occasionally, the grammar of a message does not support the regex used by the token. However, if you do not want to modify the token regex that works for all other messages, then you can override the token regex to allow for exceptions in the token's definition. Overrides to a token's regex have the following characteristics:

- they apply only to the token filter that contains the token with the overrides
- they do not modify token properties in the token set

Procedure:

1. Enable the **Override Regex** selector on the Token Manager pop-up.
2. Edit the regex expression in the **New Regex** field. Note that other fields in the pop-up cannot be edited.
3. Save the token. The token definition is overwritten with the new regex and is saved in the Token List.

Highlighting patterns in log lines

Quick Flex Parser Tool uses highlighting to indicate when a pattern in a log line matches the regex defined in a token, base token filter, or token filter. The tool applies highlighting differently, depending on whether you are in the Log View or the Token Filter Editor.

- Highlighting in the Log View
- Highlighting in the Token Filter Editor
- Highlighting in the Base Regex Editor

Highlighting in the Log View

Quick Flex Parser Tool applies highlighting in the Log View depending on whether you are working with a base regex or a Token Filter:

- Base regex - If your base regex pattern provides a partial match with the log line, then the matching portion of the line is highlighted in purple from the 0^{th} character in the line to the N^{th} character.
- Token Filter - If the selected Token Filter pattern matches the entire log line, then the entire line will be highlighted in green. However if a base regex is also valid for the line, then the log line will be highlighted in purple for the base token filter match and the remaining part of the line will be highlighted blue for the token filter match.

Highlighting in the Token Filter Editor

Quick Flex Parser Tool applies highlighting to a log line for the base regex and Token Filter according to their regex patterns. The matched portions of the line are highlighted in blue, for example:

```
word_a word_b 1 C
```

Highlighting in the Base Regex Editor

After you click **Tokenize**,when you place the cursor in a token in the base regex, then the corresponding piece of the log line is highlighted and information about the token is displayed if the log line has content expected to be covered by that regex.

If you place the cursor in a token in the base regex and highlighting is not displayed in the log line, then this means that the token is not present in the log line.

Managing and testing token filters

You can perform the following actions in the Token Filter Manager:

- Manage token filters on the Token Filter List tab
- Test token filters on the Token Filter Test tab

Manage token filters

Navigation: Log View>Token Filter Manager>Token Filter List tab

About:

Use the Token Filter List tab of the Token Filter Manager to view the content and status of the token filters you have created. For each token filter, the table displays the tokens used in the filter, whether it is currently being used to parse the log file and the number of log lines it has matched.

The position of the token filters in the list is important. Quick Flex Parser Tool applies token filters to the log file from top to bottom. Typically, token filters are ordered from most specific to least specific.

Procedure:

For each token filter in the Token Filter list tab, you can perform the following actions:

- View the list of tokens that comprise the token filter.
- View the number of log lines each token filter has matched and whether it is currently being used to parse the log file.
- Enable or disable the token filter.
- Change the position of the selected token filter in the list by clicking **Move Up** or **Move Down**.
- Double-click the token filter name or click **Edit** to display the definition of the token filter.
- Select the token filter and click **Delete** to remove the token filter from the project.
- Test the validity of your token filters against the log file. See "Test token filters".

Test token filters

Navigation: Log View>Token Filter Manager>Token Filter Test tab

About:

Use the Token Filter Test tab of the Token Filter Manager to test the performance of your base regex and token filters against the log file. You can test a single token filter or any combination of filters.

The Token Filter Test region displays the list of token filters, the tokens contained by the token filter, and its status: enabled, disabled, or invalid. You can select filters and click **Display Results** to display the performance of the filters against the log file in the Results region. Click **Export** to export the results to a CSV-format file.

Note: You can export results only if you select the Base Regex filter, a single filter, or all filters.

The Results regions display grids that identify the tokens used in the filter, the schema events they are mapped to, and any assignments that are applied. It also displays grids that identify matched and unmatched lines for selected token filters. Each section of the Results region can be exported individually to a CSV-format file.

Procedure:

You can perform the following actions in the Token Filter Test tab:

Select the Base Regex filter and click **Display Results** to display the following information. You can click **Export** to save the **Unmatched Results** to a CSV-format file.

- **Multi-header match**—This grid displays the names of the tokens in the base regex and the values of any associated assignment names and schema fields.
- **Match Results against Base Regex**—This grid displays the tokens used in the base regex and any schema elements and assignments associated with them. An additional grid displays the raw log lines matched against the base regex and their contents.
- **Unmatched Results against Base Regex**—This grid displays the log lines that do not match the base regex.

Select one token filter from the list of filters and click **Display Results**to display the following information. You can click **Export** to save the **Match Based on Line** results to a CSV-format file.

- **Multi-header match**—This grid displays the names of the tokens in the token filter and base regex and the values of any associated assignment names and schema fields.
- **Match Results against <*token name*>**—This grid displays the tokens used in the selected token filter and any schema elements and assignments associated with them. An additional grid displays the parts of the message that are matched by each token in the filter.
- **Matched Based on Line**—This grid displays a list of messages that are matched by the selected token filter.

Select multiple token filters and click **Display Results**to display the following information. **Export** is not available for this scenario.

- **Matched Lines against <*token filter names*>**—This grid displays a list of the messages that are matched by more than one token filter.

  Select a line in the grid to open a Details pop-up that lists the values of the tokens used in each token filter.

Select all token filters and click **Display Results**to display the following information. You can click **Export** to save the **Unmatched Lines** to a CSV-format file.

- **Matched against Selected Token Filters** —This grid displays a list of the messages that are matched by more than one token filter.
- **Unmatched Lines**—This grid displays a list of the messages that are not matched by any token filter.

Click **View** on an unmatched line. The line opens in the Token Filter Editor where you can continue to work on the message.

Generate a parser file

About:

The Quick Flex Parser Tool can generate a parser file suitable for use in the ArcSight FlexConnector framework. The parser file contains the definitions of your tokens, base regex, token filters, and token mappings.

The minimum requirements for generating a parser file is a base regex which successfully parses the log file.

Procedure:

1. Click **Generate Parser** in the Log View to generate a parser file. You can modify the generated content and copy it to a separate file.
2. Click **Export** to save the parser properties file. By default, the file will be saved as <*project_name*>`.properties` in the folder that stores your project files.

Note: For syslog log files, the header must be restored to the file before the parser file is tested against the connector.

ArcSight token types

Token types are important because tokens can be mapped only to ArcSight event fields with matching types. Event fields and their types are listed in the *ArcSight Console User's Guide*, in the Reference Guide, under Data Fields.

Type	Meaning	Format
Integer	A number from -2147483648 to 2147483647.	n/a
IPAddress	An IPv4 address (for example: 1.1.1.1). For IPv6-aware parsers, this can be an IPv4 or an IPv6 address (for example: fdeb:f59b:2e13:56c9:xxxx:xxxx:xxxx:xxxx).	n/a
Long	A number from -9223372036854775808 to 9223372036854775807.	n/a
MacAddress	An Ethernet MAC address of the form: 00-06-3E-22-51-B9 or 00:06:3E:22:51:B9.	n/a
String	Any free form sequence of characters.	n/a
TimeStamp	A date, a time or a date and a time.	Date/time format (see "Date and Time Format Symbols")

Date and time format symbols

The following date and time formats are defined in Quick Flex Parser Tool:

- MMM dd HH:mm:ss.SSS zzz
- MMM dd HH:mm:ss.SSS
- MMM dd HH:mm:ss zzz
- MMM dd HH:mm:ss
- MMM dd yyyy HH:mm:ss.SSS zzz
- MMM dd yyyy HH:mm:ss.SSS
- MMM dd yyyy HH:mm:ss zzz
- MMM dd yyyy HH:mm:ss
- ddMMyyyy HH:mm:ss
- MM-dd-yyyy HH:mm:ss
- yyyy-MM-dd HH:mm:ss.SSS
- yyyy-MM-dd HH:mm:ss

For example, for this format: `yyyy-MM-dd HH:mm:ss`

Use single quotes around text that is not meant to be interpreted as date format characters. Use this example for a date like: 2016.07.04 AD at 12:08:56 PDT.

```
yyyy.MM.dd G 'at' HH:mm:ss z
```

Use two single quotes to insert a single quote. Use this example for a date like: Wed, Jul 4, '16.

```
EEE, MMM d, ''yy
```

This table contains date and time format symbols:

Symbol	Meaning	Presentation	Examples
G	Era designator	(Text)	AD
y	Year	(Number)	2016 or 06
Y	Week year	Year	2016;16
M	Month in year	(Text & Number)	July or Jul or 07
w	Week in year	(Number)	27
W	Week in month	(Number)	2
D	Day in year	(Number)	129

Symbol	Meaning	Presentation	Examples
d	Day in month	(Number)	10
F	Day of week in month	(Number)	2 (indicating 2nd Wed. in July)
E	Day in week	(Text)	Tuesday or Tue
u	Day number of week	(1=Monday, ..., 7=Sunday)	Number
a	Am/pm marker	(Text)	AM or PM
H	Hour in day (0~23)	(Number)	0
k	Hour in day (1~24)	(Number)	24
K	Hour in am/pm (0~11)	(Number)	0
h	Hour in am/pm (1~12)	(Number)	12
m	Minute in hour	(Number)	30
s	Second in minute	(Number)	55
S	Millisecond	(Number)	978
z	Time zone	General time zone	Pacific Standard Time or PST or GMT-08:00
Z	Time zone	RFC 822 time zone	-0800 (indicating PST)
X	Time zone	ISO 8601 time zone	-08; -0800; -08:00

ArcSight assignments

An assignment can be either a mapping or a rule. Mappings are mapped to ArcSight event fields, such as `event.sourceAddress`. The type of the token must match the type of the ArcSight Event field.

See the numbered Range Notes (n) following this table for further explanations of certain field ranges.

A rule provides a level of indirection between the user and the ArcSightESM schema field a value is mapped to. For more information, see "Quick Flex Parser Tool rules".

The Assignments drop-down list in the Quick Flex Parser Tool contains both mappings and rules. This table lists ArcSight mappings. For descriptions of the rules, see "Quick Flex Parser Tool rules".

ArcSight Rules, Mappings, and Schema Names	Type	Length	Range
ACL Name (rule)	See "Quick Flex Parser Tool rules".		
Additional Data (rule)	See "Quick Flex Parser Tool rules".		
AV Engine Version (rule)	See "Quick Flex Parser Tool rules".		
Application Protocol event.applicationProtocol	String	31	n/a
Base Event Count event.baseEventCount	Integer	n/a	0 -> 2^{31}-1
Bytes In event.bytesIn	Long	n/a	0 -> 2^{31}-1
Bytes Out event.bytesOut	Long	n/a	0 -> 2^{31}-1
Category Behavior event.categoryBehavior	String	1023	n/a (1)
Category Device Group event.categoryDeviceGroup	String	1023	n/a (1)
Category Object event.categoryObject	String	1023	n/a (1)
Category Outcome event.categoryOutcome	String	1023	n/a (1)
Category Significance event.categorySignificance	String	1023	n/a (1)
Category Technique event.categoryTechnique	String	1023	n/a (1)

ArcSight Rules, Mappings, and Schema Names	Type	Length	Range
Crypto Signature event.cryptoSignature	String	512	n/a
Custom URI event.customURI	String	-	n/a (2)
Destination Account (rule)	See "Quick Flex Parser Tool rules".		
Destination Address (rule)	See "Quick Flex Parser Tool rules".		
Destination Address event.destinationAddress	IPAddress	n/a	IPv4 (3)
Destination Dns Domain event.destinationDnsDomain	String	255	n/a
Destination Host (rule)	See "Quick Flex Parser Tool rules".		
Destination Host Name event.destinationHostName	String	1023	n/a
Destination Mac Address event.destinationMacAddress	MacAddress	n/a	MAC (4)
Destination Nt Domain event.destinationNtDomain	String	255	n/a
Destination Port event.destinationPort	Integer	n/a	0 ->65535
Destination Process Name event.destinationProcessName	String	1023	n/a
Destination Service Name event.destinationServiceName	String	1023	n/a
Destination Translated Address event.destinationTranslatedAddress	IPAddress	n/a	IPv4 (3)
Destination Translated Port event.destinationTranslatedPort	Integer	n/a	0 -> 65535
Destination Translated Zone URI event.destinationTranslatedZoneURI	String	-	n/a (2)
Destination User Id event.destinationUserId	String	1023	n/a
Destination User Name event.destinationUserName	String	1023	n/a
Destination User Privileges event.destinationUserPrivileges	String	1023	n/a

ArcSight Rules, Mappings, and Schema Names	Type	Length	Range
Destination Zone URI event.destinationZoneURI	String	-	n/a (2)
Device Action event.deviceAction	String	63	n/a
Device Address (rule)	See "Quick Flex Parser Tool rules".		
Device Address event.deviceAddress	IPAddress	n/a	IPv4 (3)
Device Custom Date 1 event.deviceCustomDate1	TimeStamp	n/a	n/a (5)
Device Custom Date 1 Label event.deviceCustomDate1Label	String	1023	n/a
Device Custom Date 2 event.deviceCustomDate2	TimeStamp	n/a	n/a (5)
Device Custom Date 2 Label event.deviceCustomDate2Label	String	1023	n/a
Device Custom IPv6 Address 1 event.deviceCustomIPv6Address1	IPv6 Address	n/a	IPv6 (8)
Device Custom IPv6 Address 1 Label event.deviceCustomIPv6Address1Label	String	1023	Should be “Device IPv6 Address”. See also "Device Address or Host" in "Quick Flex Parser Tool rules".
Device Custom IPv6 Address 2 event.deviceCustomIPv6Address2	IPv6 Address	n/a	IPv6 (8)
Device Custom IPv6 Address 2 Label event.deviceCustomIPv6Address2 Label	String	1023	Should be “Source IPv6 Address”. See also "Source Address or Host" in "Quick Flex Parser Tool rules".
Device Custom IPv6 Address 3 event.deviceCustomIPv6Address3	IPv6 Address	n/a	IPv6 (8)
Device Customer IPv6 Address 3 Label event.deviceCustomerIPv6Address3Label	String	1023	Should be “Destination IPv6 Address”. See also "Destination Address or Host" in "Quick Flex Parser Tool rules".
Device Custom Number 1 event.deviceCustomNumber1	Long	n/a	-2^{63} -> $2^{63}-1$

ArcSight Rules, Mappings, and Schema Names	Type	Length	Range
Device Custom Number 1 Label event.deviceCustomNumber1Label	String	1023	n/a
Device Custom Number 2 event.deviceCustomNumber2	Long	n/a	- 2^{63} -> 2^{63}-1
Device Custom Number 2 Label event.deviceCustomNumber2Label	String	1023	n/a
Device Custom Number 3 event.deviceCustomNumber3	Long	n/a	-263 -> 263-1
Device Custom Number 3 Label event.deviceCustomNumber3Label	String	1023	n/a
Device Custom String 1 event.deviceCustomString1	String	1023 (4.x) 4000 (5.x)	n/a
Device Custom String 1 Label event.deviceCustomString1Label	String	1023	n/a
Device Custom String 2 event.deviceCustomString2	String	1023 (4.x) 4000 (5.x)	n/a
Device Custom String 2 Label event.deviceCustomString2Label	String	1023	n/a
Device Custom String 3 event.deviceCustomString3	String	1023 (4.x) 4000 (5.x)	n/a
Device Custom String 3 Label event.deviceCustomString3Label	String	1023	n/a
Device Custom String 4 event.deviceCustomString4	String	1023 (4.x) 4000 (5.x)	n/a
Device Custom String 4 Label event.deviceCustomString4Label	String	1023	n/a
Device Custom String 5 event.deviceCustomString5	String	1023 (4.x) 4000 (5.x)	n/a
Device Custom String 5 Label event.deviceCustomString5Label	String	1023	n/a
Device Custom String 6 event.deviceCustomString6	String	1023 (4.x) 4000 (5.x)	n/a
Device Custom String 6 Label event.deviceCustomString6Label	String	1023	n/a

ArcSight Rules, Mappings, and Schema Names	Type	Length	Range
Device Dns Domain event.deviceDnsDomain	String	255	n/a
Device Domain event.deviceDomain	String	1023	n/a
Device Event Category event.deviceEventCategory	String	1023	n/a
Device Event Class Id event.deviceEventClassId	String	1023	n/a
Device External Id event.deviceExternalId	String	255	n/a
Device Facility event.deviceFacility	String	1023	n/a
Device Host Name event.deviceHostName	String	63	n/a
Device Host (rule)	See "Quick Flex Parser Tool rules".		
Device Inbound Interface event.deviceInboundInterface	String	15	n/a
Device Mac Address event.deviceMacAddress	MacAddress	n/a	MAC (4)
Device Nt Domain event.deviceNtDomain	String	255	n/a
Device Outbound Interface event.deviceOutboundInterface	String	15	n/a
Device Payload Id event.devicePayloadId	String	128	n/a
Device Process Name event.deviceProcessName	String	1023	n/a
Device Product event.deviceProduct	String	63	n/a
Device Receipt Time event.deviceReceiptTime	TimeStamp	n/a	n/a (5)
Device Severity event.deviceSeverity	String	63	n/a
Device Time Zone event.deviceTimeZone	String	255	n/a

ArcSight Rules, Mappings, and Schema Names	Type	Length	Range
Device Translated Address event.deviceTranslatedAddress	IPAddress	n/a	IPv4 (3)
Device Translated Zone URI event.deviceTranslatedZoneURI	String	-	n/a (2)
Device Vendor event.deviceVendor	String	63	n/a
Device Version event.deviceVersion	String	31	n/a
Device ZoneURI event.deviceZoneURI	String	-	n/a (2)
End Time event.endTime	TimeStamp	n/a	n/a (5)
External Id event.externalId	String	40	n/a
File Create Time event.fileCreateTime	TimeStamp	n/a	n/a (5)
File Hash event.fileHash	String	255	n/a
File Id event.fileId	String	1023	n/a
File Modification Time event.fileModificationTime	TimeStamp	n/a	n/a (5)
File Name event.fileName	String	1023	n/a
File Path event.filePath	String	1023	n/a
File Permission event.filePermission	String	1023	n/a
File Size event.fileSize	Long	n/a	0 -> 2^{63}-1
File Type event.fileType	String	1023	n/a
Flex Date 1 event.flexDate1	TimeStamp	n/a	n/a (5)
Flex Date 1 Label event.flexDate1Label	String	128	n/a

ArcSight Rules, Mappings, and Schema Names	Type	Length	Range
Flex Number 1 event.flexNumber1	Long	n/a	- 2^{63} -> 2^{63}-1
Flex Number 1 Label event.flexNumber1Label	String	128	n/a
Flex Number 2 event.flexNumber2	Long	n/a	-2 ^{63} -> 2^{63}-1
Flex Number 2 Label event.flexNumber2Label	String	128	n/a
Flex String 1 event.flexString1	String	1023	n/a
Flex String 1 Label event.flexString1Label	String	128	n/a
Flex String 2 event.flexString2	String	1023	n/a
Flex String 2 Label event.flexString2Label	String	128	n/a
Group (rule)	See "Quick Flex Parser Tool rules".		
Instance (rule)	See "Quick Flex Parser Tool rules".		
Message event.message	String	1023	n/a
Name event.name	String	512	n/a (9)
Object (rule)	See "Quick Flex Parser Tool rules".		
Old File Create Time event.oldFileCreateTime	TimeStamp	n/a	n/a (5)
Old File Hash event.oldFileHash	String	255	n/a
Old File Id event.oldFileId	String	1023	n/a
Old File Modification Time event.oldFileModificationTime	TimeStamp	n/a	n/a (5)
Old File Name event.oldFileName	String	1023	n/a
Old File Path event.oldFilePath	String	1023	n/a

ArcSight Rules, Mappings, and Schema Names	Type	Length	Range
Old File Permission event.oldFilePermission	String	1023	n/a
Old File Size event.oldFileSize	Long	n/a	0 -> 2^{63}-1
Old File Type event.oldFileType	String	1023	n/a
Raw Event event.rawEvent	String	4000	n/a (7)
Request Client Application event.requestClientApplication	String	1023	n/a
Request Context event.requestContext	String	2048	n/a
Request Cookies event.requestCookies	String	1023	n/a
Request Method event.requestMethod	String	1023	n/a
Request Url event.requestUrl	String	1023	n/a
Rule Name (rule)	See "Quick Flex Parser Tool rules".		
Signature Version (rule)	See "Quick Flex Parser Tool rules".		
Source Account (rule)	See "Quick Flex Parser Tool rules".		
Source Address (rule)	See "Quick Flex Parser Tool rules".		
Source Address event.sourceAddress	IPAddress	n/a	IPv4 (3)
Source Dns Domain event.sourceDnsDomain	String	255	n/a
Source Host (rule)	See "Quick Flex Parser Tool rules".		
Source Host Name event.sourceHostName	String	1023	n/a
Source Mac Address event.sourceMacAddress	MacAddress	n/a	MAC (4)
Source Nt Domain event.sourceNtDomain	String	255	n/a
Source Port event.sourcePort	Integer	n/a	0 -> 65535

ArcSight Rules, Mappings, and Schema Names	Type	Length	Range
Source Process Name event.sourceProcessName	String	1023	n/a
Source Service Name event.sourceServiceName	String	1023	n/a
Source Translated Address event.sourceTranslatedAddress	IPAddress	n/a	IPv4 (3)
Source Translated Port event.sourceTranslatedPort	Integer	n/a	0 -> 65535
Source Translated Zone URI event.sourceTranslatedZoneURI	String	-	n/a (2)
Source User Id event.sourceUserId	String	1023	n/a
Source User Name event.sourceUserName	String	1023	n/a
Source User Privileges event.sourceUserPrivileges	String	1023	n/a
Source Zone URI event.sourceZoneURI	String	-	n/a (2)
Start Time event.startTime	TimeStamp	n/a	n/a (5)
Transport Protocol event.transportProtocol	String	31	n/a (6)
Virus Name (rule)	See "Quick Flex Parser Tool rules".		

Range Notes

1. Although these fields can be set using the FlexConnector properties file, the recommended way is to create a categorization file. For more about the possible values, see the "Categories" topic in the *Console Help* or the *ArcSight Console User's Guide*. Also, see "FlexConnectors and Categorization" in the *FlexConnector Developer's Guide*.
2. Although URI fields can be set using the FlexConnector properties file, these are really links to resources in the database. Therefore, it is recommended that those fields be set using the network-model and customer-setting features.
3. This is an IPv4 address (from 0.0.0.0 to 255.255.255.255) or an IPv6 address (xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx).
4. This is a MAC address: XX:XX:XX:XX:XX:XX or XX-XX-XX-XX-XX-XX.
5. This is a timestamp stored as milliseconds since January 1, 1970.

6. The options are: TCP, UDP, ICMP, IGMP, ARP.
7. Set `PreserveRawEvent` to `Yes` to have the connector automatically preserve the original event log received from the device. With the default `No`", you can configure this field. To find the `PreserveRawEvent` field in the ArcSight Console interface, go to the **Connectors resource tree > Configure > Default tab > Content >Processing section > PreserveRawEvent**.
8. For a non-IPv6-aware parser, the IPv6 fields (deviceCustomIPv6Address1, 2, and 3) should consistently use 1 for device, 2 for source, and 3 for destination. The labels for them will automatically be set if the IPv6 address field is set, but if your ArcSight Console parser sets them explicitly, it should use the exact strings shown above.

   For an IPv6-aware parser, the IPv6 fields (deviceCustomIPv6Address1, 2, and 3) can contain either IPv4 or IPv6 addresses. In practice, these fields should rarely be used. If they are, the labels should be set to an appropriate value.
9. The `name` field is mandatory.

Quick Flex Parser Tool rules

A mapping rule provides a level of indirection between the user and the ArcSight ESM schema field a value is mapped to. A value comes from either a token, the value captured by the token's regular expression when it is used in the token filter, or the result of an operation that is part of the token or token filter.

A mapping rule provides:

- support for common operations so that you do not need to repeatedly implement them in each token filter or parser
- a user-friendly name for the schema field
- the ability to change how a value would be applied to the schema without requiring the user to change token filters or parsers

There is a distinction between selecting a mapping that simply writes to a schema field and one that has operations. The majority of users will simply do mapping.

The available operations, when the mapping rule has operations, are described in "ArcSight operations". In this case, the mapping rule supports these uses:

- the value must be tested and modified in some way that relates to the schema field
- the destination schema field must be selected based on an ArcSight mapping convention. It supports consistent mapping for cases when there are no natural schema fields to map to
- support the complexity of the ArcSight schema when a value might be mapped to different places

The following table describes the mapping rules available in Quick Flex Parser Tool.

Rule Name	Description and Arguments
ACL Name	Defines the name of the Access Control List (ACL). **Arguments:** • ACL Name—The value of ACL Name is mapped to event.deviceCustomString1. • ACL Label—If defined, the value is mapped to event.deviceCustomString1Label. If it is not defined, "ACL name" is mapped to event.deviceCustomString1Label.
Additional Data	Allows you to specify a custom additionaldata name when you perform mapping. For example, when you are mapping `Token0`, you can enter `CUSTOM_NAME` as an argument. The following will appear in the parser properties file: `additonaldata.CUSTOM_NAME = Token0`

Rule Name	Description and Arguments
AV Engine Version	Defines the Anti Virus Engine version. **Arguments:** • AV Engine Version—The value of AV Engine Version is mapped to event.deviceCustomString2. • AV Engine Version Label—If defined, the value is mapped to Device Custom String 2 Label. If it is not defined, "AV Engine Version" is mapped to event.deviceCustomString2Label.
Destination Account	Identifies the target account of an event. If the account name contains a Windows domain, it will split the domain name out of the account name. The domain name is written to event.destinationNtDomain. **Arguments:** • Account Name—The domain name (if it exists) is mapped to event.destinationNtDomain and Account Name is mapped to event.destinationUserName.
Destination Address or Host	Destination target of an event; typically this will be a host address or a host name. The rule evaluates whether the target is an address or a host name and maps it to the appropriate field. **Arguments:** There are three possible mappings: • if value pattern matches an IPV4 address then the value is mapped to event.destinationAddress. • if the value pattern matches an IPV6 address then the value is mapped to event.customIPv6Address3 and "Destination IPv6 Address" is mapped to event.customIPv6Address3Label. • if neither of these conditions match, then the value is mapped to event.destinationHostName.
Device Address or Host	Device is the system where the event occurred, or from where the event was retrieved. The rule evaluates the value pattern and maps the value to the appropriate field. **Arguments:** There are three possible mappings: • if the pattern matches an IPv4 address, then the value is mapped to event.destinationAddress in the case of an address, or event.deviceAddress in the case of a host. • if the pattern matches an IPv6 address then the value is mapped to event.customIPv6Address1 and "Device IPv6 Address" is mapped to event.customIPv6Address1Label. • if neither of these conditions match, then the value is mapped to event.destinationHostName in the case of an address, or event.deviceHostName in the case of a host.

Rule Name	Description and Arguments
Group	A Group can be anything that an application or operating system refers to as a group. TheArcSight event schema does not support groups, so if you must define a group, use these conventions to handle the values. **Arguments:** • Group Name—The value of Group Name is mapped to event.deviceCustomString6. • Group Label—If defined, the value is mapped to event.deviceCustomString6Label. If it is not defined, "Group" is mapped to event.deviceCustomString6Label.
Instance	An Instance is a representation of a distinct event. If the product supports instance, use these conventions to map the values: **Arguments:** • Instance Field Value—The value is mapped to event.deviceCustomString3. • Instance Label —If defined, the value is mapped to event.deviceCustomString3Label. If it is not defined, "Instance" is mapped to event.deviceCustomString3Label.
Object	A generic object. Any object that does not have a natural rule. Use these conventions to map the values: **Arguments:** • Object Name—The value of Object Name is mapped to event.deviceCustomString6. • Object Label—If defined, the value is mapped to event.deviceCustomString6Label. If it is not defined, "Object name" is mapped to event.deviceCustomString6Label.
Rule Name	Any instance of a rule name. For example, this can be a firewall rule, a mapping rule, and so on. Use these conventions to map the values: **Arguments:** • Rule Name—The value of Rule Name is mapped to event.deviceCustomString1. • Rule Label—If defined, the value is mapped to event.deviceCustomString1Label. If it is not defined, "Rule name" is mapped to event.deviceCustomString1Label.
Signature Version	This is typically an IDS (intrusion detection system) signature version number. **Arguments:** • Signature Version—The value of Signature Version is mapped to event.deviceCustomString2. • Signature Version Label—If defined, the value is mapped to event.deviceCustomString2Label. If it is not defined, "Signature version" is mapped to event.deviceCustomString2Label.

Rule Name	Description and Arguments
Source Account	The account of the source that triggered the event. If the account name contains a Windows domain, it will split the domain name out of the account name. The domain name is written to event.destinationNtDomain. **Arguments:** • Account Name—The domain name (if it exists) is mapped to event.destinationNtDomain and Account Name is mapped to event.destinationUserName.
Source Address or Host	The address of the system or device that is the origin of an event or the location where the event occurred. **Arguments:** There are three possible mappings: • if the pattern matches an IPv4 address, then the value is mapped to event.sourceAddress. • if the pattern matches an IPv6 address, then the value is mapped to event.customIPv6Address2 and "Source IPv6 Address" is mapped to event.customIPv6Address2Label. • if neither of these conditions match, then Source Address is mapped to event.sourceHostName.
Virus Name	The name that a product assigns to a virus. **Arguments:** • Virus Name—The value of Virus Name is mapped to event.deviceCustomString1. • Virus Label—If defined, the value is mapped to event.deviceCustomString1Label. If it is not defined, "Virus name" is mapped to event.deviceCustomString1Label.

ArcSight operations

Operations are used primarily when tokens are mapped to HPE ArcSight event fields.

The following table describes the most commonly used operations. For a complete list of operations that can be used in Quick Flex Parser Tool, see "Appendix A: ArcSight Operations" in the "*FlexConnector Developer's Guide*".

The values in the **Arguments** column have the following meaning:

- token_name—the name of a token, for example, `Token0`, `TimeToken`.
- expression—can be a token name, a quoted string, or null.
- (string) constant—a quoted string, for example, `"string constant"`.
- null—an empty value, for example `,,`.
- regex_expression—a regex expression. Must be enclosed in parentheses, for example `(\s+)`.

Note: The Quick Flex Parser Tool does not support nested operations.

Operation	Arguments	Return Type	Definition

__concatenate	token_name1, expression1, expression2	String	The arguments can be literal strings or other values of various types. The result is a string that consists of all of these arguments concatenated together. Examples:
			`"Active",protocol," Ports: ",portnum`
			`"CompanyName: [", CompanyName,"]"`
			`"PF: ",PassOrBlock`

__extractNTDomain	token_name	String	The argument is a string. If it contains a back slash, the part of the string up to but not including that backslash is returned. Otherwise the entire string is returned.
__extractNTUser	token_name	String	The argument is a string in the form 'domain\user', where domain is an NT domain and user is an NT user name. The user name is returned. If there is no backslash in the string, it is returned unchanged.

Operation	Arguments	Return Type	Definition
__hexStringToAddress	token_name	IP Address	The argument is in hexadecimal. That is, it should be 8 hexadecimal digits, where each set of 2 digits is a part of the IP address, zero-filled, and with no dots. For example, “C0A80A0C” would become the IPv4 address 192.168.10.12.
__ifThenElse	token_name1, expression1|expression2|token_name2, expression3|expression4|token_name3|null, expression5|expression6|token_name4|null (See Note42 below)	String	There are four arguments. Each can be either a literal string or a regular string (although other types are converted to strings). The first two arguments are compared, and if they are equal, then the third argument is returned as the result. Otherwise (if the first two arguments differ), the fourth argument is returned.
__oneOf	token_name1, token_name2, token_name3, expression1...	String	This operation takes an arbitrary number of arguments. Each can be either a literal string or a regular string. The first one that is not null and not zero-length is returned.
__oneOfAddress	token_name1, token_name2, token_name3, expression1...	IP Address	For non-IPv6-aware parsers, this operation returns only the first non-null IPv4 address. For IPv6-aware parsers, this operation returns the first non-null IPv4 or IPv6 address.
__parseMultipleTimeStamp	token_name1, token_name2,...	TimeStamp	The first argument is a timestamp value, passed as a string. If it is null, null is returned. Otherwise, the second and any additional arguments are constant time stamp formats (as defined for Java's SimpleDateFormat class). They are used to attempt to parse the first argument. The result of the first one that works, without throwing an exception, is returned as a TimeStamp. If none of the formats work, an exception is thrown.

Operation	Arguments	Return Type	Definition
__regexToken	regex_expression1, regex_ expression2	String	The operation takes two strings as arguments. The first is the string to parse. The second is the regular expression (a literal string). If the regular expression is blank or null then the result is the same as the first argument. Otherwise the string to parse is parsed using the regular expression, and the first matching group (expression inside parentheses) is returned as a string. For example, if the arguments are “foobar” and “fo+(o.*)(r)”, the result will be “oba”. Example: `proto,".*?/(.*)"`
__regexTokenAsAddress	regex_expression1, regex_ expression2	IP Address	Similar to the regexToken operation, also taking 2 string arguments, except that the result (expected to be in 4-part dotted decimal format) is then converted from a string to an IP address. That is, if the arguments are “foo/192.168.10.12/bar” and “[a-z]+\/([0- 9\.]+)\/bar”, the result will be the IP address 192.168.10.12. For example: `dst,"(.*?)[: ].*"`
__regexTokenNoWarning	regex_expression1, regex_ expression2	String	Similar to the regexToken operation. The primary differences are that • the regular expression must match the entire string, not just be found in it, and • if the regular expression does not match, no warning is logged.

Operation	Arguments	Return Type	Definition

__safeToInteger	token_name	Integer	The argument is a single string, which is converted to an integer, or null if the string is not a valid number. Useful for log formats that use "-" to specify null values on integer fields, such as Microsoft Windows XP SP2 Personal Firewall. Examples:
			`Bytes`
			`srcPort`

__safeToLong	token_name	Long	The argument is a single string, which is converted to a long integer, or null if the string is not a valid number. Example: `time_taken`
__stringConstant	constant	String	This operator takes a single string literal argument and returns it. Example: `"Example"`
__stringToIPv6Address	token_name	IP Address	In a non-IPv6-aware parser, this operation takes a string representation of an IPv6 address as input and returns a value of type IPv6 address. This operation should not be used in a IPv6-aware parser. Instead, use the IP Address token parser or directly map the IPv6 address string to event fields.
__stringTrim	token_name	String	The argument is a string, that is returned with any leading or trailing whitespace characters removed.

Note:

1. For the __ifThenElse operation, you can substitute any of the following for operation: token_name|"constant"|operation|regex_expression|null.

Send Documentation Feedback

If you have comments about this document, you can contact the documentation team by email. If an email client is configured on this system, click the link above and an email window opens with the following information in the subject line:

Feedback on Online Help (Quick Flex Parser Tool 1.0)

Just add your feedback to the email and click send.

If no email client is available, copy the information above to a new message in a web mail client, and send your feedback to arc-doc@hpe.com.

We appreciate your feedback!