puresecurity™

Check Point SOFTWARE TECHNOLOGIES LTD.

Eventia Log Parsing Editor 1.0 Administration Guide

Revised: November 28, 2007

In This Document

Overview

In This Section

What is Eventia Log Parsing Editor?

Eventia Log Parsing Editor is a tool that lets an administrator easily and intuitively define instructions for a Check Point log server to convert third-party device syslogs to standard Check Point logs. This conversion is crucial for Check Point's Eventia Analyzer to be able to identify security events occurring in third-party devices.

For general information on Eventia Analyzer's third-party support, see "Third Party Device Support" in the *Eventia Analyzer Administration Guide*.

How Syslog Parsing Works

Syslogs differ from Check Point logs not only in their specific values, but also structurally. Syslogs are long strings of values, in different patterns. Check Point logs, on the other hand, are clearly defined pairs of field names and values.

Because of this difference, the Check Point log server needs to be instructed not only how to translate syslog values to Check Point values, but first, how to identify the sections of the syslog. Identifying syslog sections enables meaningful instructions for translating the sections' values to appropriate values for specific Check Point log fields.

For example, a syslog might contain the following:

192.168.248.7 192.168.247.5 udp denied

If the sections are the source and destination addresses, the protocol and the device's resulting action, the appropriate Check Point fields and values should be:

Source	192.168.248.7
Destination	192.168.247.5
Protocol	17
Action	reject

To achieve this translation, the log server needs to be instructed that the first section's value should be copied to the Source field; the second to the Destination field; the third section (udp) needs to be translated to a particular number for that protocol (17), and placed in the Protocol field; and the fourth section (denied) needs to be translated to a different particular word (reject) and placed in the Action field.

A particular device may produce logs with different patterns. The log server needs to check each log against a number of defined patterns, and to parse the log according to the instructions for the matched pattern.

Your task is to use Eventia Log Parsing Editor to apply section definitions to a representative cross-section of syslogs from your third-party device. These definitions enable Eventia Log Parsing Editor to consolidate similarly structured syslogs into patterns. The definitions should be wide enough that all your device's syslogs will be matched.

After the syslog patterns are fully defined, you set for each pattern the rules for translation to Check Point logs.

Once you have finished configuring the translation rules in Eventia Log Parsing Editor, you use Eventia Log Parsing Editor to compile those instructions for installation on the Check Point log server. The compiled instructions will include the various patterns that the device's syslogs could conform to, definitions of each pattern's sections, and appropriate translations to Check Point log fields. Then, Eventia Analyzer will be able to analyze the Check Point logs and detect events that have occurred in your third-party device.

The procedure for creating parsing instructions is discussed in "Creating Parsing Instructions" on page 10.

Managing Parsing Instruction Sets

When you create a parsing instruction set in Eventia Log Parsing Editor, it is in the form of a Parsing Project. You then use the Parsing Project to generate a Parsing File, and, in some cases, a dictionary file. You install the parsing file and dictionary file on the Check Point log server, for the log server to follow the files' parsing instructions.

Eventia Log Parsing Editor can use a saved parsing project as a basis for changes, but cannot open an existing parsing file. Therefore, you should always save your parsing projects for later modification. You will be able to change the instructions in the project, and to add log samples to the project to define additional syslog patterns.

Installation and Supported Platforms

Eventia Log Parsing Editor does not need to run on the same computer as the log server.

Eventia Log Parsing Editor is supported on Windows versions 98, 2000 SP3, ME, Server 2003, XP SP2, Vista.

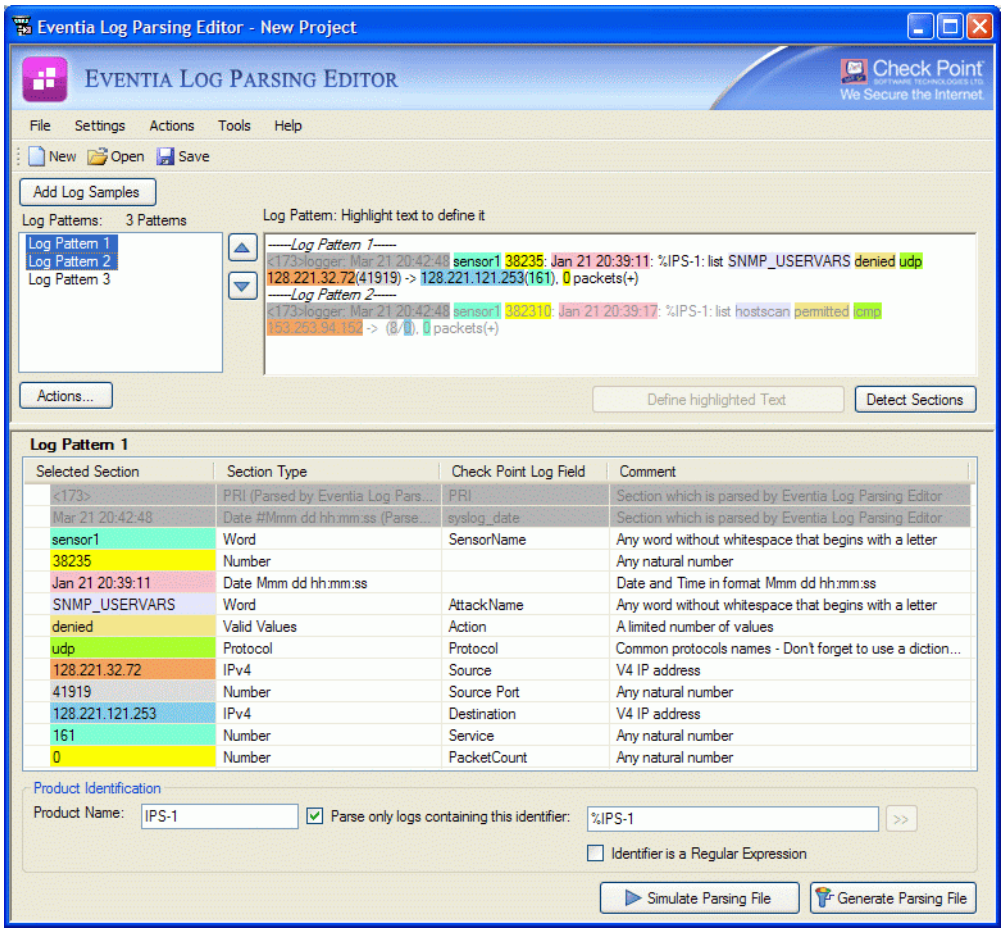
To install Eventia Log Parsing Editor, run the provided setup executable, and follow instructions.

Menus and Main Window

In This Section

Main Window

Parsing instructions are created in Eventia Log Parsing Editor's main window:



In This Section

Main Window Areas

The main window has four main areas:

- **Log Pattern List** - the upper left area of the main window lists the log patterns found in the imported syslog sample.

  As pattern sections are defined, patterns will be consolidated. You can rename a pattern by right-clicking on it. The **Actions** button below the Log Pattern List contains commands for patterns.

- **Log Sample Viewer** - the upper right area of the main window displays the full text of a sample syslog of the pattern or patterns selected in the Log Pattern List. Sections that are highlighted in color have definitions, and therefore appear in the Pattern Details Viewer.
- **Pattern Details Viewer** - the lower part of the main window lists:
  - The defined sections from the sample syslog.

  and, for each defined section:
  - The **Section Type**, that is, its definition.
  - The **Check Point Log Field** mapped to the section.
  - A user's **Comment**. The default comments relate to the section types.
- **Product Identification** - the bottom of the window contains fields for device information. See "Creating Parsing Instructions" on page 10. The fields are:
  - The **Product Name** of the device generating the logs. This name will appear in the generated Check Point logs, in the Product field.
  - **Parse only logs containing this identifier** - to enhance log server performance, select this option, and type a string section unique to syslogs produced by the specific device. This will prevent the log server from checking these patterns against syslogs from other devices.

Buttons in the Main Window

The main window also contains the following buttons:

- **Get Log Samples** / **Add Log Samples** - imports sample syslogs for pattern definition. See "Creating Parsing Instructions" on page 10.
- **Actions** - opens a menu of the following pattern actions:
  - **Compare Patterns** - enables viewing sample syslogs of multiple patterns in the Log Sample Viewer.
  - **Combine Duplicate Patterns** - compares all patterns and combines similarly defined patterns into one pattern.

    Generally, Eventia Log Parsing Editor automatically detects pattern similarities. However, **Combine Duplicate Patterns** is useful, in case two patterns have become similar, not as a result of a change made directly to either one of them, but as a result of a change automatically applied (if Automatically Apply Section Changes to All Patterns is enabled) to one or both of them, because of a change made to a third pattern.
  - **Remove Selected Patterns** - removes, from the parsing sample, patterns that are selected in the Log Pattern List.
- **Define Highlighted Text** / **Edit Section** - opens properties window for section definition. See "Creating Parsing Instructions" on page 10.
- **Detect Sections** - defines known section types, such as protocols, dates, and IP addresses.
- **Simulate Parsing File** - simulates the results of the parsing file that would be generated from the current parsing project. See "Parsing File Simulation" on page 17.
- **Generate Parsing File** - generates parsing file, and, if mapping is used, a dictionary file. These files must then be installed on the log server. See "Creating Parsing Instructions" on page 10 and "File Installation" on page 19.

Menu Commands

In This Section

File Menu

From the file menu you can start **New** projects, or **Open** and **Save** existing ones.

Regarding projects, see "Managing Parsing Instruction Sets" on page 3.

Settings Menu

The Settings menu contains toggle options that can be enabled or disabled. The options are:

- **Automatically Detect Section Types** - when enabled, upon importing syslog samples, Eventia Log Parsing Editor automatically defines known section types, such as protocols and IP addresses. To have Eventia Log Parsing Editor detect section types later on, use the **Detect Sections** button.
- **Automatically Apply Section Changes to All Patterns** - when enabled, when a section change is made, Eventia Log Parsing Editor automatically applies section changes to all patterns that, before the changed section, are defined in the same way as the changed pattern, and that in the changed section's position have a section to which the change can be applied.
- **Replace Spaces and Tabs with Whitespace** - when enabled, Eventia Log Parsing Editor treats spaces and tabs in syslog text as whitespace. The effect of this is that multiple consecutive spaces and/or tabs are equivalent to a single space.
- **Enable Fixed and Conditional Values** - enables adding a Fixed or Conditional Value to the end of a pattern (or a group). See "Mapping a Syslog Section to Multiple Check Point Log Fields" on page 14 and "Setting a Fixed Check Point Log Field Value for a Syslog Pattern" on page 15.
- **Enable Division into Section Groups** - enables dividing a pattern into groups, by right-clicking on a section in the Pattern Details Viewer and selecting **Begin New Group**. The effect of grouping is that if a part of a syslog matches a group, it will be parsed even if subsequent parts are not matched by subsequent groups.

In addition, the Settings menu contains the following command:

- **Enable Warning Messages** - re-enables warning messages for which **Do not show this message again** has been previously selected.

Actions Menu

The Actions menu contains the following commands:

- **Generate Parsing File** - generates parsing file, and, if mapping is used, a dictionary file. These files must then be installed on the log server. See "Creating Parsing Instructions" on page 10 and "File Installation" on page 19.
- **Simulate Parsing File** - simulates the results of the parsing file that would be generated from the current parsing project. See "Parsing File Simulation" on page 17.

Tools Menu

The Tools menu contains the following commands:

- **Mapping Manager** - opens the mapping manager, where you can define and edit mapping for value translation from syslog sections to Check Point log fields. See "Mapping a Syslog Section to a Single Check Point Log Field" on page 14.
- **Conditional Values Manager** - opens the Conditional Values manager, where you can define and edit Conditional Values. A Conditional Value defines a source field, a target field, and value translations, for mapping one Check Point log field to another. See "Mapping a Syslog Section to Multiple Check Point Log Fields" on page 14.
- **Check Point Log Fields Manager** - opens the Check Point log fields manager, where you can define custom fields for Check Point logs.

Help Menu

The Help menu contains the following commands:

- **Help Topics** - opens an HTML version of this manual.

  In any Eventia Log Parsing Editor window, you can press F1 or click **Help** for context-sensitive help.
- **About** - displays version information about Eventia Log Parsing Editor.

Context (Right-Click) Menus

Log Pattern List

Right-clicking a pattern in the Log Pattern List produces the following commands:

- **Apply Sections to All Patterns** - applies the current pattern's definitions to all other patterns, where relevant.
- **Rename** - renames the pattern.
- **Remove** - removes the syslogs of the selected pattern from the project sample.

Pattern Details Viewer

Right-clicking a section row in the Pattern Details Viewer produces the following commands:

- **Edit** - opens the section's properties window for section definition.
- **Remove** - removes the section's definitions, and accordingly removes the section from the Pattern Details Viewer.
- **Begin New Group** (available only when **Enable Division into Section Groups** is enabled in the Settings menu) - defines the section as the first of a new group. See "Combining Inclusive Syslog Patterns" on page 13.

Right-clicking **Added Values** (available when **Enable Fixed and Conditional Values** is enabled in the Settings menu) in the Pattern Details Viewer produces the following two commands:

- **Add Fixed Value** - opens the Add Fixed Value window, to add a fixed value to a Check Point log field. See "Setting a Fixed Check Point Log Field Value for a Syslog Pattern" on page 15.
- **Add Conditional Value** - opens the Add Conditional Value window to set a secondary mapping of one Check Point log field to another. See "Mapping a Syslog Section to Multiple Check Point Log Fields" on page 14.

Creating Parsing Instructions

This section describes the high-level workflow for creating Parsing Instructions with Eventia Log Parsing Editor. Some of the steps in this section are discussed in more detail in following sections.

To create Parsing Instructions with Eventia Log Parsing Editor:

1. From the **File** menu, start a **New** project or **Open** an existing one.
2. Click **Get Log Samples** or **Add Log Samples** to import sample syslogs for parsing.
3. In the **Product Identification** area of the main window:
   a. Type the **Product Name** for the device generating the logs. This name will appear in the generated Check Point logs, in the Product field.
   b. Optional: to enhance log server performance, select **Parse only logs containing this identifier** and type a string section unique to syslogs produced by the specific device. This will prevent the log server from checking the current patterns against syslogs from other devices. In addition, any syslog containing the product identifier will generate a Check Point log with the Product field set to the Product Name, even if the syslog does not match any parsing pattern.

      If identification of the device for all of its syslogs is not possible by a unique string, you can select **Identifier is a Regular Expression** and type a regular expression. Make sure the expression matches sections from all the device's syslogs and is unique to the device's syslogs.
4. Define syslog patterns to make the syslogs identifiable and meaningful for translation. Details of this process are in "Syslog Patterns" on page 11.
5. To enhance log server performance, when you have defined all of the syslog patterns, arrange the patterns' order from most-occurring to least-occurring, according to your expectations from the device. The log server will attempt to match every syslog to each pattern in the order of this list, until it finds a match.

   To change the patterns' order, use the up and down arrows next to the Log Patterns List.
6. Set rules for value translation to Check Point log fields. Details of this process are in "Setting Rules for Syslog Value Translation to Check Point Logs" on page 14.
7. You can simulate the parsing file that your parsing project would generate. Simulation can test the validity of your parsing definitions for a wider sample of syslogs, and the results of your translation instructions. For details, see "Parsing File Simulation" on page 17.
8. From the **File** menu, **Save** your project. The project can serve as a basis for later modification.
9. To generate and save a parsing file, click **Generate Parsing File**. In the resulting confirmation window, you can choose to generate a parsing file from specified syslog patterns, and whether to add a sample syslog of each pattern to the text of the generated file, as a comment.

   In addition to the parsing file, if mapping is used in the parsing instructions, a dictionary file is generated and saved in the same folder as the parsing file.
10. Install the parsing file and the dictionary file on the log server. For details of this process, see "File Installation" on page 19.

Syslogs received by the log server will now be parsed and translated according to your instructions.

Syslog Patterns

In This Section

Overview

The Check Point log server needs to be instructed how to identify a syslog pattern, and, accordingly, the specific sections of the syslog. Identifying syslog sections enables meaningful instructions for translating the sections' values to appropriate values for specific Check Point log fields.

The log server can identify a syslog section in a meaningful and efficient way only if the section is defined properly. Proper definitions include all possible instances of that section that should signify the same type of information that can be converted to Check Point log fields according to the same rules.

For example, the part of a syslog which contains a date should be defined in such a way that the definition covers all possible dates, but so that other types of information are not included in the definition. Most standard types of definitions (such as standard date and time formats) are available for simple application to syslog sections. You may have to customize other definitions.

Each syslog section's definition includes its location as immediately following the specific previous section. So, sections of different syslogs are recognized as identically defined only if they follow identically defined section sequences. For this reason, the log server must be able to identify the syslog's sequential pattern of sections in order to identify the syslog's sections.

A particular device may produce logs with different patterns. The log server checks each log against a number of defined patterns, and translates the log according to the instructions for the matched pattern.

Your task is to use Eventia Log Parsing Editor to apply section definitions to a representative cross-section of syslogs from your third-party device. These definitions enable Eventia Log Parsing Editor to consolidate similarly structured syslogs into patterns. You define your syslogs so that all your device's syslogs will be matched, by as few patterns as possible.

Minimizing the number of patterns necessary to recognize all of a device's syslogs simplifies managing the definitions and improves log server performance.

Defining Syslog Patterns

Defining syslog patterns is one stage of the process of creating parsing instructions. For the whole process, see "Creating Parsing Instructions" on page 10.

Before you define the syslog patterns, make sure the following are all enabled in the **Settings** menu:

- **Automatically Detect Section Types**
- **Automatically Apply Section Changes to all Patterns**
- **Replace Spaces and Tabs with Whitespace**

To define syslog patterns, for each pattern in the Log Patterns List, perform the following:

1. In the Log Patterns List, select the log pattern.

A sample syslog appears to the right, in the Log Sample Viewer.

The first part of the log sample, the syslog prefix, is highlighted in gray, and is defined by the system. It does not need to be manually defined.

Sections that are highlighted in other colors have been automatically identified, but you can and should check the definitions, in the following steps.

2. Starting in the syslog immediately after the gray-highlighted section, and continuing in order of the syslog, define each section of the sample log, as follows:

   a. If the section is not meaningful in itself, and is a fixed string that does not have alternative values, it does not need to be defined, and log server performance will be improved by leaving it undefined. In this case, move on to the next section.

   b. Select the section, and click **Define Highlighted Text** or **Edit Section** (depending on whether the text has already been identified or not).

   Note - It may happen that part of the section is already identified, but you want to define a wider section, so that the already identified part is only part of the section. In this case, first remove the part that is already defined. To do this, right-click the section in the detailed log pattern area in the lower part of the window and select **Remove**.

   c. If you are prompted to remove spaces or tab characters from the selection's end, click **Yes**.

   d. Define the **Section Type**.

      You can select a standard definition from the list, or click **More** to customize a regular expression or to view the regular expression for the selected definition. To then use predefined regular expression building blocks, click **>>**. For regular expression syntax details, see "Appendix: Regular Expression Syntax" on page 21.

      In defining the section, follow the following guidelines:

      - Most standard types of definitions (such as standard date and time formats) are available for simple application to syslog sections. You may have to customize other definitions, with regular expressions. For details on regular expressions, see "Appendix: Regular Expression Syntax" on page 21.
      - Section definitions should be as wide as possible, that is, they should include all possible variations of the field, with the condition that they should be limited to values of similar significance that can be copied or translated to Check Point log fields according to the same rules.
      - To widen the range of values included in a definition, it may be useful to compare different log patterns. To do this, go back to the main window. Below the Log Patterns List, click **Actions** and select **Compare Patterns**.
      - If the section is a fixed string that does not have alternative values, select from the Section Type list **Static Text**.
      - If a section may contain a limited number of specific values, select **Valid Values** from the **Section Type** list. An editable list of possible values appears in the window.
      - If a section may be absent from a syslog, and yet the rest of the syslog can be interpreted according to the same pattern as if it had followed this section, select **Set section as optional ()?** from the regular expression building block list.

   e. Click **OK**.

3. When you have finished defining the entire log sample of a pattern, in the Log Patterns List, right-click the pattern and select **Apply Section Changes to all Patterns**.

4. Below the Log Patterns List, click **Actions** and select **Combine Duplicate Patterns**.

5. **Optional:** For management efficiency and log server performance enhancement, perform the following:

   To compare syslog patterns, below the Log Patterns List, click **Actions** and select **Compare Patterns**.

   Check if another pattern is now fully defined and is included in the pattern you just defined, being identical to its beginning. In other words, the two patterns are identical except for the end of the pattern you just defined, which is absent from the other pattern. In this case, follow the instructions in "Combining Inclusive Syslog Patterns" on page 13.

When all syslog patterns are fully defined, set rules for translation to Check Point logs, as discussed in "Setting Rules for Syslog Value Translation to Check Point Logs" on page 14.

Combining Inclusive Syslog Patterns

You can enhance management efficiency and log server performance, if one pattern is included in another pattern, being identical to its beginning. In other words, the two patterns are identical except for the end of the longer pattern, which is absent from the shorter pattern.

In this case, divide the longer pattern into Groups so that the first group is identical to the shorter pattern. The log server will use the first group's instructions for all syslogs that match this group's definitions, regardless of whether they have the longer pattern's extension or not.

To divide the longer pattern into groups:

1. In the **Settings** menu, enable **Enable Division into Section Groups**.
2. In the Log Patterns List, select the longer pattern.
3. In the Pattern Details Viewer, right-click the first section that is absent from the shorter pattern. Select **Begin New Group**.

Note - If the first section that is absent from the shorter pattern is undefined, it must be defined (for example, as static text) in order to be able to begin a new group from that point.

A group title is added above the section.

4. If the shorter pattern is not automatically combined into the longer one, then below the Log Patterns List, click **Actions** and select **Combine Duplicate Patterns**.

Setting Rules for Syslog Value Translation to Check Point Logs

Setting translation rules is one stage of the process of creating parsing instructions. For the whole process, see "Creating Parsing Instructions" on page 10.

You do not have to set translation rules for every syslog section. If a particular section's values are not significant for your purposes, you can decide to leave its translation rules undefined. In this case, the section's values will not be expressed in the Check Point log.

In This Section

Mapping a Syslog Section to a Single Check Point Log Field

To set a syslog section's translations rules to a single Check Point log field:

1. To open the section's Properties window, either select the section in the Log Sample Viewer and click **Edit Section**, or, in the Pattern Details Viewer, right-click the section and select **Edit**.
2. To enable mapping the syslog section to a Check Point log, select **Check Point Log Field** and select a field from the list.

   To define a custom log field, click **Add**. You can open the Check Point Log Fields Manager by clicking **Edit**, or from the **Tools** menu.
3. If the syslog section's values are appropriate for the Check Point log field, in their original form, leave **Use Mapping** clear. If the values need to be converted, select **Use Mapping** and select a mapping from the list.

   To define a custom mapping, click **Add**. To view or edit a mapping's conversion values, click **Edit**. You can also access the Mapping Manager from the **Tools** menu.
4. Click **OK**.

Mapping a Syslog Section to Multiple Check Point Log Fields

For a given syslog pattern, you can map a syslog section's values to more than one Check Point log field, by using Conditional Values.

A Conditional Value maps one Check Point field to another according to defined conversion rules. Source field values thus become the condition according to which target field values are set.

To map a syslog section to multiple Check Point log fields, first set translation rules to one Check Point log field, as detailed in the previous section, "Mapping a Syslog Section to a Single Check Point Log Field" on page 14. This field will be the source field for a Conditional Value. Then, create a Conditional Value for a secondary mapping of the first Check Point log field to each additional Check Point log target field.

The log server applies Conditional Values after setting the source field according to a syslog section's values. So, the syslog section's values indirectly determine the target fields' values.

A rule for converting a source field's particular value to another value for the target field is called a Condition. A Conditional Value can include multiple conditions.

To create and set a Conditional Value:

1. If necessary, create the source and/or Check Point log target fields, as follows:
   a. From the **Tools** menu, select **Check Point Log Fields Manager**.
   b. Click **New**, and type a field name.
   c. Click **OK**, and **Close** the manager.
2. Create a Conditional Value. as follows:
   a. From the **Tools** menu, open the **Conditional Values Manager**.
   b. **Add** a Conditional Value.
   c. Type a **Name** for the Conditional Value, and select **Source** and **Target** Fields for the Conditional Value.
   d. **Add** Conditions.

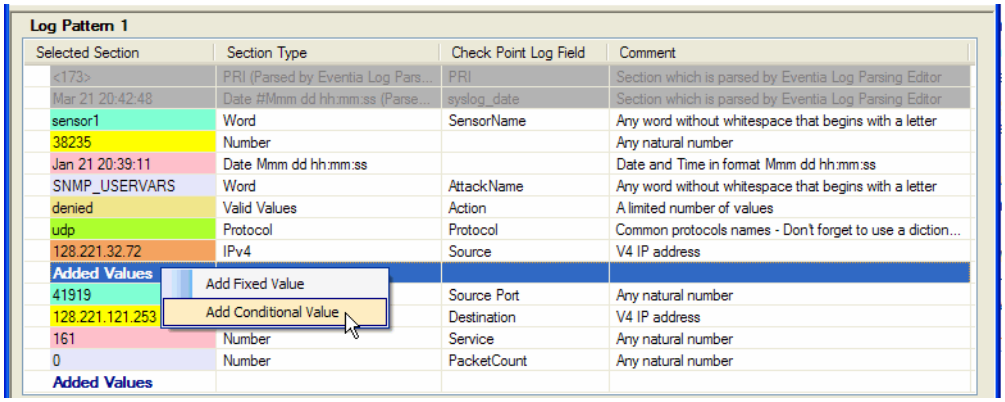
      You can set target values for specific source values, or select **Default** to define a target value for all other source values. You can enter more than one source field value at a time, separated by commas.

      For each entered Condition, click **OK**.
   e. Click **OK** to enter the Conditional Value into the Conditional Values Manager. **Save** the changes to the Conditional Values Manager.

Note - Changes to a Conditional Value are not saved until the Conditional Values Manager is saved.

3. In the Log Patterns List, select the pattern for which you want to define a Conditional Value.
4. In the Pattern Details Viewer, right-click **Added Values**, and select **Add Conditional Value**.

Note - If the syslog pattern is divided into groups (either because you manually started a new group or because the pattern was long enough that it was automatically divided), add the Conditional Value immediately following the source field's group. Otherwise, it may happen that a particular syslog is matched by the part of the pattern containing the source field, and yet the Conditional Value will not be applied.



5. Select the **Conditional Value Name** from the list, and click **OK**.

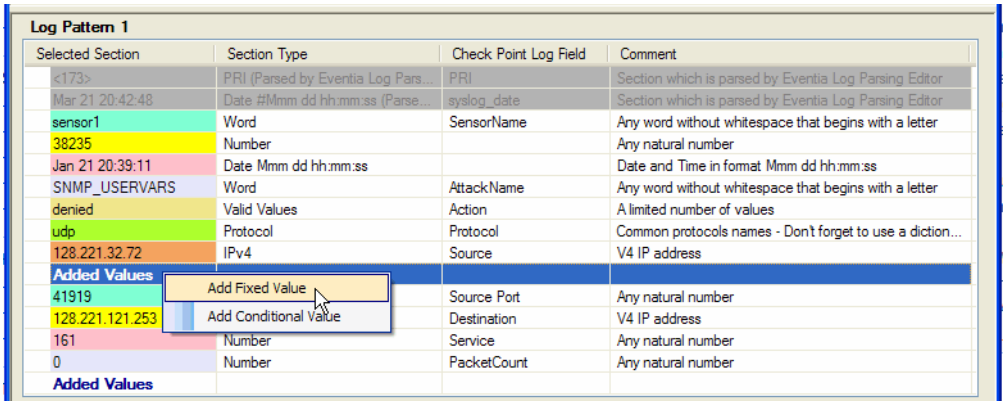
Setting a Fixed Check Point Log Field Value for a Syslog Pattern

You can cause a generated Check Point log to contain a fixed value to indicate that the Check Point log is based on a particular syslog pattern. You can set this value to be placed in a customized field.

To set a fixed value:

1. If necessary, create the Check Point log field to contain the fixed value, as follows:
   a. From the **Tools** menu, select **Check Point Log Fields Manager**.
   b. Click **New**, and type a field name.
   c. Click **OK**, and **Close** the manager.
2. In the Log Patterns List, select the pattern for which you want to define a fixed Value.
3. In the Pattern Details Viewer, right-click **Added Values**, and select **Add Fixed Value**.

Note - If the syslog pattern is divided into groups (either because you manually started a new group or because the pattern was long enough that it was automatically divided), you can add the Fixed Value to the end of any group.



4. Select the **Check Point Log Field** from the list, and type a **Field Value**.
5. Click **OK**.

Parsing File Simulation

You can simulate the parsing file that your parsing project would generate, showing the results that the parsing file would produce when applied to sample syslogs. Simulation tests the validity of your parsing definitions for a wider sample of syslogs, and the results of your translation instructions.

In This Section

Simulating a Parsing File

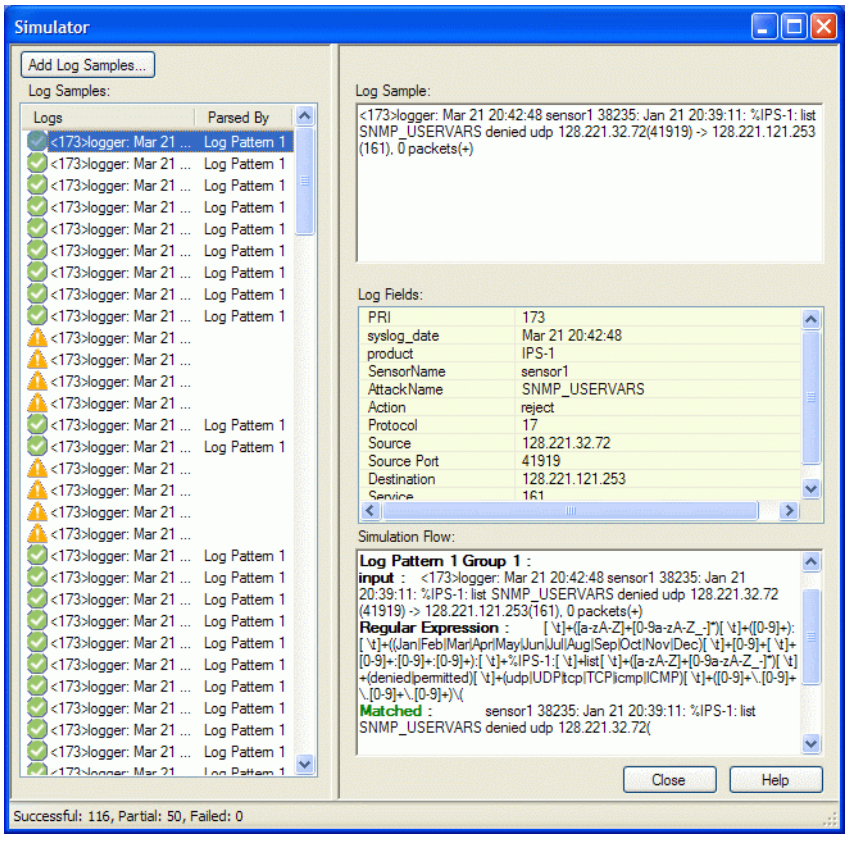
To simulate a the parsing file for a particular parsing project:

1. If the parsing project is not yet open, then from the **File** menu, **Open** the project.
2. At the bottom of the main window, click **Simulate Parsing File**.
3. Select **Simulate All Patterns**, or choose which parsing patterns should be used.
4. Click **Simulate**.
5. In the Simulator window, Click **Add Log Samples** to import syslogs to be parsed and translated.

The simulation results appear in the Simulator window. For an explanation of the simulation results, see the next section, "Simulation Results" on page 17.

Simulation Results

In the Log Samples pane of the Simulator window (on the left), select a syslog from the list.



The selected syslog's full text appears in the **Log Sample** pane.

The **Log Fields** pane displays a list of the generated Check Point log's fields, and the values applied to them.

The **Simulation Flow** pane includes:

- **Log Parsed Successfully**, with the matched log pattern; or, **Failed to parse sample**.
- For successfully parsed syslogs, for each group in the pattern (excluding the syslog prefix):
  - **input** - the part of the syslog being tested against the group's definitions. This input excludes parts of the syslog already matched by previous groups, and includes all parts of the syslog yet to be checked.
  - **Regular Expression** - the group's definitions, in regular expression form.
  - **Matched** - the part of the syslog matched by the group's definitions; or, **Failed to Parse**, if only previous groups were matched.

File Installation

Installing the Log Server File Utility

In order to be able to install or remove parser files and dictionary files on the log server, you must first install the Log Server File Utility on the log server.

The Log Server File Utility is supplied as a file called `addParsingFile`, and comes in different versions according to the Log Server's operating system. Take the utility file from Eventia Log Parsing Editor's installation directory, under the directory named like your Log Server's operating system. For example, if your Log Server runs on Windows Server, and you installed Eventia Log Parsing Editor in the default location, take:

```
C:\Program Files\Eventia Log Parsing Editor\Windows\addParsingFile.exe
```

To install it on the log server, save it on the log server. If the log server is on a UNIX / LINUX platform, such as SecurePlatform or SOLARIS, save to:

```
$FWDIR/bin
```

If the log server is on a Windows platform, save to:

```
%FWDIR%\bin
```

Installing the Parsing file and Dictionary File on the Log Server

This procedure is the final step in creating parsing instructions for the log server. For the full process, see "Creating Parsing Instructions" on page 10.

To install a parsing file, and, if relevant, a dictionary file, on the log server:

1. Make sure the Log Server File Utility is installed on the log server. See the previous section, "Installing the Log Server File Utility".
2. Copy the parsing file, and, if relevant, the dictionary file, to the log server.
3. Run:

   ```
   addParsingFile -p <ParsingFile> [-d <DictionaryFile>]
   ```

   `<ParsingFile>` and `<DictionaryFile>` can be paths to the respective files.

The files are now installed on the log server, and received syslogs will be parsed and translated according to the files' instructions.

Log Server File Utility (addParsingFile) Syntax

When run without arguments, `addParsingFile` displays usage instructions.

The full command syntax is:

```
addParsingFile [-p <ParsingFile>] [-d <DictionaryFile>] [-rp <ParsingFile>] [-rd <DictionaryFile>]
```

The above parameters are:

Table 1

Parameter	Description
-p <ParsingFile>	Installs <ParsingFile>. Overwrites existing file of same name.
-d <DictionaryFile>	Installs <DictionaryFile>. Overwrites existing file of same name.
-rp <ParsingFile>	Removes <ParsingFile>.
-rd <DictionaryFile>	Removes <DictionaryFile>.

<ParsingFile> and <DictionaryFile> can be paths to the respective files.

Appendix: Regular Expression Syntax

Eventia Log Parsing Editor uses Regular Expression V8 regexp(3). This section discusses syntax details.

In This Section

Building Blocks and Definitions

An *atom* is one of the following:

Table 2

Atom	Matched by the Atom
A regular expression in parentheses	Any match for the regular expression
A range	See below
.	Any single character
^	The null string marking the beginning of the input string
$	The null string marking the end of the input string
**** followed by a single character	The character
A single character with no other significance	The character

A *range* is a sequence of characters enclosed in brackets: []. Except when special range characters are used (see below), the range matches any single character from the sequence.

To include a literal **]** in the sequence, make it the first character (following a possible **^** - see below).

Special range characters are:

- If the sequence begins with: **^**, the range matches any single character not from the rest of the sequence.
- If two characters in the sequence are separated by: **-**, this is shorthand for the full list of ASCII characters between them. For example, [0-9] matches any decimal digit.

  To include a literal **-** in a range, make it the first or last character.

A *piece* is an atom possibly followed by one of the following symbols:

- * - an atom followed by * matches a sequence of zero or more matches of the atom.
- **+** - an atom followed by **+** matches a sequence of one or more matches of the atom.
- **?** - an atom followed by **?** matches a match of the atom, or the null string.

A *branch* is zero or more consecutive pieces. It matches consecutive matches, in the same order, of the respective pieces.

Every regular expression is zero or more branches, separated by: **|** . The regular expression matches anything that matches one of the branches.

Logical Resolution

- If a regular expression, by the rules above, can match two different parts of the input string, it will match only the one which begins earliest.
- If both begin in the same place but match different lengths, or match the same length in different ways, only one possibility will be considered a match. The match is the first possibility, according to the following priorities:
  - The possibilities in a list of branches are considered from left to right.
  - The possibilities for *****, **+**, and **?** are considered from longest to shortest.
  - Nested constructs are considered from the outermost to the innermost.
  - Consecutive constructs are considered from left to right.
- If there is more than one choice to be made, each choice is subject to the decision on the previous choices.

  For example, **(abla)b*c** could match `abc' in one of two ways. The first choice is between `ab' and `a'; since `ab' is earlier in the expression, and does lead to a successful overall match, it is chosen. Since the `b' is already spoken for, **b*** must match its last possibility - the empty string - because it is subject to the earlier choice.
- In the particular case where no **|** s are present and there is only one *, **+**, or **?**, the decision on where to start the match is the first choice to be made. Therefore, subsequent choices are subject to it even if this leads them to less-preferred alternatives. The net effect is that the longest possible match will be chosen. For example, **ab***, presented with `xabbbby', will match `abbbb'. If **ab*** is tried against `xabyabbbz', it will match `ab' just after `x', due to the begins-earliest rule.

Documentation Feedback

Check Point is engaged in a continuous effort to improve its documentation. Please help us by sending your comments to:

cp_techpub_feedback@checkpoint.com