CHAPTER 16

Configuring Custom Devices

When you define a custom device, you must define a custom device and log parser. The type of log parser is not defined by the custom device type or the log parser templates, but when you define an instance of the custom device type itself. When you are defining an instance of the custom device, you are required to specify the reporting method, which is either SNMP TRAP or SYSLOG. You are prompted to select either SYSLOG or SNMP as the device type. It is this designation that determines what kind of traffic MARS is expecting to receive from the reporting device.

Adding User Defined Log Parser Templates

MARS allows the user to enter any SYSLOG or SNMP device into the network topology, configure it to report data to the MARS and query the data using free-form query.

User needs to specify the incoming data format so that MARS can parse and retrieve session information from arbitrary logs.

Note While the raw message for an event does include the header information, MARS removes the header prior to sending the payload to the custom parser. When writing a parser log template, do not include the header fields.

To add a user-defined log parser template, you must perform the following tasks:

1. Add a custom Device or Application type. See Define a Custom Device/Application Type, page 16-2.
2. Add a log parser template. See Add Parser Log Templates for the Custom Device/Application, page 16-3.
3. Add device with the above custom Device or Application type. See Add Custom Device or Application as Reporting Device, page 16-13.

Until each of these tasks is completed, MARS is unable to parse the logs from the reporting device, even if it is receiving those events.

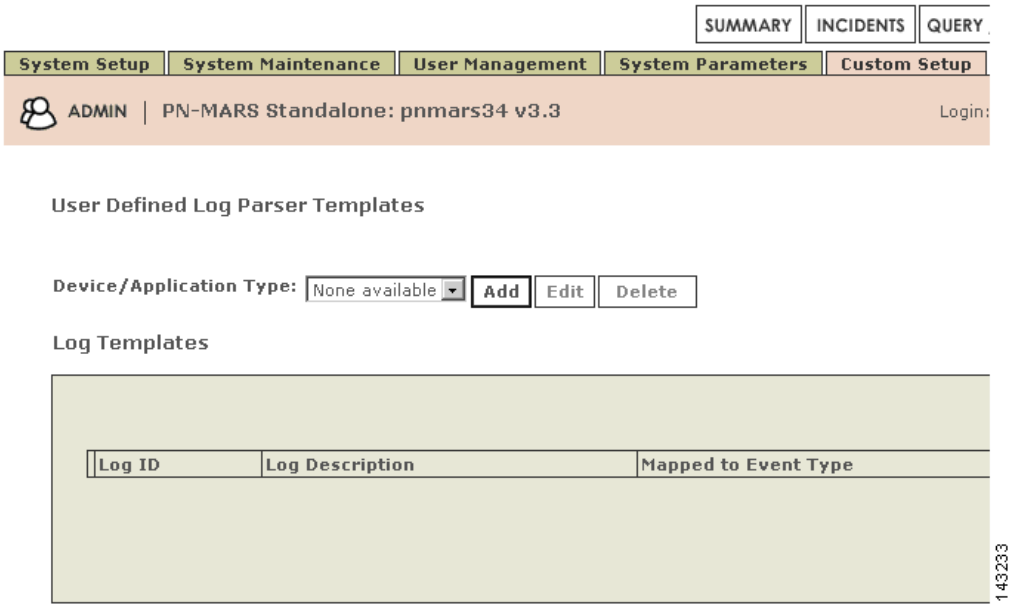
Define a Custom Device/Application Type

The device or application type provides a logical grouping for the custom log templates that you define. MARS uses this logical container to select which custom log templates to apply to traffic received from a reporting device of that device type. You must perform this task before you can begin to define a custom log parser.

Step 1 Go to **Admin >Custom Setup** tab

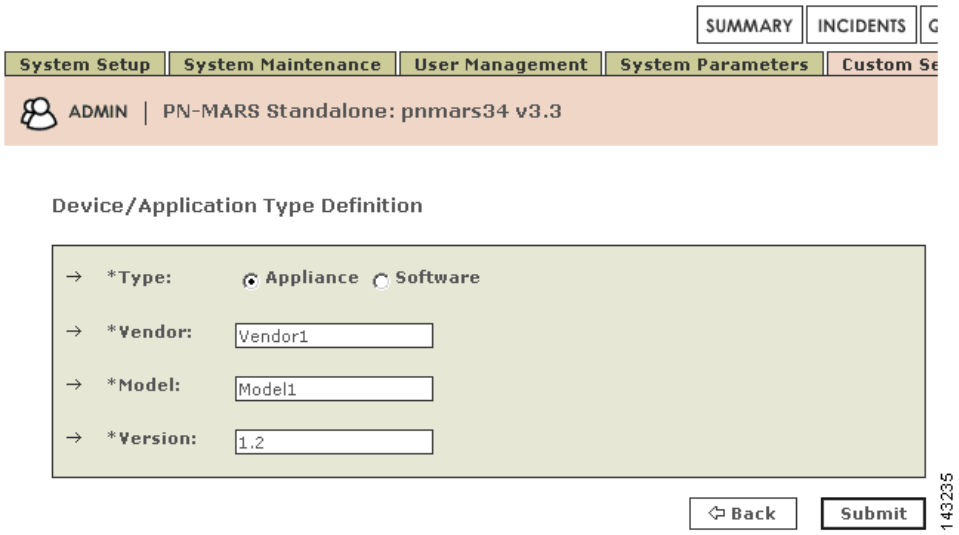
Step 2 Click the **User Defined Log Parser Templates**

Figure 16-1 User Defined Log Parser Template



Step 3 On the next screen, click **Add** button which is located next to the Device/Application type list

Figure 16-2 Device Type Definition



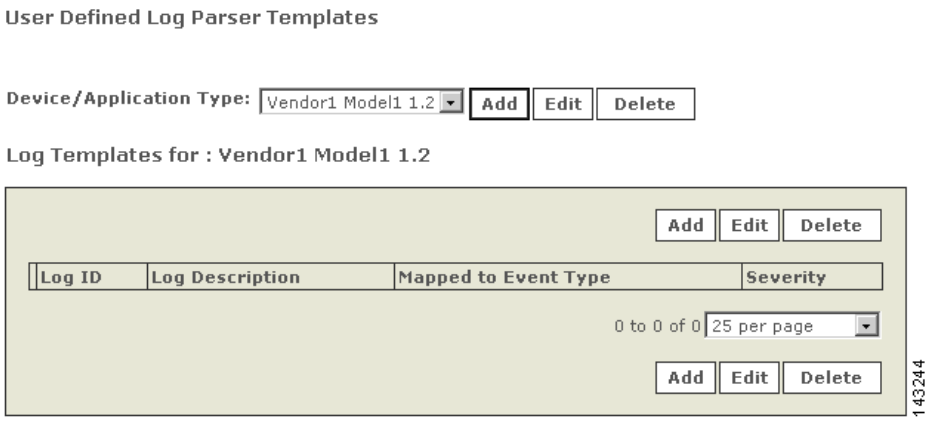
Step 4 Choose the Type - Appliance or Software.

- Appliance - A hardware device that can send logs to the MARS Appliance
- Software - An application running on a host and the host can be configured to send logs to the MARS Appliance

Step 5 Enter the Vendor, Model and Version for the Device or Application. (For Example, Cisco PIX 7.0)

Step 6 Click **Submit**.

Figure 16-3 User Defined Device/Application Type



Add Parser Log Templates for the Custom Device/Application

While the raw message for an event does include the header information, MARS removes the header prior to sending the payload to the custom parser. When writing a parser log template, do not include the header fields.

Note If a log template attempts to parse the header information, the custom parser will fail and indicate that the message is an Unknown Device Event Type. The parser only receives the payload information.

Step 1 Go to the **Admin > Custom Setup** tab.

Step 2 Click the **User Defined Log Parser Templates**.

Step 3 Select the newly created/existing Device or Application from the Device/Application Type list.

Step 4 To add a log template, click **Add** located in the Log Templates for area.

A log template ties directly to the particular message that you want to parse. A log template is composed of one or more Event Types that describe the contents of the message. Using the Event Types, MARS parses the message when it is received.

Step 5 Enter a value in the **Log ID** field. This value is a unique string value that identifies the log message.

The Log ID field provides an opportunity to map this message number or another moniker used by the device to the custom event type that you are developing. You can use this value to clarify the device messages for which you have developed custom event types.

Step 6 Enter **Description** - a description of the log message.

Step 7 Map the log to an Event Type.

Note The MARS Appliance comes with a number of predefined Event Types. To display them, select **System** (for example) from the list above the Event Type select window and click **Get**)

Step 8 New **Event Types** can be added by clicking **Add** below the Event Type list.

Figure 16-4 Mapping Log to Event Type

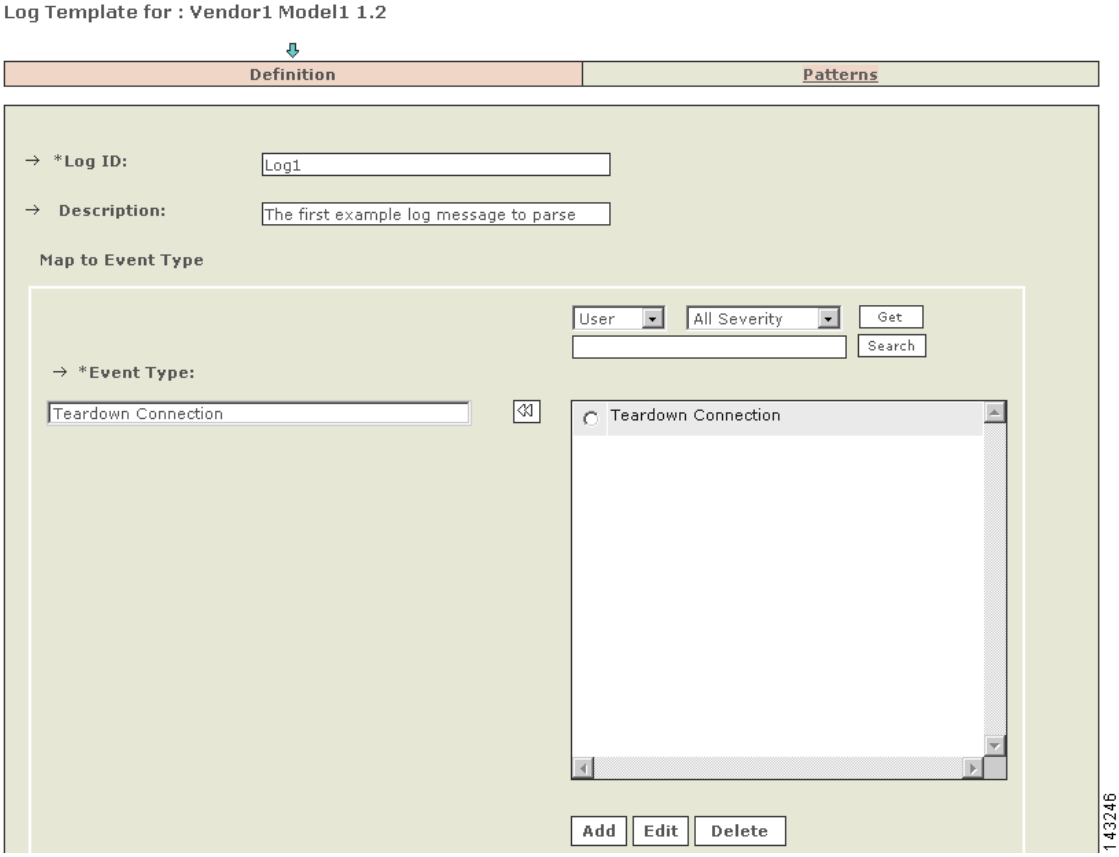
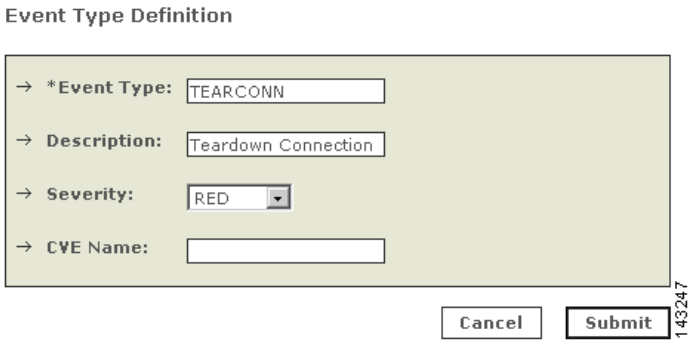


Figure 16-5 Define Event Type

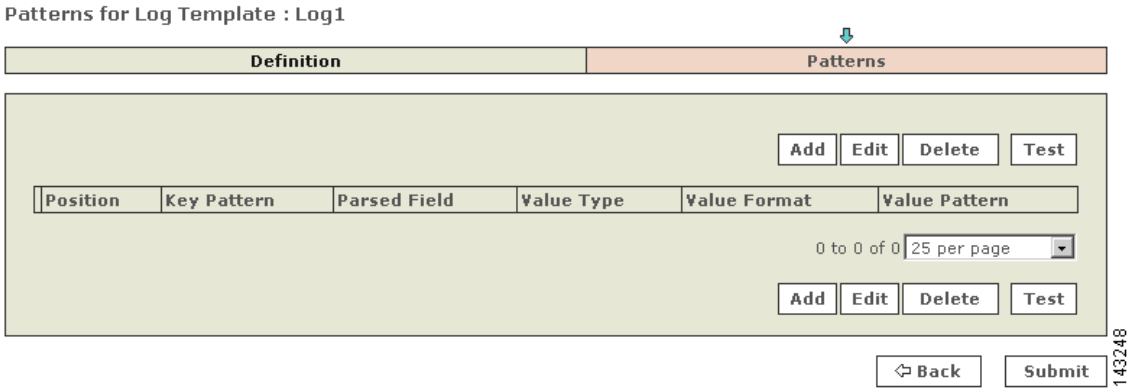


Step 9 Add new Event type and its information and click **Submit (optional)**

Step 10 Click **Apply** - the **Patterns** link will become enabled.

Step 11 Click the **Patterns** link.

Figure 16-6 Define Event Patterns



Step 12 Click **Add** to input patterns.

The parsing patterns for the example above are specified to match the following example raw message reported in an event.

```
Teardown TCP connection 1000 faddr 67.126.151.132/80 gaddr 198.133.219.28/43246 laddr
10.1.1.30/890 (sudha) duration 01:00:02 bytes 1000000 (TCP FINs)
```

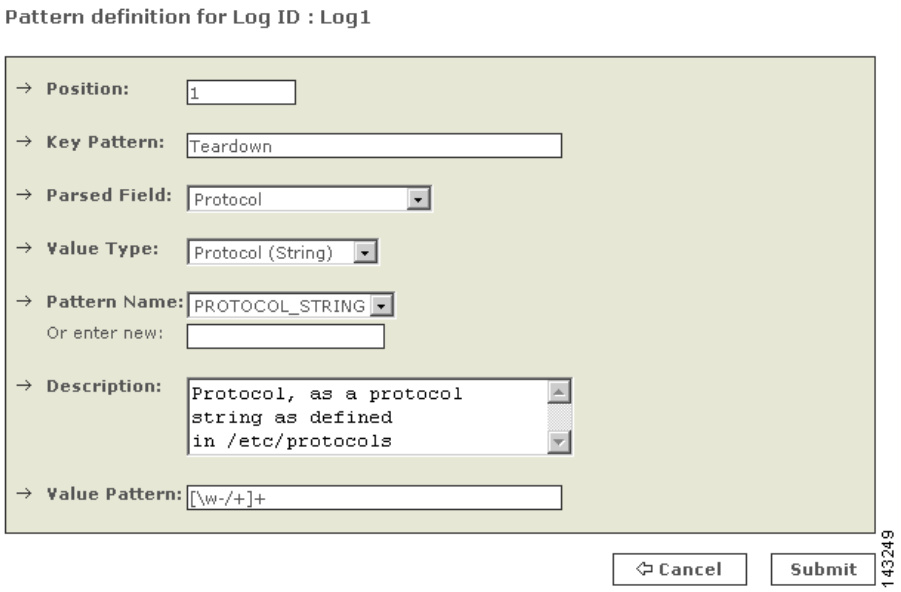
Step 13 The first step is to identify the values in the log that need to be parsed and stored in MARS events.

Step 14 Currently MARS supports the following parsed value fields in its events:

- Source address
- Destination address
- Source port
- Destination Port
- Protocol
- NAT Source address
- NAT Destination address
- NAT Source port
- NAT Destination Port
- NAT Protocol
- Device Time stamp
- Session Duration
- Received Time stamp
- Exchanged Bytes
- Reported User

Step 15 The parsing format can now be thought of as being made up of several KEY pattern followed by VALUE patterns. Both KEY and VALUE patterns are regular expressions based on the library PCRE which is perl-compatible regular expressions (see, Appendix B, "Regular Expression Reference," for details on syntax). Note that a KEY can be an empty string. A log format consists of several KEY-VALUE sub-pattern pairs.

Figure 16-7 Define Pattern for a Log



Step 16 In the above example, the **Position** refers to the position of each KEY-VALUE sub-pattern pair. These KEY-VALUE sub-pattern pairs are concatenated in the order of their positions and used for matching against the raw message in an event. It does allow arbitrary whitespace between KEY and VALUE patterns, as well as between KEY-VALUE sub-patterns.

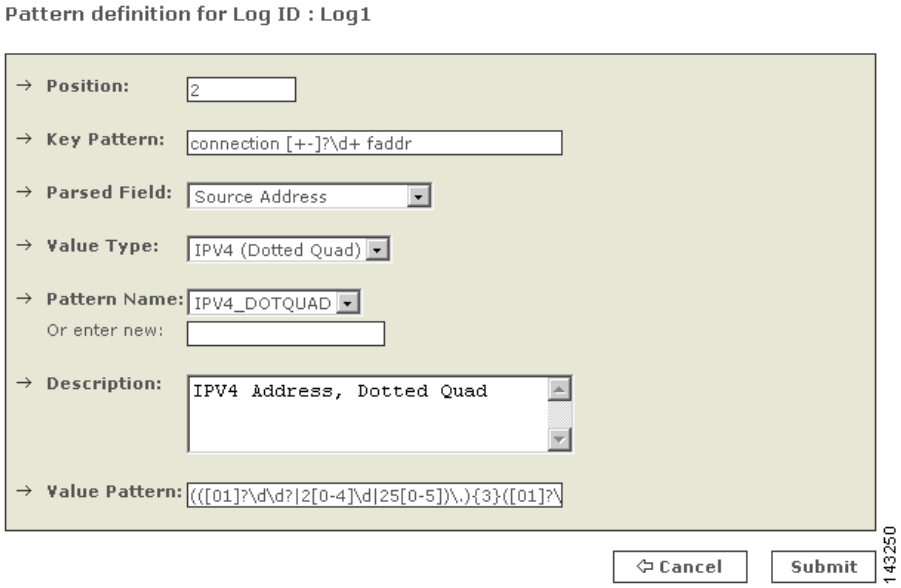
Step 17 In the above example, the **Key-Pattern** is "**Teardown**" is a simple regular expression that does not have any wildcards or repetitions.

Step 18 The **Parsed Field** is one of fields of a MARS event that has been fully parsed. In the above case, it is the protocol field.

Step 19 The **Value Type** gives indication to the parser on what kind of value to expect so that suitable parsing action can be applied on the matching sub-pattern string. By "**Choosing Protocol (String)"** as the value type above, we indicate that the protocol field is coming in the form of a string as defined in the file /etc/protocols in a UNIX system. For example, "**TCP**" is the string that will be captured by the value pattern. The **Value Type** will indicate that TCP is to be converted into its protocol number, 6.

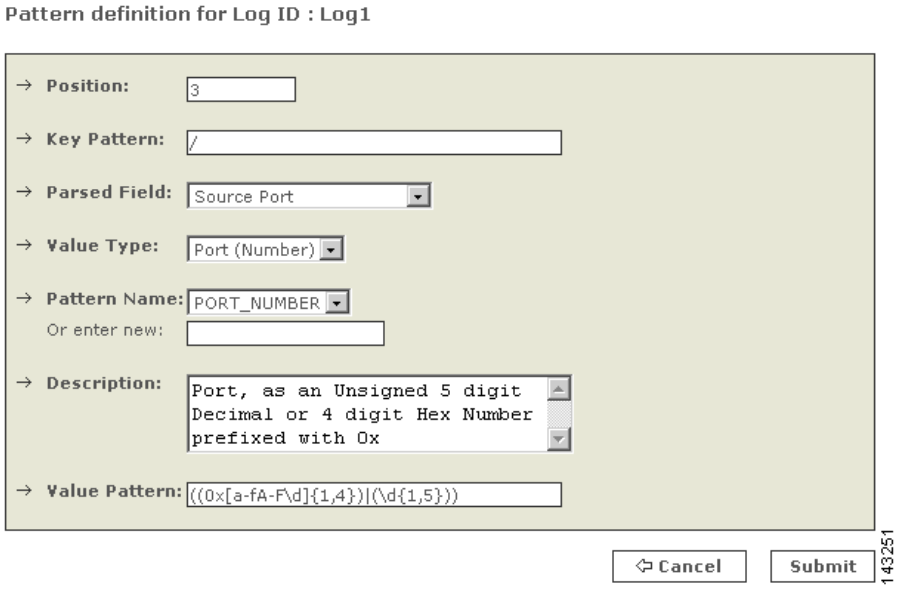
Step 20 **Pattern Name** is a mnemonic given to standard regular expression patterns available for the user who is specifying the log format. There are several common pre defined patterns with appropriate names. In the edit box right below the **Pattern Name** list, a user can add new value names to identify value patterns that may be commonly used in their logs. In the above figure, the value pattern captures all word-character strings that may also include the characters '-', '/' and '+'.

Figure 16-8 Sub Pattern Definition



Step 21 The above KEY-VALUE sub-pattern in the 2nd position of the log format. Notice that the key pattern isn't just a simple string, it is a regexp \d+ which matches against all unsigned decimal numbers. The parsed field where the above value is stored is the Source Address which is specified as a dotted quad. Notice the somewhat complicated regexp that only capture IP addresses in the correct range (0.0.0.0 through 255.255.255.255).

Figure 16-9 Third Position of Pattern Definition



Step 22 The above is for a source port. PORT_NUMBER is the **Pattern Name,** provided for the above **Value Pattern** with the **Description** above.

Step 23 Repeat for every position of **Pattern definition**.

Figure 16-10 The above example is a 12 KEY-VALUE sub-pattern pieces.

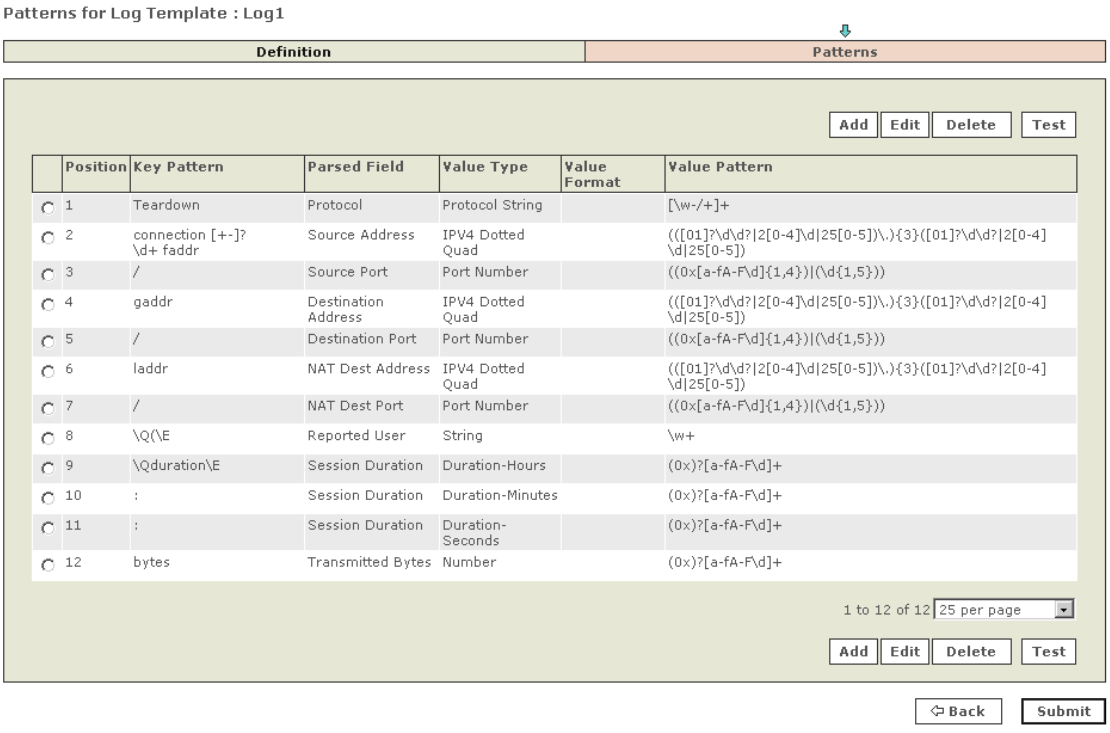


Figure 16-11 Log template for the device type 'Vendor1 Model1 1.2'.

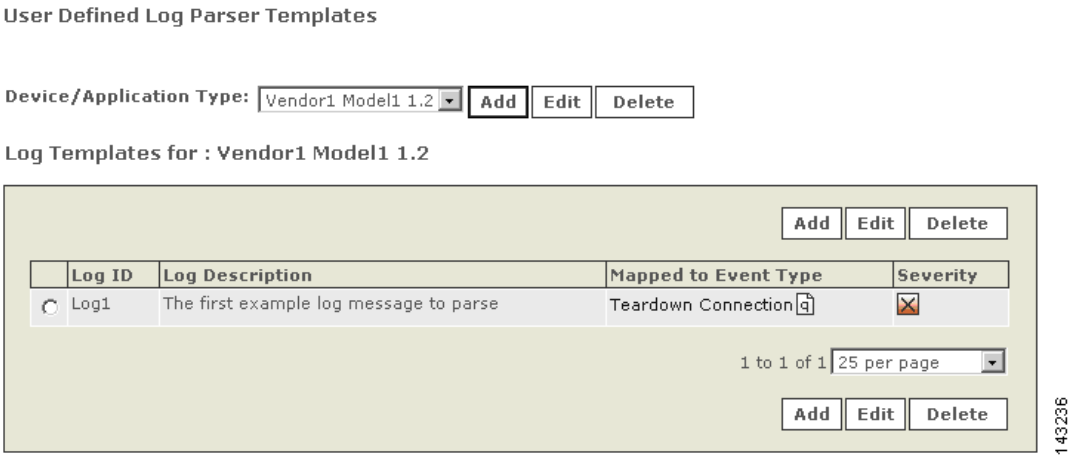


Figure 16-12 New software based Apache Webserver application.

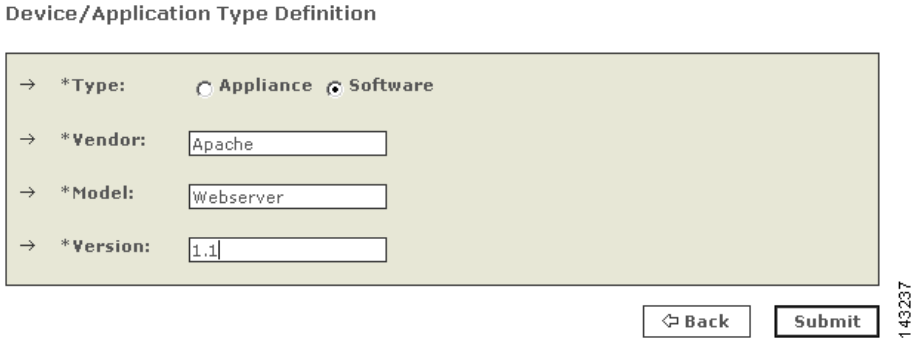
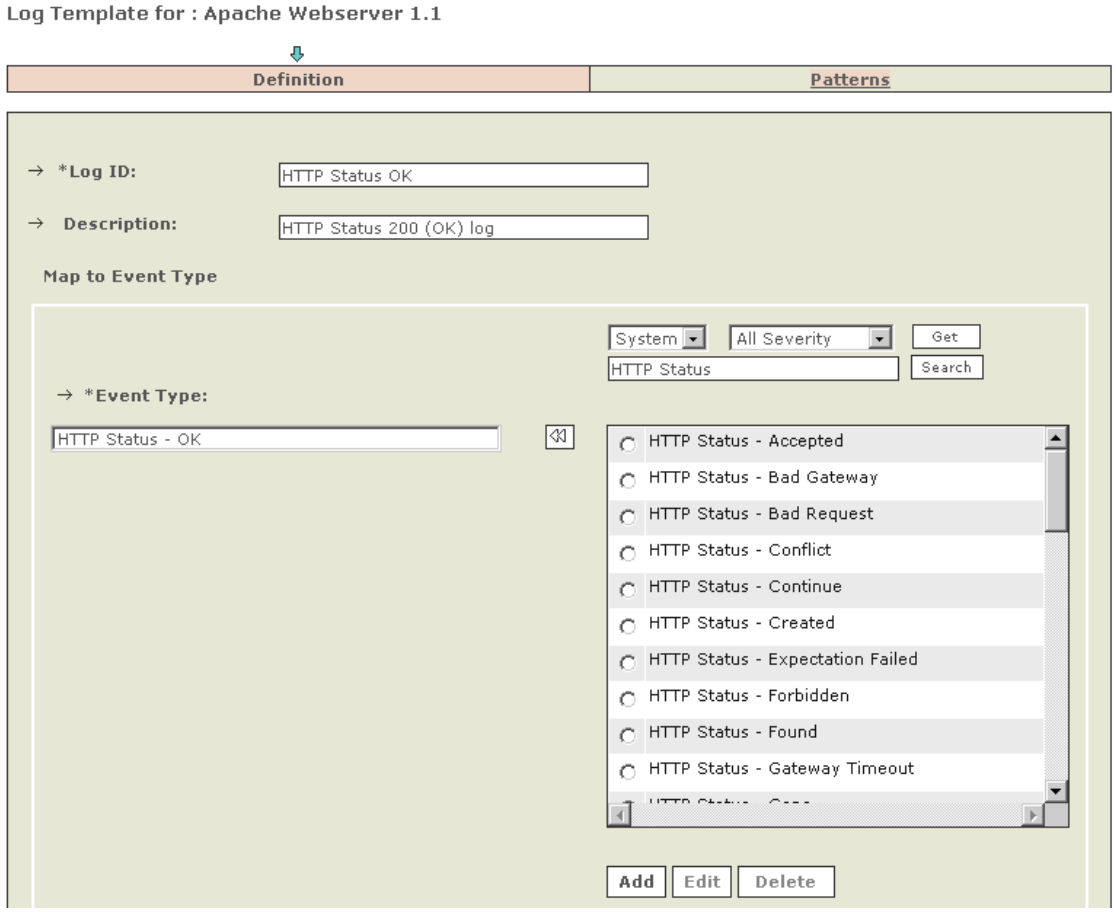


Figure 16-13 Sample Definition for Apache Webserver1.1

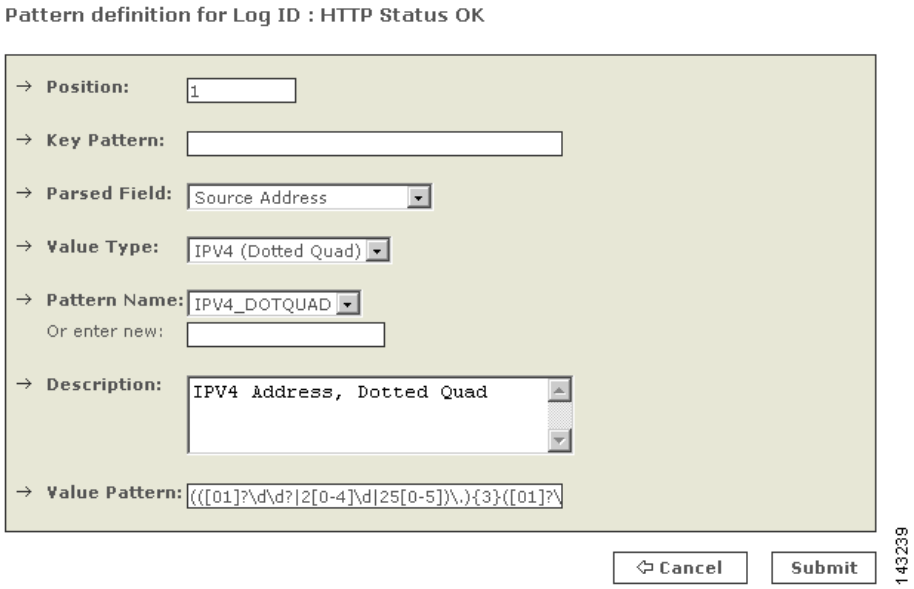


Step 24 Define the log template for a HTTP Status OK log message. And associate a system defined event type. In order to find the event type, specify the search string 'HTTP Status' and find it defined as above.

Step 25 The parsing patterns for 'HTTP Status OK' are specified to match the following example raw message reported in an event.

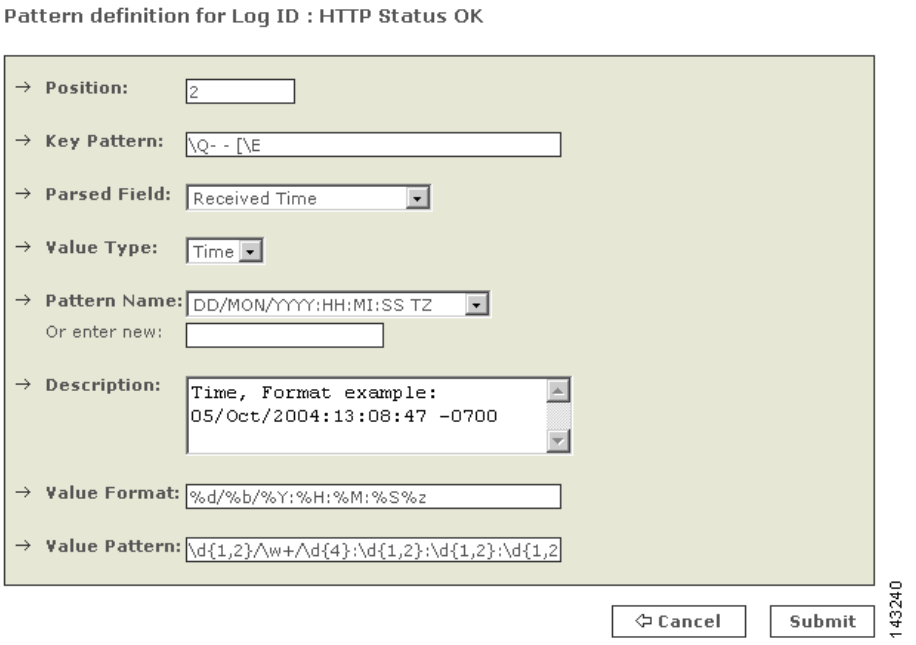
```
155.98.65.40 - - [21/Nov/2004:21:08:47 -0800] "GET /~shash/ HTTP/1.0" 200 1633 "-"
"Lynx/2.8.2rel.1 libwww-FM/2.14"
```

Figure 16-14 Key Pattern for HTTP Status OK



Step 26 The key pattern is empty above since the log message starts with a value pattern.

Figure 16-15 Position 2 Key Pattern for HTTP Status OK



Step 27 The **Parsed Field** above is a Date/Time field. In addition to **Value Pattern**, a value format is required since a date/time can be specified in arbitrarily different ways. Details on how to specify the value format are given in Appendix F. Several pattern names with a few of the commonly used date/time formats have been predefined.

Figure 16-16 Position 3 Key Pattern for HTTP Status OK

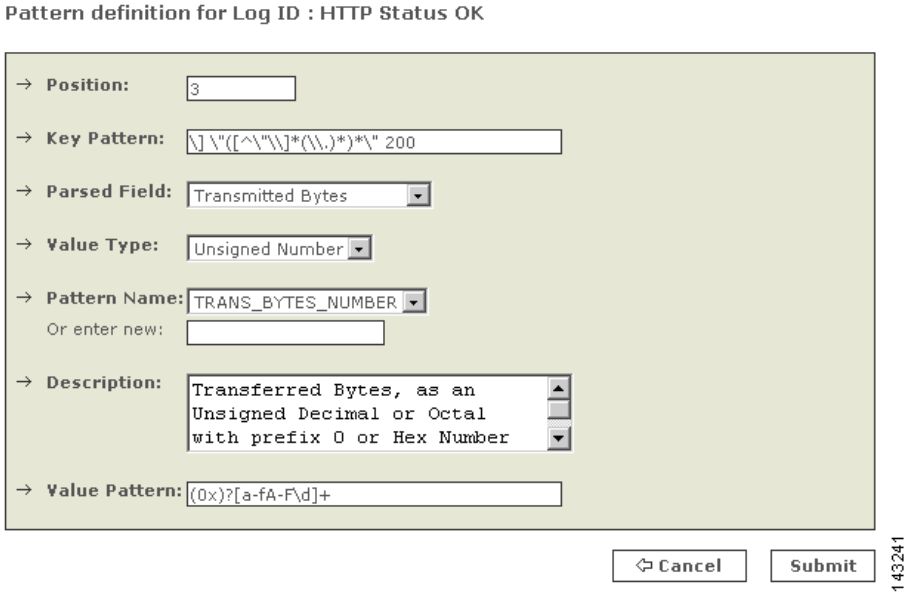


Figure 16-17 Pattern log for HTTP Status OK

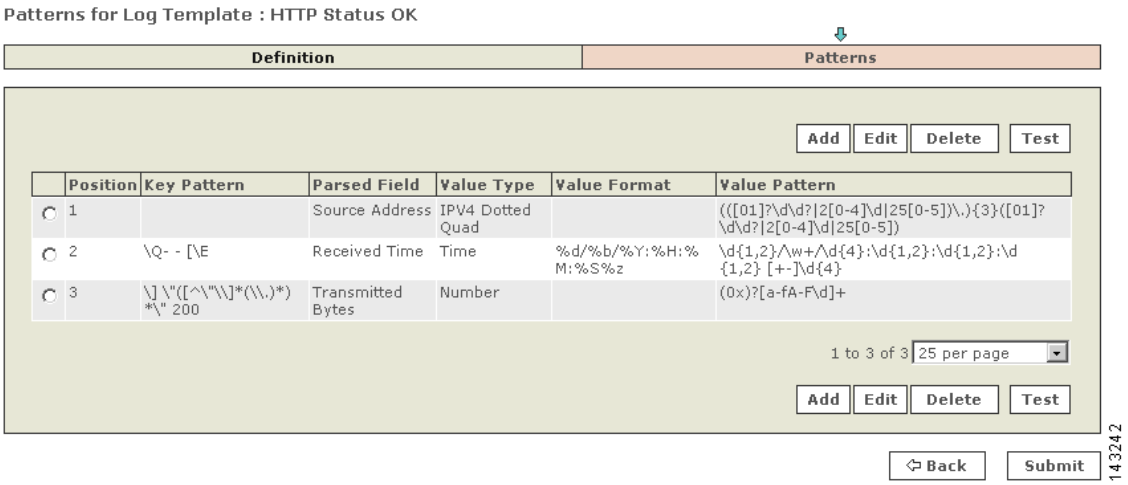
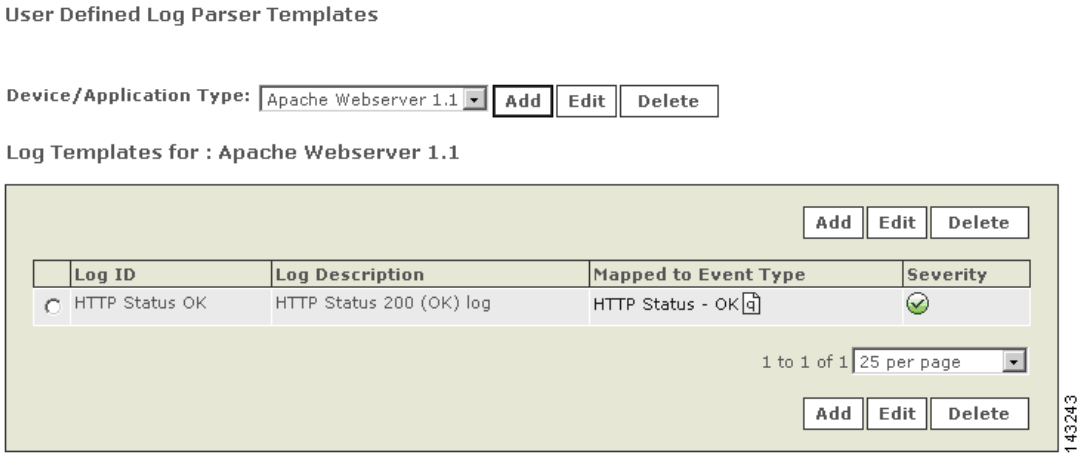


Figure 16-18 User Defined Log Parser for Apache Webserver1.1



Once the log template is defined and submitted, you must define a reporting device based on the custom device.

Add Custom Device or Application as Reporting Device

Adding the new custom device or application is similar as adding predefined device or application. The type of device that you create depends on whether it was defined as a custom appliance or software device. If it is software-based, you must first define a host and then add a reporting application. Otherwise, you can select the appliance type directly from the Device Type list.

The following example adds an instance of the newly defined Apache Webserver1.1 software application.

Step 1 Go to the **Admin** tab.

Step 2 Click the **Security and Monitor Devices**.

Step 3 Click **Add** to add a new device.

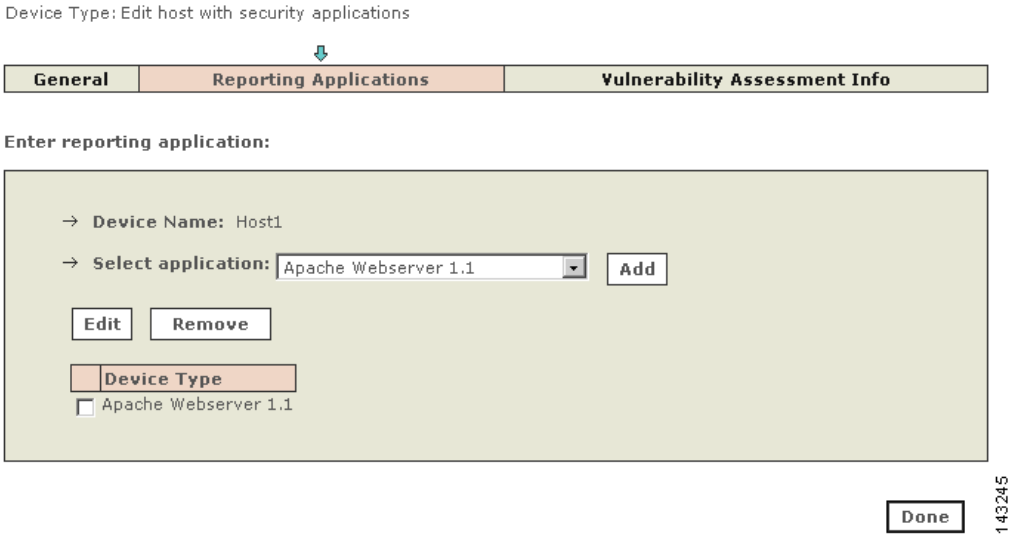
Step 4 From the **Device Type** list, select **Add SW security apps on a new host**.

Step 5 Fill in name and other host details and click **Apply**.

Step 6 Click on **Reporting Applications**.

Step 7 Select **Application** (e.g., Apache Webserver.1.1) from the list and click **Add**.

Figure 16-19 Adding the Customer Application to MARS



Step 8 Select either SNMP TRAP or SYSLOG as the Reporting Method in the resulting window, and click **Submit**.

This option determines what type of traffic will be processed by the custom log parser.

Step 9 Click **Done**.