

---

# **RSA enVision<sup>®</sup>**

# **Event Source Integration**

---

## **Student Guide**

RSA Educational Services



The Security Division of EMC

ED ENVSN ESI  
Rev A0

## Notice

RSA, The Security Division of EMC endeavors to ensure all documentation is accurate and up to date. However, due to the nature of technological evolution the information in this document is subject to change without notice. This document reflects no commitment on the part of RSA, The Security Division of EMC. This document is written and distributed as an instructional tool used as part of an RSA Educational Services training program.

RSA, the RSA logo, RSA Security, ACE/Server, BSAFE, ClearTrust, eFraud Network, enVision, Event Explorer, Fraud Action, Keon, RC2, RC4, RC5, RC6, RSA SafeProxy, RSA SecurBook, RSA Secured, the RSA Secured logo, SecurCare, SecurID, SecurWorld, and Smart Rules are either registered trademarks or trademarks of RSA Security Inc. in the United States and/or other countries. All other goods and/or services mentioned are trademarks of their respective companies.

Copyright © 2010 by RSA Security Inc. All rights reserved.

[ED ENVSN ESI](#)

Rev A0

---

# Contents

---

## Unit 1

<b>Principles of Logging .....</b>	<b>1-1</b>
Objectives .....	1-1
<b>Overview .....</b>	<b>1-3</b>
What is a Log?.....	1-3
Device versus Event Source .....	1-3
Event Messages and Log Data .....	1-3
<b>Event and Log Types .....</b>	<b>1-5</b>
Microsoft Windows Events Classifications .....	1-7
Windows Event Examples .....	1-7
Syslog Protocol.....	1-9
Messages Within the Network.....	1-11
UNIX Syslog Message Format .....	1-13
<b>Syslog, UNIX, and enVision .....</b>	<b>1-15</b>
How Log Messages Arrive at enVision .....	1-15
Log Message Processing by syslogd .....	1-15
Event Source Discovery and Classification .....	1-17
The enVision Event Viewer .....	1-17
Exercise: Preview an Existing XML File .....	1-19

## Unit 2

<b>Log Collection Methods and Formats.....</b>	<b>2-1</b>
Objectives.....	2-1
<b>Collection Services for New Event Source Logs .....</b>	<b>2-3</b>
SNMP .....	2-3
File Reader .....	2-3
Open Database Connectivity (ODBC) .....	2-3
Collection .....	2-3
<b>Universal Device Collection Configuration .....</b>	<b>2-5</b>
Manage Services .....	2-5
SNMP .....	2-7
NIC Trapd Service.....	2-7
Manage SNMP Types .....	2-9
Configuring SNMP Traps.....	2-9

Sample SNMP Trap Data and Logs .....	2-11
Manage File Reader .....	2-13
Configuring File Reader Service for FTP .....	2-13
File Reader Service for SFTP .....	2-15
Sample File Reader Logs .....	2-15
Manage ODBC Types .....	2-17
Configuring ODBC Types .....	2-17
Sample ODBC Logs .....	2-19
<b>Multiple Protocols or Event Sources .....</b>	<b>2-21</b>
<b>Extracting Logs .....</b>	<b>2-23</b>
Isolating and Extracting Logs from the IPDB .....	2- 23
<b>Uncovering Patterns .....</b>	<b>2-25</b>
Sample Solaris File .....	2-25
Tips .....	2-27
Obtain Log Data .....	2-29
Scenario .....	2-29
Inject Log Data .....	2-30
Export the Log Data and Look for Patterns .....	2-31

## Unit 3

<b>How RSA enVision Interprets Logs.....</b>	<b>3-1</b>
Objectives .....	3-1
<b>Data Parsing .....</b>	<b>3-3</b>
Data Parsing Flow .....	3-3
<b>Log Message Contents .....</b>	<b>3-5</b>
Header and Payload .....	3-5
syslog and Log Message Header .....	3-5
<b>Header .....</b>	<b>3-7</b>
Header Contents .....	3-7
Header Example .....	3-7
<b>Message ID .....</b>	<b>3-9</b>
Defining the Message ID in the Header .....	3-9
Message ID Example .....	3-9
<b>Event Source Discovery, Status, and Analysis .....</b>	<b>3-11</b>
Event Source .....	3-11
Event Source Details .....	3-11
Event Source Files .....	3-13
<b>Classes and Subclasses .....</b>	<b>3-15</b>
Event Source Classes .....	3-15
Identifying the Class of a New Event Source .....	3-17
<b>Database Tables .....</b>	<b>3-19</b>
enVision Database Tables.....	3-19

Database Tables and XML Variables .....	3-19
Ways to Select a Database Table .....	3-21
enVision Query Tool Example .....	3-23
Selecting a Database Table .....	3-25
Database Tables in \etc\sqltbl .....	3-27
Variables in a Database Table .....	3-29
<b>Payload .....</b>	<b>3-31</b>
Example.....	3-31
Message Details .....	3-33
<b>Event Categories .....</b>	<b>3-35</b>
enVision Taxonomy .....	3-35
Detail Example .....	3-35
<b>enVision Event Source Files .....</b>	<b>3-37</b>
The .ini File .....	3-37
The .txt Files .....	3-39
Exercise: Determine the Table and Field Values .....	3-41
Scenario .....	3-41

## Unit 4

<b>Creating Support Files .....</b>	<b>4-1</b>
Objectives .....	4- 1
<b>RSA enVision EventSource Integrator .....</b>	<b>4-3</b>
Event Source Integrator Interface.....	4-5
<b>Defining the Header in ESI .....</b>	<b>4-7</b>
Overview.....	4-7
Defining the Header ID .....	4-7
Defining the Message ID in the Header.....	4-9
Defining the Time Stamp in the Header .....	4-11
Defining Variables in the Header.....	4-13
Assigning Null Values to Header Variables .....	4-13
Defining the Payload in the Header .....	4-13
<b>Defining the Message in ESI .....</b>	<b>4-15</b>
Payload.....	4-15
Defining the Message ID .....	4-15
Defining the Severity Level.....	4-17
Defining the Message Variables.....	4-17
Defining the Message Event Category .....	4-19
Defining the Message Static Values.....	4-19
Defining Functions.....	4-21
Static Text.....	4-21
Exercise: Create the Event Source File .....	4-23
Scenario .....	4-23

Select an Event .....	4-24
Define the Header .....	4-26
Define the Message.....	4-28
Validate the XML File and Parse the Message .....	4-32
<b>Modifying Event Source XML Files .....</b>	<b>4-35</b>
Edit Modes.....	4-35
Complete Edit.....	4-37
Tree View .....	4-37
Code View .....	4-37
Editing an Existing XML File.....	4-39
Cloning .....	4-41
<b>Validation and Parsing .....</b>	<b>4-43</b>
Validating an XML File.....	4-43
Validating XML Errors in Code View .....	4-43
Parsing .....	4-45
Reporting .....	4-45
Exercise: Create Additional Messages .....	4-47
Create the Request Message.....	4-47
Create the Terminated Normal Message .....	4-50
Edit the Terminated Normal Message .....	4-53
Create the Terminated Abnormal Message.....	4-55
<b>Event Source Package .....</b>	<b>4-57</b>
<b>Deploying the Event Source Files .....</b>	<b>4-59</b>
Exercise: Create, Deploy, and Test the Event Source Package .....	4-61
Create and Deploy the Event Source .....	4-61
Test the Event Source Integration .....	4-63

## Unit 5

<b>Advanced Topics .....</b>	<b>5-1</b>
Objectives.....	5-1
<b>Functions.....</b>	<b>5-3</b>
PARMVAL Function.....	5-3
SYSVAL Function.....	5-5
CALC Function .....	5-5
HDR Function .....	5-7
SUM, MIN, Max Functions.....	5-7
EVNTTIME Function.....	5-9
<b>Conditional Variables .....</b>	<b>5-11</b>
Conditional Variable Syntax .....	5-11
Value Map .....	5-13
Regular Expressions .....	5-15
<b>Additional Information .....</b>	<b>5-17</b>

---

Advanced Topics Examples .....	5-17
Removing an Event Source .....	5-17
Reporting a Problem.....	5-17
<b>Event Source Updates .....</b>	<b>5-19</b>
How the enVision Event Source Update Works .....	5-19
How the Update Applies Each Patch .....	5-19
Enhancement Requests .....	5-19
Exercise: Add Advanced Functions (Optional) .....	5-21
Add the CALC Function.....	5-21
Add a Value Map .....	5-22
<b>Appendix A</b>	
<b>enVision Taxonomy .....</b>	<b>A-1</b>
<b>Appendix B</b>	
<b>Sample Database Tables.....</b>	<b>B-1</b>

---

# SECTION



---

# Introduction

---

## Course Objectives

Upon completion of this course, you should be able to:

- Explain the steps and tasks that must be completed before enVision can collect and analyze logs from a new event source
- Discover and extract event log messages from an unknown event source
- Create and refine a device XML file so enVision can collect and analyze the event source's event log messages and map message data to the appropriate enVision database table
- Evaluate the new device XML file to find errors and identify opportunities to reduce data and increase efficiency

## Course Audience

This course is intended for the following audience:

- System and network security specialists responsible for configuring RSA enVision event sources and providing basic development support for such activity

## Course Prerequisites

We recommend that students have:

- A functional knowledge of system logging functions (in particular, syslog generation) and Networking Fundamentals
- Familiarity with basic operations involving security event reporting and analysis
- Familiarity with basic development activity such as scripting
- Basic understanding of enVision's database tables and fields

---

# S E T T I N G

---

# Principles of Logging

---

## Objectives

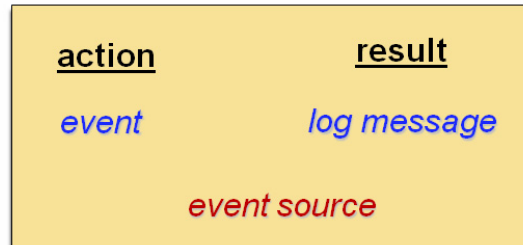
After completing this unit, you should be able to:

- Differentiate between events and log messages
- Describe how log messages are organized
- Describe how the syslog protocol is used in enVision
- Identify the structure of support files

S  
E  
T  
O  
N

## What is a Log?

- ▶ A record of an *event* occurring within hardware or software systems
  - Also referred to as: *log, system log, syslog, Windows Event*
  - Consists of: *messages, log data, event message*
- ▶ Generated at and by an *event source*



# Overview

## What is a Log?

Logs are records of events occurring within hardware or software systems. Terminology referring to log, system log, syslog, Windows Event, etc. are common when referring to logs and logged information.

Typically, logs are generated at and by an *event source* and consist of:

- Messages
- Log data
- Event message

## Device versus Event Source

Traditionally, it had been sufficient to simply refer to a computer or network-related machine as a device, with no need to further identify just where in the device a log was generated or where the event took place. More recently, the scope of logging has expanded to take into account machines that may have multiple functions. For example, it would be quite common to find a server with some application running on its operating system. The events that occur with the system itself (hardware or OS) might generate far different log messages than the application (for example, a web server).

Therefore, RSA is transitioning the terminology from device to the more descriptive event source, and this will be reflected in the user interface in subsequent releases, SPs, and updates.

## Event Messages and Log Data

*Event message* is the term for log data before it is captured by RSA enVision. RSA enVision processes event messages produced by host, network, security, and storage event sources.

*Log data* refers to any message sent to RSA enVision from an event source. When a message is captured in enVision, it is considered an event. The terms event message and log data are interchangeable.

S  
E  
E  
T  
O  
N

## Examples of Events and Logs

Event	Corresponding Log
Cisco PIX firewall denying a network connection	%PIX-2-106007: Deny inbound UDP from 12.15.105.5/8 to 192.168.1.5/9 due to DNS flag
Sun Solaris system shut down	Apr 23 15:04:22 shutdown: [ID 600729 auth.crit] reboot by rashcroft
FreeBSD Unix ssh root user login	Feb 18 17:14:27 www2 ssh[54036]: Accepted keyboard-interactive/pam for root from 127.0.0.1 port 50496 ssh2



---

## Event and Log Types

The table on the slide emphasizes the difference between events and logs. We can describe an event, but we cannot create the series of strings, characters, and codes that make up the log.

Conversely, without referring to documentation or without significant practice, we cannot readily translate what happened (the event) by just looking at the logs and interpreting those characters.

Every vendor of a product, as well as every developer of an application, will determine what events need to be recorded, and how to format the log message. The slide shows some examples of logs that correspond to specific events.

S  
E  
C  
U  
R  
I  
T  
Y

## Microsoft Windows Events Classifications

- ▶ Windows NT 3.1 through XP
  - Log types: Application, Security, System, Directory, File Replication
  - Event types: Information, Error, Warning, Success Audit, Failure Audit
- ▶ Vista and Windows Server 2008
  - Event Log categories: Windows Logs, Applications and Services Logs
    - Windows Logs: Application, Security, Setup, System, Forwarded Events
    - Applications and Services Logs: Admin, Operational, Analytic, Debug



The Security Division of EMC 8

## Example: Windows Events

- ▶ Messages Identifiers (pre-Vista)
  - Event ID 528: Successful Logon (Security, Success Audit)
  - Event ID 540: Successful Network Logon (Security, Success Audit)
  - Event ID 538: User Logoff (Security, Success Audit)
  - Event ID 539: Logon Failure; Account Lockout (Security, Failure Audit)
  - Event ID 134: Removable Storage ARRIVAL (Information)
  - Event ID 135: Removable Storage REMOVAL (Information)
- ▶ New Windows Events Identifiers
  - Vista Event ID = pre-Vista Event ID + 4096
  - Successful Logon is now Event ID 4624



The Security Division of EMC 9



---

## Event and Log Types, continued

### Microsoft Windows Events Classifications

Microsoft serves as an example of a vendor with the requirement to record events, even if the product is limited to (only) software.

This is where the term *event source* applies, since it is somewhat unnatural to refer to an operating system by the term of *device*.

Microsoft has an extensive classification structure in their logging capabilities, all centered around the term *Windows events*. It is beyond the scope of this class, as well as unnecessary to know to use enVision, to go into any detail about Windows events or the classification structure, but serves as a demonstration of the vendor-specific nature of events and their logs.

Microsoft also illustrates the changing nature of the vendor environment, as the structure, classification and operations of Windows events has changed dramatically with the introduction of Windows Vista and Windows Server 2008.

### Windows Event Examples

The slide on the left provides a few examples of how Microsoft uses their structure to associate events with particular message identifiers, as well as to point out the changes rendered with the Vista operating system.

This particular events list would undoubtedly be of interest to security analysts, especially ID 134 and ID 135, which could be used to monitor the presence and use of ubiquitous and possibly notorious USB drives.

S  
E  
T  
O  
N

## RFC 5424: The Syslog Protocol

- ▶ March 2009
  - Obsoletes RFC 3164 from August 2001
- ▶ Describes the protocol used to convey event notification messages
  - Layered architecture
    - Separates message content from message transport
    - Allows for vendor extensions
- ▶ Defines the basic concepts and functions of logs
  - Does not describe storage or analysis of log messages



## RFC 5424: The Syslog Protocol, continued

- ▶ Should support UDP-based transport
  - Vendors implement as meets their requirements
- ▶ RSA enVision implements syslog as UDP
  - RSA-specific definitions of concepts and principles
  - TCP configuration is optional
- ▶ RFC 5424 serves as an example
  - Detailed understanding of RFC 5424 is not required to use enVision effectively



---

## Event and Log Types, continued

### Syslog Protocol

RFC 5424 defines the Syslog protocol.

Syslog is a logging application that transmits a maximum 2048-byte text message to the syslog receiver. The receiver is commonly called syslogd, syslog daemon or syslog server.

Syslog messages may be sent via the User Datagram Protocol (UDP) or the Transmission Control Protocol (TCP). The data is sent in cleartext; although not part of the syslog protocol itself, an SSL wrapper may be used to provide for a layer of encryption through SSL/TLS.

Syslog uses the port number 514.

Syslog is typically used for computer system management and security auditing. While it has a number of shortcomings, syslog is supported by a wide variety of devices and receivers across multiple platforms. Because of this, syslog can be used to integrate log data from many different types of systems into a central repository.

Syslog is standardized within the Syslog working group of the Internet Engineering Task Force (IETF). The details can be found in RFC5424.

S  
E  
E  
T  
O  
N

### Syslog, UNIX and enVision – “on the wire”

802.3 header	IP header	UDP header	data	CRC
	<i>Version, IHL, DiffServ, Total Length, Ident., Flags, Offset, TTL</i> Protocol 17 (UDP) Header Checksum Source Address 10.10.50.5 Destination Address 10.10.10.31	<i>Source Port</i> <i>Destination Port</i> 514 <i>Length</i> <i>Checksum</i>	<34>1 2010-02-22T08:17:05.52-4:00 acomputer.rsa.com evtslog - ID810491 - BOM'su root' failed for tbrady on /dev/pts/8	

The Security Division of EMC 12

---

## Event and Log Types, continued

### Syslog Protocol, continued

#### Messages Within the Network

Syslog messages are transported on the LAN within frames and packets, as is all other local network traffic. By design, enVision has the syslog daemon installed and does not require any configuration to collect logs. In the case of the UNIX system shown in the slide on the left, the event source has been configured to send its logs to enVision, with its address (source address) and the enVision appliance's address (destination address) in the IP header, along with a protocol indicator of 17 (UDP), and the rest of the required and optional fields in an IPv4 header.

IPv6 headers have a much different format. However, for the purposes of this discussion, the relevant fields for enVision remain the source address and protocol. Within the UDP header, a port is indicated for this particular service on the enVision appliance. Thus, enVision can associate a particular message with an IP address, as enVision processes this IP and UDP information.

The next step is to process the information in the data field, which is the syslog message. Each syslog event will be completely transported within one IP/UDP packet. Therefore, each packet contains information about only one event.

Notice the series of spaces, alphanumeric and other characters that are listed below the data field. This is an example of a UNIX syslog message that we will be using throughout this class.

## UNIX syslog Message Format (Selected Fields)

- ▶ Program name
  - Examples: login, su, ftpd, telnetd
- ▶ Facility name/value
  - Kernel (kern/0), Regular user processes (user/2), Mail system (mail/2), Line printer system (lpr/6), Authorization system (auth/4), Other system daemons (daemon/15), News subsystem (news/7), UUCP system (uucp/8), Syslog (log/13), Reserved for site-specific use (local0/16 through local7/23), Any (\*)
- ▶ Priority name/Level
  - Emergency condition (emerg/0), Condition that should be corrected immediately (alert/1), Critical (crit/2), Ordinary error (err/3), Warning (warning/4), Condition should possibly be handled in a special way (notice/5), Informational (info/7), Debugging (debug/7), none
    - mail.err, lpr.notice, auth.notice, \*.crit, mail.\*



The Security Division of EMC 13

## UNIX syslog Message Format (IP/UDP Packet)

```
<34>1 2010-02-22T08:17:05.52-4:00 acomputer.rsa.com
evtslog - ID810491 - BOM'su root' failed for tbrady
on /dev/pts/8
```

- PRIVAL of <34> decodes to: 34 divided by 8 = 4, therefore Facility is "security/authorization messages" and the remainder is 2, therefore Severity is "Critical"
- 1 is Version One of syslog
- On the morning of February 22, 2010 at 17 minutes 5 seconds past 8 U.S. Eastern Time
- At the computer system named "acomputer" at rsa.com
- Application named "evtslog"
- No (optional) Process ID provided, hence the "-" for nil value
- The Message ID is "ID810491"
- No (optional) Structured Data included, hence the next "-" for nil value
- "BOM" (Byte Order Mark) means the rest will be encoded in UTF-8
- Next is the MSG part; a free-form message that provides information about the event. This part will be interpreted from cryptic UNIX Solaris to narrative English:
  - An attempt to change ownership of the session from an ordinary user to administrative user 'su root' failed, by a user "tbrady" logged into the computer using a terminal emulator "dev/pts/8".



The Security Division of EMC 14

SECURITY

---

## Event and Log Types, continued

### Syslog Protocol, continued

#### UNIX Syslog Message Format

There is information carried in the data field of a message packet that is different from the IP/UDP protocol headers.

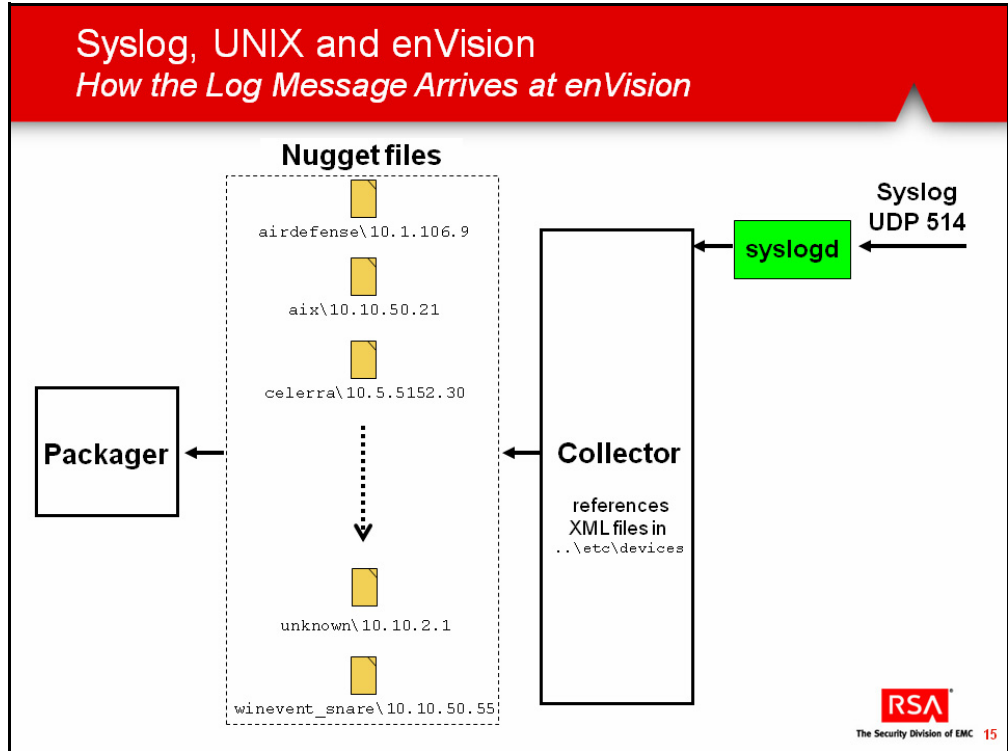
This belongs to the entity that is simply using IP/UDP to get a message delivered, much like we rely on the Post Office, UPS or FedEx. This is where the log message is carried, encoded in a bit or byte orientation, beginning with a description of the circumstances of the event. It is beyond the scope of this class to go into too many details concerning the layout and operations of this message that was generated by a UNIX computer.

However, it illustrates that events, as well as the circumstances surrounding them, are represented in log messages. Furthermore, it might help to understand how enVision works and interprets such logs into events. For example, we know that Alerts can display eight levels of Severity. In UNIX, these are encoded as Priority or Level.

The bottom slide on the left provides a detailed look at our example message, the syntax and meaning of which is determined only after extensive research using IETF RFCs, vendor documentation, and Google searches.

This process can be referred to as *parsing*, and we will examine how enVision does its parsing in Unit 3 of this course.

S  
E  
E  
T  
O  
N



### Syslog, UNIX and enVision

*Log Message as Processed by syslogd*

```
Feb 22 08:17:05 [10.10.50.5] Feb 22 08:17:05 su: [ID 810491 auth.crit]'su root' failed for tbrady on /dev/pts/8
```

enVision's syslogd compiles information from many parts of the message, such as:

- When the message was received (enVision Timestamp)
- Where the message came from - IP address from within the IP header
- What is the event source Timestamp, Priority, Severity from the syslog fields
- What is the Program Name and Message ID, as well as the rest of the log message from the UNIX fields

The Security Division of EMC 16



# Syslog, UNIX, and enVision

## How Log Messages Arrive at enVision

As shown in the diagram on the left, the log message arrives at the enVision appliance and the IP/UDP headers are analyzed.

A syslog daemon, or syslogd then processes the information within the data field.

Once that is complete, syslogd then hands off information to the Collector. The Collector then attempts to identify the device type against all the known device types represented by XML files in a folder named `..\etc\devices` and adds a Time Stamp and an IP address. The next step is the creation of nugget files, which will ultimately be picked up by the Packager at the top of every minute.

## Log Message Processing by syslogd

The slide to the lower left shows the information that the syslog daemon, syslogd, passes to the Collector. The information that is parsed from this log message and the IP/UDP headers includes:

- When the message was received:
  - The enVision Time Stamp
- Where the message came from:
  - The IP address from within the IP header

The payload may include:

- The event source Time Stamp, Priority, and Severity from the syslog fields
- The Program Name and Message ID
- The remaining log message from the UNIX fields

S  
E  
E  
T  
O  
N

## Syslog, UNIX and enVision Event Source Discovery and Classification

The Collector references the XML files in `..\etc\devices`

- ▶ Each log message is processed against all the XML files until a match is found for the Message (Device Discovery)
- ▶ If a match is found, event source type is determined, and logs are filed into `..\tmp\nuggets\ by IP address
 
  - The event source is listed by name, its status is "Active" and it is now being monitored and analyzed`
- ▶ If it matches more than one device, it is also discovered as unknown since the Collector cannot determine what it is
- ▶ If no match found during the Sample interval (time or count), event source is listed as "unknown" and all logs are filed into `..\tmp\nuggets\unknown` by IP address
  - The event source is listed as "Candidate" and is not analyzed



The Security Division of EMC 17

## Syslog, UNIX and enVision enVision Event Viewer

The screenshot shows the RSA enVision Event Viewer interface. The main window displays a table of events with the following columns: Index, Date/Time, Device, and Event. A single event is visible in the table.

Index	Date/Time	Device	Event
1	2010/02/22 08:17:05.625 EST	10.10.50.5	Feb 22 08:17:05 su: [D 810491 auth.crit] 'su root' failed for tbrady on kdevlpts8

At the bottom of the interface, there is a status bar that reads: "Displaying matching Events 1 Thru 2 of 2355696 Total Events, Server time 18.8616 seconds, Transfer to Client in 0.14 seconds, Effective Transfer ..."

18

## Syslog, UNIX, and enVision, continued

### Event Source Discovery and Classification

Events are discovered and classified via the Collector as follows:

1. The Collector references the XML files in `..\etc\devices`.
2. Each log message is processed against all the XML files until a match is found for the message
3. If a match is found, the event source type is determined, and logs are filed into `..\tmp\nuggets\<devicetype>` by IP address.
  - The event source is listed by name, its status is set to “Active” and it is now being monitored and analyzed
4. If it matches more than one device, it is also discovered as unknown since the Collector cannot determine what it is
5. If no match is found during the Sample interval (time or count), the event source is listed as unknown and all logs are filed into `..\tmp\nuggets\unknown` by IP address.
  - In this case, the event source is listed as Candidate and is not analyzed

### The enVision Event Viewer

Regardless of whether the log is from a known or unknown source, that log message will be viewable in enVision’s Event Viewer tool.

The slide to the lower left shows how the log message in the previous example is displayed in enVision. Even in this Event Viewer window, however, we still cannot readily determine just what the event was that generated this log message.

At this point, we might make a note of the IP address shown and research it at Manage Monitored Devices to see if is known or unknown.

# SECTIONS

## Exercise 1 Preview an Existing XML File

### The goal of this exercise is:

- Open an existing XML file
- Explore the mechanisms used for creating a new event source

Step	Action
1	Navigate to the Device Support files directory: <b>c:\nic\4000\&lt;nodename&gt;\etc\devices</b>
2	Navigate to the <b>ciscopix</b> directory.
3	Right-click the file: <b>ciscopixmsg.xml</b> , select <b>Open with</b> and select <b>Notepad++</b> or <b>WordPad</b> .
4	Write down the number of <b>HEADER</b> tag(s) used. _____
5	List any other tags used. _____ _____ _____ _____
6	Note the <b>level="#"</b> entry that varies from <b>0</b> to <b>7</b> as you scroll down through various <b>MESSAGE</b> tags. This represents the Severity level, where the higher number is a less critical event and a value of zero is interpreted as emergency.
7	Note the <b>eventcategory="#####"</b> This is the codification of the enVision Taxonomy that is used throughout the NIC System. Refer to Appendix A for the enVision message taxonomy.
8	In the first <b>MESSAGE</b> tag, note the <b>tableid="21"</b> entry. This is the Database Table that the message will be parsed to. 21 is the internal reference for what the user will see as the displayed database table name of <b>FireWall System</b> .

## Exercise 1, continued

Step	Action
9	<p>Note the <b>id1="#"</b> and <b>id2="#"</b> entries</p> <p>These are very important fields that must be unique within this particular Device Support file.</p> <p><b>Note:</b> Although it is always derived from the original (Vendor) Message ID. We will see when and how to modify this in later Exercises.</p>
10	<p>Note the <b>content="..."</b> field</p> <p>This is where you define every variable that will be used in a particular Database Table. In addition to that, this field can be used to assign variables, summarize other variables referred to as Conditional Variables, and generally modify the performance of the MESSAGE tag.</p>
11	<p>Scroll down though the collection of MESSAGE tags and look for other values for tableid.</p> <p><b>Note:</b> There are at least 50 with tableid="21". After that, you should find:</p> <p>1 – FireWall Level, 9 – FireWall Security, 12 – FireWall Accounting, 17 – FireWall Email Security, 3 – FireWall AAA Authentication, 28 – VPN Security, 10 – VPN Accounting, 8 – VPN System and possibly others.</p>
12	<p>The total quantity of <b>MESSAGE</b> tags found in the XML file should be the same count that you would find in enVision under <b>Overview &gt; System Configuration &gt; Messages &gt; Manage Messages &gt; Device / Type IN Cisco PIX Firewall</b>, - well over 1,000 messages.</p> <p><b>Note:</b> Be sure to UNCHECK the <b>Apply Filter to monitored devices only</b> box, so that all messages will display.</p>
13	<p>To see how many messages are supported for this event sources, select <b>Manage Messages &gt; Manage Messages to Parse</b>.</p>
14	<p><b>Close</b> the file without saving.</p>

---

# Log Collection Methods and Formats

---

## Objectives


After completing this unit, you should be able to:

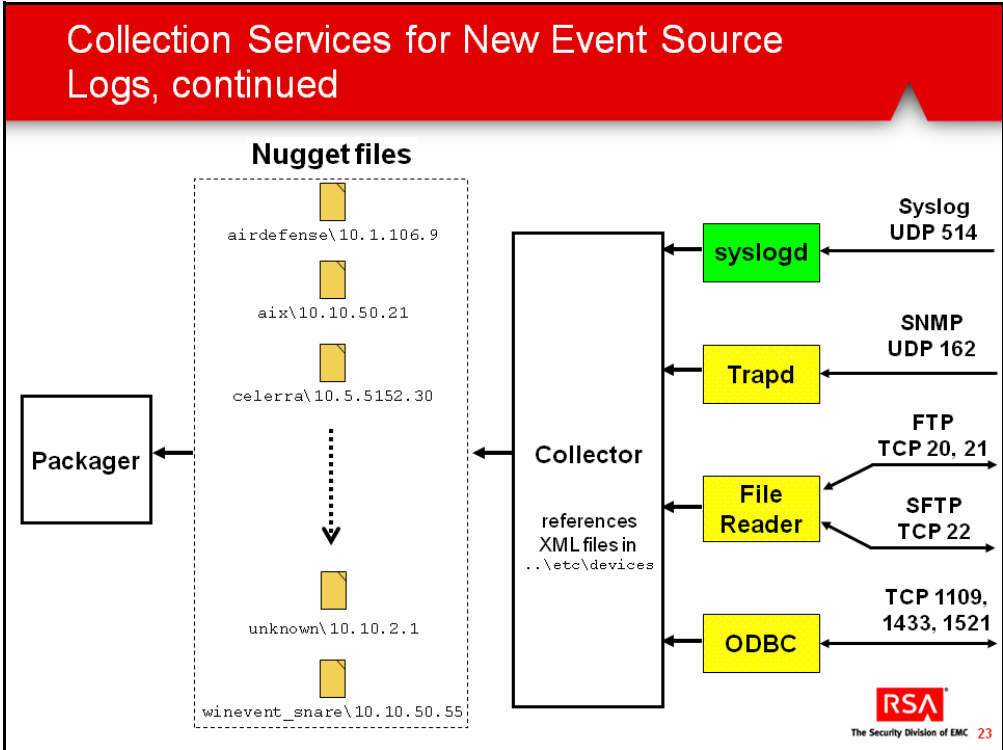
- List enVision's alternative log collection methods
- Identify when to use a particular collection service
- Outline the process to set up an alternative collection service
- Extract log files

S  
E  
C  
U  
R  
I  
T  
Y  
E  
N  
V  
I  
S  
I  
O  
N

### Collection Services for New Event Source Logs

- ▶ SNMP
- ▶ File Reader
  - FTP
  - SFTP - downloadable from RSA SecurCareOnLine
- ▶ ODBC


  
The Security Division of EMC 22





# Collection Services for New Event Source Logs

The focus of this class is on the creation of the event source XML files, which requires that the log messages from the new event source be analyzed from the syslog format.

However, not every vendor or application developer uses syslog, so to put the collection of logs from legacy or emerging event sources into the perspective of what might be encountered in the field, we need to mention other methods.

## SNMP

Simple Network Management Protocol (SNMP) is a UDP-based network protocol. It is used mostly in network management systems to monitor network-attached devices for conditions that warrant administrative attention. Data is sent to enVision in the format of traps which are processed by the NIC trapd Service.

SNMP is a component of the Internet Protocol Suite as defined by the Internet Engineering Task Force (IETF). It consists of a set of standards for network management, including an application layer protocol, a database schema, and a set of data objects.

## File Reader

Log files are text files that contain structured log messages, typically all sharing the same structure. The messages use a delimiter character to separate the various fields. Some devices provide a header line at the top of the file to describe the structure.

Most devices keep a single log file, but you may have a device that logs to multiple files, each containing different log messages.

When working with these types of files you must have a way to get the log file to the enVision system. enVision needs the log file to process and forward the events to the NIC Collector Service, which uses the file transfer protocol, commonly referred to as FTP.

RSA also supports a Secure FTP Agent that can be configured to check for a particular file as often as once every minute (refer to <https://knowledge.rsasecurity.com>).

## Open Database Connectivity (ODBC)

ODBC provides a standard software API method for using database management systems (DBMS). The designers of ODBC aimed to make it independent of programming languages, database systems, and operating systems.

## Collection

All of enVision's collection services continue to rely on the services provided by the IP/UDP or IP/TCP packet protocols, thus identifying the general content of the data field. The port numbers used by the various enVision collection services are listed on the slide shown on the left.

S  
E  
T  
U  
P

## Universal Device Collection Configuration - Manage Services

► System Configuration > Services > Manage Services

System Configuration - RSA enVision - Windows Internet Explorer

Use this window to display the current processing status of the NIC services and to start and stop the services.

Site: USPSAMIDODL1C

Start/Stop Service	Service Name	Node	Status	Logging Enabled	Logging Level
<input type="checkbox"/>	NIC Alerter	USPSAMIDODL1C	Stopped	<input checked="" type="checkbox"/>	6
<input type="checkbox"/>	NIC App Server	USPSAMIDODL1C	Running	<input checked="" type="checkbox"/>	6
<input type="checkbox"/>	NIC Asset Collector Service	USPSAMIDODL1C	Running	<input checked="" type="checkbox"/>	6
<input type="checkbox"/>	NIC Asset Processor Service	USPSAMIDODL1C	Running	<input checked="" type="checkbox"/>	6
<input type="checkbox"/>	NIC Collector	USPSAMIDODL1C	Running	<input checked="" type="checkbox"/>	6
<input type="checkbox"/>	NIC DHCP Polling Service	USPSAMIDODL1C	Stopped	<input checked="" type="checkbox"/>	6
<input type="checkbox"/>	NIC File Reader	USPSAMIDODL1C	Stopped	<input checked="" type="checkbox"/>	6
<input type="checkbox"/>	NIC FW-1 Lea Client	USPSAMIDODL1C	Running	<input checked="" type="checkbox"/>	6
<input type="checkbox"/>	NIC Locator	USPSAMIDODL1C	Running	<input checked="" type="checkbox"/>	6
<input type="checkbox"/>	NIC Logger	USPSAMIDODL1C	Stopped	<input checked="" type="checkbox"/>	6
<input type="checkbox"/>	NIC ODBC Service	USPSAMIDODL1C	Stopped	<input checked="" type="checkbox"/>	6
<input type="checkbox"/>	NIC Packager	USPSAMIDODL1C	Running	<input checked="" type="checkbox"/>	6



The Security Division of EMC 24

---

# Universal Device Collection Configuration

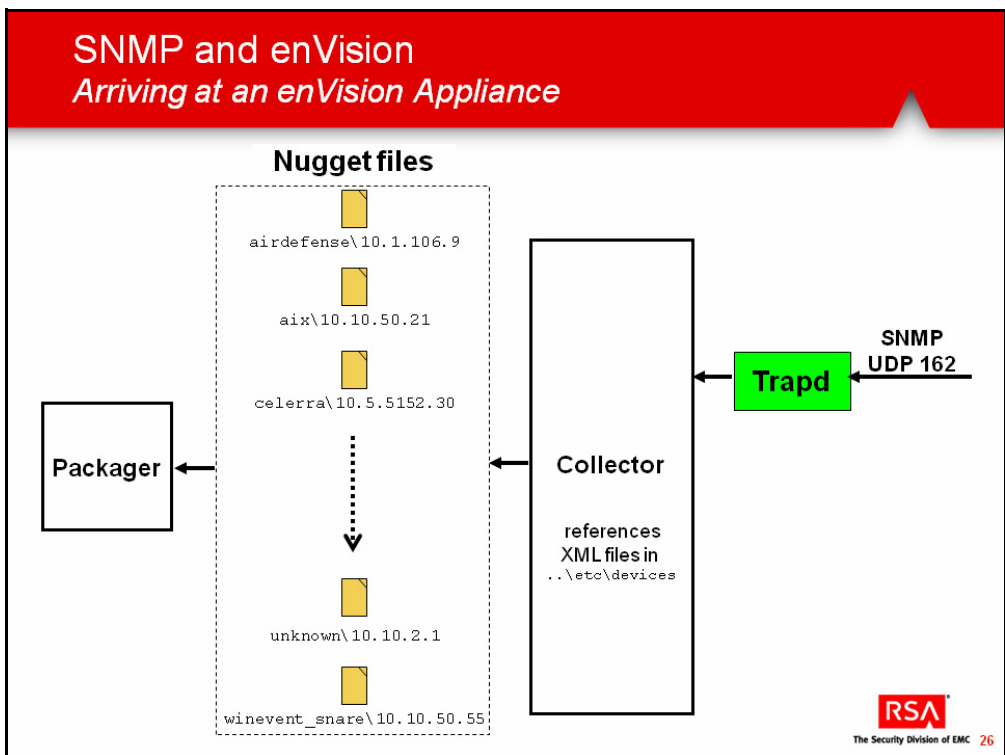
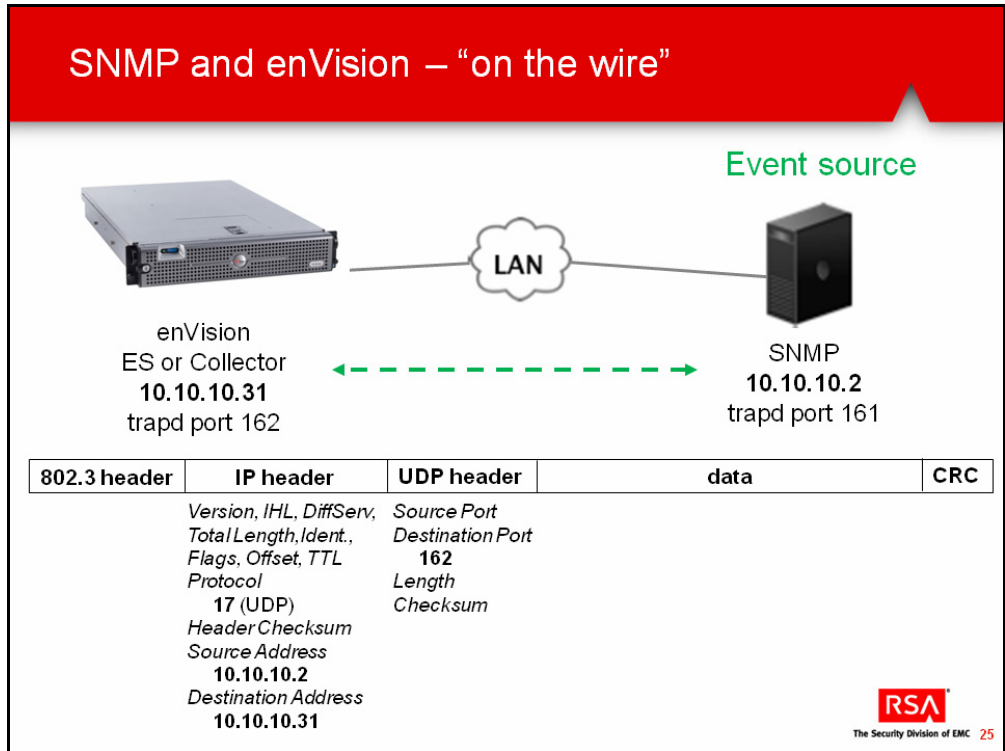
## Manage Services

Since enVision uses syslog by default, that Collection service cannot be stopped/started, and is integral to the operation of the NIC Collector Service.

Some event sources may require processing by a specific service other than syslog, for example, File Reader, SNMP or ODBC. These services are discussed on the following pages. If a different Collection Service is required, use the Manage Services function to start the service.

In the example shown on the left, notice that both the NIC File Reader and NIC ODBC Services are not running. If we were to capture the rest of this display, however, we would see that NIC Trapd Service is running.

S  
E  
E  
T  
O  
N



## Universal Device Collection Configuration, continued

### SNMP

SNMP traps are used by various products to send alerts and notifications triggered by specific conditions. The traps are structured based on a Management Information Base (MIB) available with the product.

SNMP traps are different than syslog, because unlike a syslog message that consists of a single line, an SNMP trap is a data structure that contains a number of fields in a given order.

Products that support SNMP traps publish their MIB, which describes that data structure. enVision has the capability of accepting traps from event sources that can produce them, but it needs some information in order to create an internal syslog event that represents that trap.

SNMP may be used in the opposite direction as well, and the originating event source would need to have an SNMP daemon configured for its port 162. The SNMP configuration for the event source determines how it is used.

### NIC Trapd Service

The NIC Trapd service collects the SNMP traps, formats them based on the UDS device configuration and sends them to the Collector.

S  
E  
T  
T  
I  
N  
G

## Universal Device Collection Configuration Manage SNMP Types

Delete	Name	Tag	Vendor ID
<input type="checkbox"/>	Trend	TRENDMICRO	6101.141
<input type="checkbox"/>	Symantec AntiVirus	SYMANTECAV	343.2.5
<input type="checkbox"/>	Dragon IDS	DRAGONIDS	4471.0
<input type="checkbox"/>	Websense	WEBSENSE	114364
<input type="checkbox"/>	Symantec Intruder Alert	INTRUDERALERT	3088.0.11
<input type="checkbox"/>	Computer Associates Integrated Threat Management	CAITM	48.879
<input type="checkbox"/>	CipherTrust IronMail	IRONMAIL	7441
<input type="checkbox"/>	EMC Celerra	CELERRA	1139.2
<input type="checkbox"/>	Websense 6.3	WEBSENSE	23365
<input type="checkbox"/>	Lotus Domino	LOTUSDOMINO	334.72
<input type="checkbox"/>	Sophos	SOPHOS	2604.2.1
<input type="checkbox"/>	EMC Clarion	CLARION	1981
<input type="checkbox"/>	Dell RAC	DRAC	3183.1.1
<input type="checkbox"/>	mysql	MySQL	24993

27

## Configuring SNMP Traps

28



## Universal Device Collection Configuration, continued

### Manage SNMP Types

There are a number of pre-configured SNMP traps available in enVision. You can review their configurations by clicking on the hyperlink in the Manage SNMP Traps configuration window shown in the slide.

### Configuring SNMP Traps

To configure enVision to collect SNMP traps from an event source, the enVision administrator must enter the following information:

Data Field	Explanation
Name	The name to which the event source is referred in enVision.
Tag	The tag for the SNMP trap. This tag is prepended to syslog to identify the event source.
Vendor ID	The ID that identifies the specific event source sending the traps. This information is available in the vendor's MIB.
Vendor ID location	The location of the vendor ID.
Message ID location	<ul style="list-style-type: none"> <li>No messages defined Select this option if you do not want to define a specific Message ID location.</li> <li>Location The line number of the trap in which the Message ID resides.</li> </ul>
Output field delimiter	<p>The delimiter separating the fields in the syslog message.</p> <p><b>Note:</b> If this box is left blank, a space will be used to separate the fields.</p>
Field locations	<ul style="list-style-type: none"> <li>All Include all field locations.</li> <li>Locations Specific trap line numbers separated by a comma. The order in which you type these locations is the order enVision uses to create the syslog message. You can enter a single location or multiple locations. If you are entering multiple locations, you must separate each location with a comma. Each location can be either a numeric value (in the range of 0-99) or a character string that is preceded by a #.</li> </ul>

S  
E  
T  
O  
N

## Sample Trap Data

```
10.19.52.55
10.19.52.55
system.sysUpTime.0 290:4:34:11.76
.iso.org.dod.internet.snmpV2.snmpModules.snmpMIB.snmpMIBObje
cts.snmpTrap.snmpTrapOID.0 enterprises.1139.2.0.1
enterprises.1139.2.1.1.1 27
enterprises.1139.2.1.1.2 7
enterprises.1139.2.1.1.3 6
```



## Sample Trap Data, continued

```
enterprises.1139.2.1.1.4 "Feb 26 15:17:36 2010 CFS:6:7 Slot
2: 1266009448: The file system size (fs /iSCSI_SQL_Log)
dropped below the threshold of (90%) "
.iso.org.dod.internet.snmpV2.snmpModules.snmpCommunityMIB.sn
mpCommunityMIBObjects.snmpTrapAddress.0 10.19.52.55
.iso.org.dod.internet.snmpV2.snmpModules.snmpCommunityMIB.sn
mpCommunityMIBObjects.snmpTrapCommunity.0 "public"
.iso.org.dod.internet.snmpV2.snmpModules.snmpMIB.snmpMIBObje
cts.snmpTrap.snmpTrapEnterprise.0 enterprises.1139.2
```

---

### What is sent to the Collector:

```
@10.19.52.55 %CELERRA Feb 26 15:17:36 2010 CFS:6:7 Slot 2:
1266009448: The file system size (fs /iSCSI_SQL_Log)
dropped below the threshold of (90%)
```





## Universal Device Collection Configuration, continued

### Sample SNMP Trap Data and Logs

The files shown on the slides to the left provide an example of how logs collected by SMTP might look.

S  
E  
C  
U  
R  
I  
T  
Y

### Universal Device Collection Configuration Manage File Reader

System Configuration - RSA enVision - Windows Internet Explorer

Use this window to display the current universal file reader configurations.

Delete	Name	Device Tag
<input type="checkbox"/>	NetCache	
<input type="checkbox"/>	BlueCoat	
<input type="checkbox"/>	ACS	
<input type="checkbox"/>	CiscoContentEngine	
<input type="checkbox"/>	IS	
<input type="checkbox"/>	Oracle	
<input type="checkbox"/>	APACHE	APACHE
<input type="checkbox"/>	RSA ACE	RSAACE
<input type="checkbox"/>	MSISA	MSISA
<input type="checkbox"/>	MSISA PF	MSISAPP
<input type="checkbox"/>	Exchange	MSEXchange
<input type="checkbox"/>	Exchange2007	MSEXchange
<input type="checkbox"/>	generic	GENERIC_FLEREADER
<input type="checkbox"/>	ISERES	
<input type="checkbox"/>	MSSQL	MSSQL
<input type="checkbox"/>	tripwire	
<input type="checkbox"/>	Solaris_BSM	SBSM
<input type="checkbox"/>	BlueCat FI FF	CACHFI OWFI FF

Apply Add

31

### Configuring File Reader Service for FTP

System Configuration - RSA enVision - Windows Internet Explorer

Use this window to add or to modify the universal file reader configurations.

Manage File Reader - Add/Modify Universal File Reader Definition

Name:

Device Tag:

Data start line:

Message ID location:

Delimiters:

Acceptable delimiters:

Field delimiter:

Line delimiter: CR

Apply Cancel

The Security Division of EMC 32

## Universal Device Collection Configuration, continued

### Manage File Reader

There are a number of pre-configured File Reader services available in enVision for supported event sources that have File Reader Collection. You can view their configurations by clicking on the hyperlink in the Manage File Reader configuration window shown at left.

### Configuring File Reader Service for FTP

Configure a new File Reader Definition with the following fields:

Data Field	Explanation
Name	The event source name. This is the first part of the directory where the input files will reside. The directory is in the format name_ip and is located in the ftp folder, for example, CELERRA_10.10.10.5.
Device Tag	The tag added to the syslog message to identify the event source. The tag is prepended to the syslog in the format, %device tag, for example, %CELERRA.  The name and device tag should be the same.
Data start line	The numerical line value to indicate where data starts.
Message ID location	The numerical field to indicate the Message ID location. This is the field in the input data that is the Message ID. This field should be left blank.
Field delimiter	The delimiter separating the fields.  The default value for this field is a '.' (period). You may leave these fields blank if the field to the right is left blank.  You can type either a displayable delimiter or the code for a non-displayable character.
Line delimiter	The delimiter separating lines.  The default value for this field is CR (carriage return). You can use CR or CR LF for Windows and LF only for UNIX.

S  
E  
C  
U  
R  
E  
N  
E  
S  
S

## Configuring File Reader Service for SFTP

- ▶ NIC SFTP Agent is the executable (Windows only)
- ▶ At the SecurCare website where the SFTP executable can be downloaded, there is a hyperlink (**Device Setup**) where you can retrieve the configuration document
- ▶ This downloads a file named “NIC SFTP Agent,” which contains extensive and detailed instructions
  - This document is entitled: *RSA enVision NIC SFTP Agent Configuration Article*
- ▶ SecurCare also includes a UNIX shellsript for SFTP



## Sample File Reader Files

```
Feb 26 10:00:05 [10.10.11.15] %SITESCOPE-4: 09:59:26
02/26/2010 good DAWeb_Portal - 24X7 Applications 24X7:
Customer Portal and Applications: Customer Portal (DAWeb)
- 24X7: URL Check - Customer Portal Login Page MEDH 1.609
sec 24:103881 200 1609 ok 10792 1264492797 0 16 1577 16 0
200 0
```

```
Feb 26 10:03:15 [10.10.11.15] %SITESCOPE-4: 10:02:14
02/26/2010 good MK - Platewatch Printers 24X7 Applications
24X7: MK (Manufacturing Knowledge): MK - Platewatch 24X7
Monitored in KC DC: Process - MK PlatePress Image 7 on
DOKCMKPW02 DCMNKC running 30:69014 running true true 1 0 0
```

```
Feb 26 10:06:06 [10.10.11.15] %SITESCOPE-4: 10:04:35
02/26/2010 good Processedslammqp002 Applications 24X7:
Sonic Messaging: Sonic Messaging on edslammqp002 (IH)
24X7: Process - Sonic Messaging Broker Domain1.ctIH12 on
edslammqp002 running 3:75053 running true no data 1 0 0
```



---

## Universal Device Collection Configuration, continued

### File Reader Service for SFTP

RSA also supports a Secure FTP Agent, NIC SFTP Agent, which can be downloaded from RSA SecurCare: <https://knowledge.rsasecurity.com>. This agent is for Windows only. You can also download the configuration document, called *RSA enVision NIC SFTP Agent Configuration Article* by clicking the Device Setup link.

---

Note: There is also a UNIX shellsript that is available on RSA SecurCare.

---

The NIC SFTP Agent is used in enVision for both asset import and event collection. It sends data to enVision via Secure FTP. You may want to use this so that the data is encrypted, for example, if it is coming from a different domain.

You can configure the NIC SFTP Agent to check for a particular file as often as once every minute and as little as once every day. At each check, any new data written to the file since the last check will be sent via Secure FTP to the configured host.

Error and informational logs generated by the Agent are sent via UDP to the configured logging host. All configuration information is stored in `sftpage.conf` (the SFTP Agent Configuration file).

The following operating systems support the NIC SFTP Agent: Windows 2000, 2003, and XP.

### Sample File Reader Logs

The files shown in the slide on the left are provided as an example of how logs collected by File Reader might look.

These were collected from Hewlett Packard's SiteScope monitoring tool.

S  
E  
E  
T  
O  
N

## Universal Device Collection Configuration Manage ODBC Types

Delete	Name	Description	Device Name	Device Description
<input type="checkbox"/>	ActivIdentity	Collects events from ActivIdentity ActivCard AAA Server	ActivIdentity ActivCard AAA Server	
<input type="checkbox"/>	CiscoNCM	Network Compliance Manager	CiscoNCM	Network Compliance Manager
<input type="checkbox"/>	EMCDocumentum	EMC DOCUMENTUM	emcdocumentum	EMC DOCUMENTUM
<input type="checkbox"/>	enVisionSample	Sample type that finds new devices as they are added to the enVision database.	enVision	Reads new devices from the device_list table.
<input type="checkbox"/>	ePolicy	McAfee ePolicy Orchestrator	McAfee ePolicy Orchestrator	
<input type="checkbox"/>	ePolicy36X	McAfee ePolicy Orchestrator 3.6 or 3.6.1	McAfee ePolicy Orchestrator	
<input type="checkbox"/>	ePolicy4.5	McAfee ePolicy Orchestrator	epolicy	McAfee ePolicy Orchestrator
<input type="checkbox"/>	ePolicy4.x	Audit events from McAfee ePolicy Orchestrator	McAfee ePolicy Orchestrator	
<input type="checkbox"/>	ePolicyvirus	McAfee ePolicy Orchestrator	McAfee ePolicy Orchestrator Virus	
<input type="checkbox"/>	ePolicyvirus4.5	Anti Virus Events from McAfee ePolicy Orchestrator	McAfee ePolicy Orchestrator	
<input type="checkbox"/>	ePolicyvirus4.x	Anti Virus Events from McAfee ePolicy Orchestrator	McAfee ePolicy Orchestrator	
<input type="checkbox"/>	FoundScan	McAfee Foundscan	McAfee FoundScan	
<input type="checkbox"/>	HIPSEX	McAfee Host Intrusion Prevention Server	McAfee Host Intrusion Prevention Server	

35

## Configuring ODBC Types

Manage ODBC Types - Add/Modify Universal ODBC Definition

Name:

Description:

Device Name:

Device description:

Tag:

Default interval:  Seconds

Output field delimiter:

Data query:

Address column:

Level column:

Event ID column:

Max tracking query:

Tracking column:

36

## Universal Device Collection Configuration, continued

### Manage ODBC Types

There are also a number of pre-configured ODBC Types available in enVision. You can view their configurations by clicking on the hyperlink in the Manage ODBC Types configuration window shown in the slide.

### Configuring ODBC Types

Use the Add/Modify Universal ODBC Definition window to add a new ODBC definition to the NIC ODBC Service or to modify an existing ODBC definition.

Data Field	Explanation
Name	The name of the ODBC Type. This should be the event source name for which the type is created. (required)
Description	Description of what this Type supports. (optional)
Device Name	Name of the event source or product that this Type collects from. (optional). This is the same name that is in the Windows Configuration system data source.
Device description	Description of the event source that this Type collects from. (optional)
Tag	The tag added to the syslog message to identify the event source. The tag is prepended to the syslog in the format, %device tag, for example, %CELERRA. (required)
Default interval	Interval in seconds that the NIC ODBC Service waits between connections to the device to retrieve data. This is the default for all instances of the ODBC Type, but may be overridden for each individual ODBC Definition for the NIC ODBC Service. (required)
Output field delimiter	String to use between each field when writing out the event. (required)
Data query	Query to be performed against the event source's database via ODBC to extract the identified fields from the database. The data query must include the clause '%TRACKING%'. (required)

# SQL WE T O N

## Sample ODBC File

```
Feb 26 13:28:34 [10.1.14.5] %BSAFE_AUDITLOG: SAMPLE
||20091229||73849||180575||          ||010018002064||QUSER
||QZDASOINIT||861022||S||46||Bsafe authorizes access to
odbc data base          ||O||DATABASE SQL REQUEST||ZDAQ0100
||40||READ              ||          ||          ||
||SELECT 'RECORDS=' || COUNT(0) FROM OUTBND EVT WHERE
EVENT STATUS = 0 AND EVENT_TIME < CURRENT_TIMESTAMP - 60
SECOND
||
||GR||APPACCTS          ||20100226073849
```

```
Feb 26 13:28:34 [10.1.14.5] %BSAFE_AUDITLOG: FTPUSAGE
||20091229||73849||180576||          ||010018006123||QTCP
||QTFTP02846||858829||S||15||Bsafe authorizes access to Ftp
request                  ||F||FTP REQUEST          ||VLRQ0100
||7||RECEIVE FILES       ||ICSMISC  ||CCCDRS  ||
||/QSYS.LIB/ICSMISC.LIB/CCCDRS.FILE/TQU0124791.MBR
||
||GR||APPACCTS          ||20100226073849
```





## Universal Device Collection Configuration, continued

### Configuring ODBC Types, *continued*

Data Field	Explanation
Address column	Location of the Source IP address - address from which the event has been sent. If not specified, the service uses the address defined in the ODBC Definition. Valid values are valid IP addresses. Do not use the hostname. (optional)
Level column	Used as the level when constructing the event header. If not specified, the level is left out of the event header. Valid values are 0 to 7. Using any other level causes the message to be dropped. (optional)
Event ID column	Use as the event ID when constructing the event header. If not specified, then the ID is left out of the header. (optional)
Max tracking query	Query to be performed to determine the highest number in the tracking column.  Example:  <code>SELECT MAX (RowNumber) FROM master.dbo.tracelog</code>
Tracking Column	Column to be used to differentiate rows that have been read, and rows that have not. This allows you to know what data has previously been read out of the database.  Example:  <code>RowNumber</code>

### Sample ODBC Logs

The file shown in the slide on the left is an example of how logs collected by ODBC might look.

# SECTIONS

## Multiple Protocols or Multiple Event Sources

- ▶ A product may have multiple logging mechanisms
- ▶ A system may have multiple event sources
  - Configure as: Multi-Device
  - Seen as individual event sources
  - Counted toward the license

## Multiple Protocols or Event Sources

Some devices provide log information in more than one format. For example, a Cisco Content Engine appliance sends its web traffic logs using FTP and its system audit logs via syslog. enVision can easily handle this by defining both sets of logs in the same XML file. The administrator must first configure the device to send both logs to enVision, then configure the NIC File Reader Service to accept logs from that device. This collects and supports both formats from that device. The headers in the XML file discovers the Cisco Content Engine and recognizes both log message streams.

In other cases, a single system may run multiple applications from which a user wants to collect logs. A popular case is any application running on top of a Microsoft Windows platform. While they may already collect the logs from the operating system itself, they may also be interested in the logs generated by the application running on the same box. enVision handles this by allowing the administrator to configure this device as a Multi-Device. This setting tells enVision to look for more than a single device's logs from that IP address. As enVision matches log events from other devices from that IP, it will add the appropriate devices to the discovered list automatically.

Additionally, as event sources are configured at a particular IP address, they will display in Manage Monitored Devices as separate devices, which count against the upper limit on the enVision license.

SEVENTH

## Extracting Log Data

- ▶ Logs collected by SNMP, File Reader and ODBC must be extracted at the command line
  - Use the lsdata utility in ..\<nodename>\bin
- ▶ Help for lsdata is only available at the command line:
  - Overall: >lsdata -help ? or just >lsdata
  - Device help:>lsdata -help devices
  - Event help:>lsdata -help events



## Isolating and Extracting Specific Logs from the IPDB

- ▶ IPDB contains all logs from all event sources
  - ESI is only concerned about those logs from the event source under study, which will be included within “unknown”
- ▶ How to use only those logs from the new source?
  - Use lsdata utility to copy them into a separate syslog file
  - Command and arguments: >lsdata -events syslog -time start end -devices <devicespec> >><newfilename>

```
Command Prompt
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.
C:\Documents and Settings\amidod>cd c:\nic\4000\USPSAMIDDLIC\bin
C:\nic\4000\USPSAMIDDLIC\bin>lsdata -events syslog -time hour hour -devices solaris >>sampleSolaris.unx
C:\nic\4000\USPSAMIDDLIC\bin>
```

## Extracting Logs

To create the definitions that RSA enVision will use to monitor the events from an event source, you must obtain a log file from the event source that you wish to integrate with enVision, as previously described.

If the collection service that you configured when you set up the event source in enVision is syslog, you can use a log file generated by the event source to create or edit an event source XML file. If you configured any other collection service, you must use enVision's `lsdata` utility to extract the log data.

Although the command and arguments are provided, this process is only documented via the `lsdata` utility's command line interface help "man" pages.

The process of collecting logs from the methods previously discussed and the `lsdata` conversion may result in issues that are unique to your particular situation. In such cases, it is worth considering the balance between troubleshooting the process yourself, or arranging for a Professional Services engagement.

### Isolating and Extracting Logs from the IPDB

The `lsdata` utility allows you to extract information out of the IPDB.

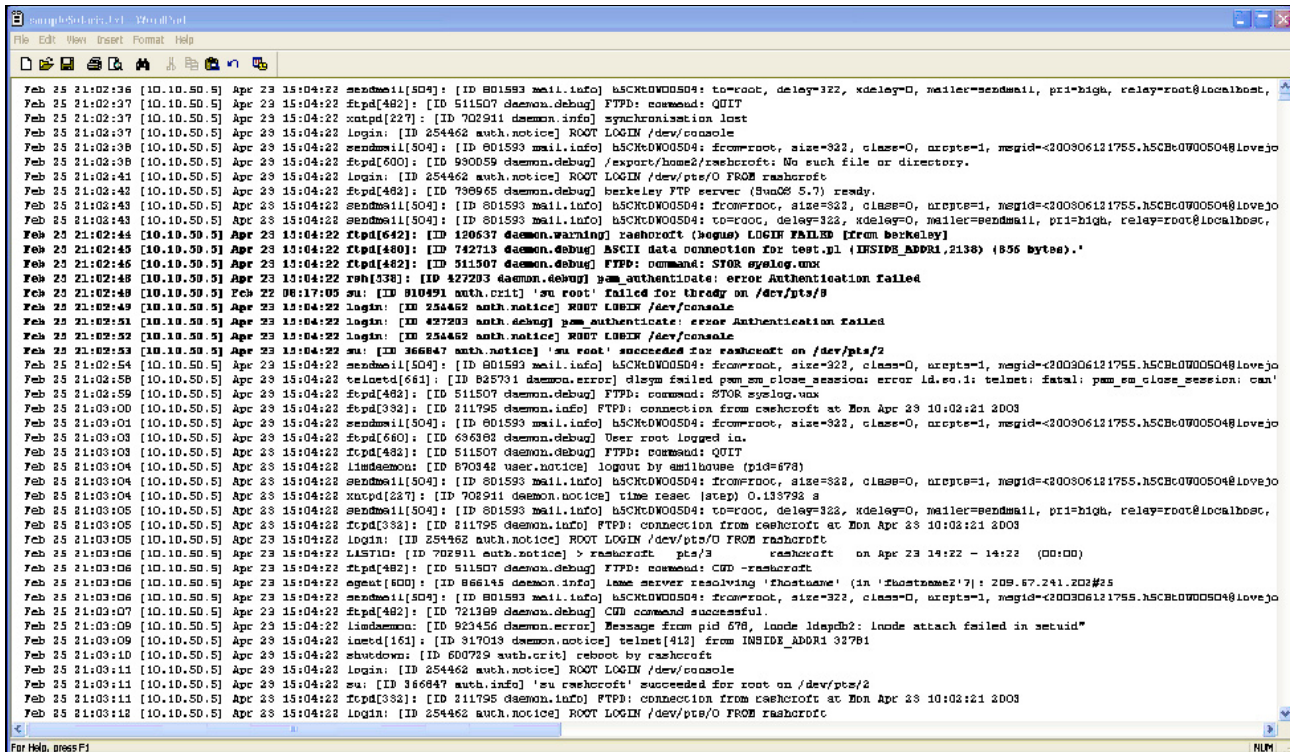
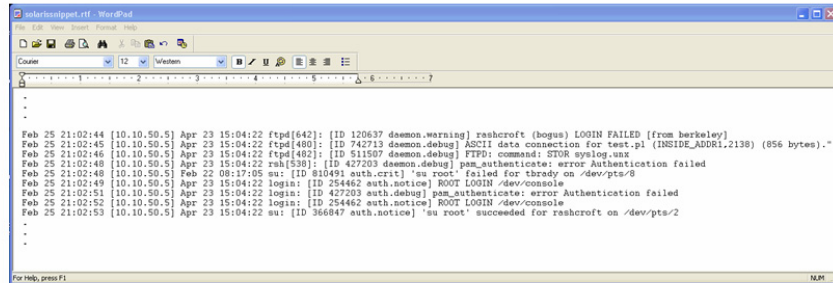
The screen capture of the `lsdata` command on the left depicts isolating events in the IPDB (syslog) over the last hour (hour hour substituting for an actual start and end time) from devices named solaris and copying (>>) them into a new file named `sampleSolaris.unx`.

The choice of the `.unx` suffix is important for retaining the syslog format of the logs. This file will then be found in the `.\bin` directory, as we have not specified any other location.

You would then use the new file to investigate all the events and proceed with the next step in the process using this file for input into the EventSource Integrator.

## Uncovering Patterns and Similarities

- ▶ Open extracted log file in a text editor
- ▶ Visually scan down through logs
- ▶ Identify groups of similarly-appearing log formats
- ▶ Observe within a given message format how there may be values that change, and those that may not



## Uncovering Patterns

With the logs extracted into a single file, this file can be opened in a text editor to show the individual log messages. Many of these log messages will be so long that they will either run off the window of the text editor, or, wrap to the next line. In the example shown at left, there are a few lines of the sampleSolaris.unx log that fit in WordPad's window. Even among these few entries, you can determine similarities and patterns, to be more thoroughly examined and used in the ESI process.

In this excerpt from the sampleSolaris.unx file there are two logs that contain **su:** in the middle of the line. While there may be further similarities between these two messages, they also have differences. One of these differences is the username, tbrady versus rashcroft.

Additional examples of patterns that might be grouped together are those that begin with: **ftpd:**, **login:** and possibly **rsh:**.

### Sample Solaris File

At this point, it is not necessary to understand the meaning of the logs, but just to group them according to appearance. It will be helpful to have a significant collection of logs, representing as many events as could occur at the event source, thus capturing as many of these patterns as possible.

The entire sample Solaris file contains hundreds of logs. The screen shot on the left contains a more extensive view of that log. The logs that are in bold font are the nine lines used in the slide above.

# SECTION



## Uncovering Patterns, continued

### Tips

Here are some tips when searching for patterns:

- In any particular grouping of messages, you should compare one to another to see if there are any recurring patterns of a character or characters.
- Pay particular attention to characters like brackets, commas, semi-colons and colons.
- Space between characters is also relevant.
- Notice where values change.

An example of this is seeing the presence of an IP address and Port, indicated by a dotted decimal notation followed by a colon and a decimal number.

The IP address should vary from one message to the next, and may even appear in a different position within other messages. This is the first step toward understanding the operation of log formats - seeing what changes (variables) and seeing what does not (fixed or static values).

# SECTION

## Exercise 2 Obtain Log Data

### The goal of this exercise is:

- To examine logs

This exercise consists of three parts:

- Simulate transmission of log data from the unsupported event source
- Export log data for the unknown event source
- Look for patterns in a collected log file

### Scenario

Bulldog Systems is a manufacturer of a wide range of IT products. Like many of its competitors (think: Cisco), Bulldog has acquired and rolled up many startups and now has products in: Access Control - DogBite; Data Loss Prevention - SniffIT; Intrusion Detection Systems - Growl; Authentication Servers - MarkIT, etc. However, Bulldog System's premier product has been their firewall proxy server product - DogHouse.

This relatively new category of IT hardware is sometimes referred to as an application layer firewall and is intended to provide a variety of functions on a single platform, going beyond the basic packet filtering features found in firewalls, to running HTTP, FTP or SMTP processes as if they were running on hosts. You researched the vendors and bought the Bulldog DogHouse and put one into production on your Company's IT infrastructure.

Since you have had enVision for about six months, you'd now like to bring the DogHouse into enVision to be monitored and analyzed, using Queries, Reports and Alerts.

The one problem is that DogHouse is so new that it is not a "supported" device within enVision. To further complicate your life, you have just found out that Bulldog's formal documentation about the content, format and function of their logs is virtually non-existent. They did explain that the logfiles are in syslog format and can be configured to be sent via UDP on port 514.

Therefore, you need to build your own XML files. You will re-create this logfile in enVision by using the Data Injector utility and a source file named: unknownSyslog.unx. This is the original file from which you will extract a few messages. You will name the new, abridged file: syslogDogHouse.unx so that you know it is a syslog collection that belongs to your new firewall proxy server.

## Exercise 2, continued

### Inject Log Data

Follow these steps to inject log data and confirm that enVision detects the event source type of unknown.

Step	Table
1	Inject the log data by running the file: <b>c:\class\injector\esisysloginjector.bat</b>
2	<p>Check that the device has been discovered as an unknown device as follows:</p> <ul style="list-style-type: none"> <li>• Navigate to the <b>C:\nic\4000&lt;nodename&gt;\tmp\nuggets</b> directory to confirm that a new folder has been created with the name unknown.</li> <li>• Navigate to <b>C:\nic\lnode\data&lt;nodename&gt;</b> to confirm that an unknown folder was created there.</li> </ul> <p>At this point, you know that messages are being received by enVision and that enVision does not (yet) recognize the source device(s) messages.</p> <p><b>Note:</b> It may take a few minutes for each of these folders to be created.</p>
3	To see the table that was used to parse the event data select <b>Overview &gt; System Configuration &gt; Manage Monitored Devices</b> .
4	<p>Check that data can be monitored using event viewer, as follows:</p> <ul style="list-style-type: none"> <li>• Log in to enVision</li> <li>• Select the <b>Analysis</b> tab and select <b>Event Viewer &gt; Message View</b></li> <li>• Click the <b>Clear Events</b> button</li> <li>• Set the Timeframe dropdown to the past <b>30 minutes</b></li> <li>• For Device Type, select <b>unknown</b> <ul style="list-style-type: none"> <li>— <b>Note:</b> Event sources other than nic are only selectable after they have been discovered by the appliance.</li> </ul> </li> <li>• Click the <b>Update Now</b> button</li> </ul> <p><b>Result:</b> The unknown devices appear in the view.</p>

## Exercise 2, continued

### Export the Log Data and Look for Patterns

In this activity, you will use the `lsdata` utility to move the collected messages into a separate file, from which you will categorize them into the form that enVision recognizes. Then you will look at the file to identify patterns that can be used to determine how to parse a message.

Follow these steps to export the log data from the unsupported DogHouse event source.

Step	Action
1	Open a command prompt and change directory to: <b>c:\nic\4000\<i>nodename</i>\bin</b>
2	Enter the following command: <b>&gt;lsdata -events syslog -time start end -devices unknown &gt;&gt;syslogDogHouse.unx</b>  <b>Result:</b> This extracts all events collected into the syslog from any event source type <b>unknown</b> and stores them in a new file, <b>syslogDogHouse.unx</b> . This file will be used to identify specific messages coming from the unknown event source(s) and for adding messages to the XML file.  <b>Note:</b> In a real-world scenario, you might use a timeframe in place of hour to collect events for a specific timeframe. For example, in our scenario we have had enVision for six months, so we may want to collect data from the past six months.
3	Open the <b>syslogDogHouse.unx</b> file with a text editor, such as, Notepad++ or WordPad to view its contents.  Look for patterns in the file, such as recognizable words like Connection, Request, Established, Terminated, IP Addresses, Port Numbers. These may be variables. Look for sets of characters that reappear in different log entries and appear to be commonly shared. These might represent fixed text.  <b>Note:</b> <code>syslogDogHouse.unx</code> is a simplified log file and may not be representative of a typical log. This log is intentionally simplified to help you distinguish logical patterns so that you can identify the parts of the message.

# SECTION

---

# How RSA enVision Interprets Logs

---


## Objectives

After completing this unit, you should be able to:

- Identify the header and message content in a log message
- List enVision's event source classes
- Describe how to determine the appropriate Database Table
- Identify the variables in a Database Table
- Determine the information needed for a specific message and identify the applicable table

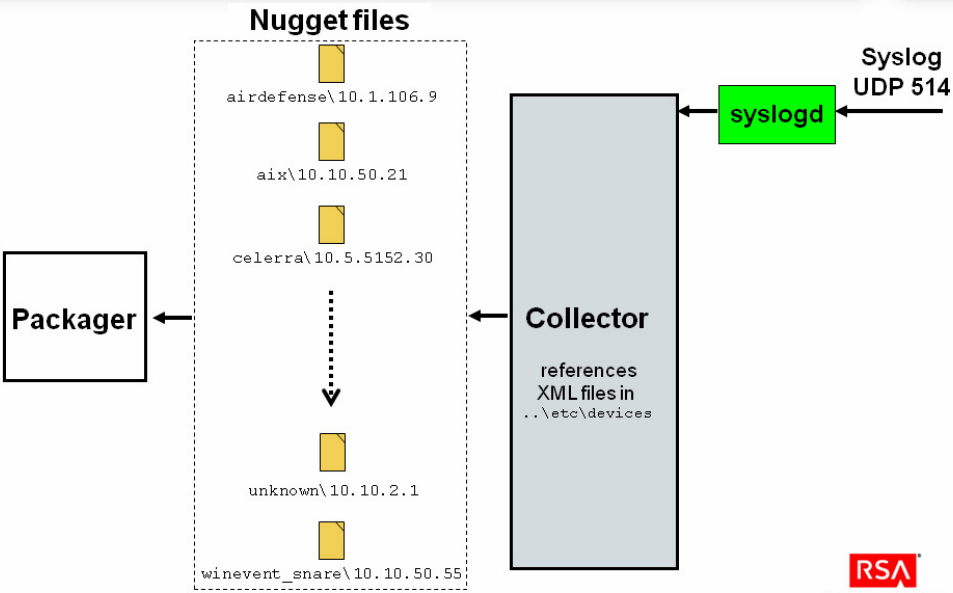
### Data Parsing

- ▶ The enVision parser uses NIC Device Markup Language to identify and process data contained in the log messages
- ▶ Logs are made up of fixed text and variable data
- ▶ Parsing flags are used to identify variables
  - Variables are dependent on the Device Type (Device Class/ Device Subclass or simply Event Source class)
  - Variables are also dependent on the Database Table used
- ▶ When a new XML file is being created, these parsing flags will need to be identified and defined



The Security Division of EMC 45

### Data Parsing, continued



**Nugget files**


- airdefense\10.1.106.9
- aix\10.10.50.21
- celerra\10.5.5152.30
- ...
- unknown\10.10.2.1
- winevent\_snare\10.10.50.55

**Collector**

references XML files in ..\etc\devices

**syslogd** ← Syslog UDP 514

**Packager**



The Security Division of EMC 46



## Data Parsing

Parsing is the process of determining the syntax and meaning from text - or in our case - the strings contained in log messages. This is done primarily at the Collector as it references a collection of files, including the event source XML file for a particular event source.

A log message is comprised of components that are put together by both the source and destination systems in order to provide notification of an event that occurred on the source system. The RSA enVision NIC DML (Device Markup Language) uses an XML file that describes data contained in a specific event source's log message. DML is used to provide a description for the entire message format of a given event source's set of possible messages based on specific rules.

The operating copy of the device DML description file is located in the associated production area, specifically the `..etc\devices` directory.

DML uses a series of variables known as parsing flags to identify and understand known values. Each flag represents a parsing and handling instruction for the identified data within the message. These handling instructions give the parser critical information about what type of data it is, if parsing is necessary, and what fields the parser needs to update in the database table to add this information.

These parsing flags vary by event source type and table to which the data is parsed. A group of the most common flags are valid across all event sources and tables and include information such as Device Address, Time Stamp, Message ID, and Message Type. Other flags are more specific and have meaning to only a few event source types or tables. Examples of these are Group, Profile, and Project, which are valid parsing instructions for configuration management event source and messages to be parsed to the User Activity table.

In order to understand the strings, we will look at them in the context of Device Type and requirements based on different Database Tables.

### Data Parsing Flow

The data flow slide on the left shows the Collector and the XML files. This is where enVision's parser uses NIC DML to process the received log message against the XML files to make sense of the log message contents.

SE  
E  
T  
O  
N


### Log Message Contents *syslog and Log Message Header*

- ▶ Syslog header - on the wire:  

```
<34>1 2010-02-22T08:17:05.52-4:00 acomputer.rsa.com  
evtslog - ID810491 - BOM'su root' failed for tbrady on  
/dev/pts/8
```
- ▶ Syslog header - after syslogd:  

```
Feb 22 08:17:05 [10.10.50.5] Feb 22 08:17:05 su: [ID  
810491 auth.crit]'su root' failed for tbrady on  
/dev/pts/8
```
- ▶ UNIX Log message header:  

```
Feb 22 08:17:05 [10.10.50.5] Feb 22 08:17:05 su: [ID  
810491 auth.crit]'su root' failed for tbrady on  
/dev/pts/8
```



The Security Division of EMC 47

### Log Message Contents *Header and Payload*

- ▶ Header  

```
Feb 28 17:31:29 [10.10.18.1] %PIX-1-101001:(Secondary)  
Failover cable OK.
```
- ▶ Payload  

```
Feb 28 17:31:30 [10.10.18.1] %PIX-1-101001:(Secondary)  
Failover cable OK.
```
- ▶ Header and Payload overlap  

```
Feb 28 17:31:30 [10.10.18.1] %PIX-1-101001:(Secondary)  
Failover cable OK.
```



The Security Division of EMC 48

---

# Log Message Contents

## Header and Payload

Typically a log message consists of two main elements: a header and a payload. The header is located after the syslog fixed components and before the message identifier. The header identifies what the message is and where it starts.

The first item in the slide at upper left shows the header and the second shows the payload. In some events, as in the third example, the payload can start in the header and there can be overlap of the header and payload.

## syslog and Log Message Header

Sometimes understanding how the information from the event makes it into either the log message header and payload can be difficult, as in the log message example shown at lower left. For example, how did auth.crit get into the payload?

SE  
FT  
ON  
N

## Log Message Contents Header

- ▶ Can contain fixed text and variables
- ▶ Must contain a <messageid> that matches an identifier in the message
- ▶ Always contains a <!payload>
- ▶ A single header can be used for multiple messages



## Header Examples

**Log:** Jul 21 09:59:57 [LOG\_ALERT] PAM\_httpd[384]: check pass;  
user unknown

**XML:** <HEADER  
id1="0001"  
id2="0001"  
content="&lt;month&gt; &lt;day&gt; &lt;time&gt;  
[LOG\_&lt;messageid &gt;] &lt;!payload&gt;" />

**Log:** %PIX-2-106007: Deny inbound UDP from 12.15.105.5/8 to  
192.168.1.5/9 due to DNS flag

**XML:** <HEADER  
id1="0001"  
id2="0001"  
content="%PIX-&lt;level&gt; -&lt;messageid&gt; :  
&lt;!payload&gt;" />



# Header

## Header Contents

The content defines the layout of the header. The content:

- Can contain fixed text and variables.
- Must contain a <messageid> variable that matches up with an identifier in the messages themselves.
- Always contains a <!payload>. The <!payload> represents the remainder of the message.

---

Note: The “!” character is required as it makes this tag a “marker.” An XML marker is a reference to another point in the XML file. If “!” is not present, then <payload> would just be a regular variable indicating the end of the message.

---

A single header can be used for multiple messages.

## Header Example

The two examples on the slide on the left show the relationship between a variable in the log and the <messageid> variable in the header tag.

In the first example, from an original Nokia IPSO Firewall event message, the header begins with a Time Stamp and [LOG\_ALERT]. However, the messageid is just ALERT and the payload begins after the ] bracket for the remainder of the characters making up the message.

The second example is from a Cisco PIX Firewall event message and demonstrates where a variable appears in the header. In this case, it is the number located between the dashes (2). This variable will be identified as the level or severity value. The payload for this message begins after the colon and includes the remainder of the characters making up this message.

It is important to choose the Message ID as concisely as possible. Otherwise, many headers might have to be created. For example, had the Message ID been **ALERT] PAM\_httpd[384]** instead, then only those messages that contained this exact string of characters could have been parsed by this one header. Since the number inside the brackets ([ ]) changes, there would need to be another header defined for each change. (Or, that number could be chosen as a header variable representing something like sessionID or processID.)

S  
E  
E  
T  
O  
N

## Defining the Message ID in the Header

- ▶ Identifying the Message ID is the most important part of designing an event source file
  - The Message ID should represent the focus of the message – what do you want to report on?
- ▶ The Message ID component of the Header is the `<messageid>` variable
- ▶ The Message ID becomes the `id2` parameter in each corresponding message definition that parses to that Header



## Message ID Example

```

%PIX-2-106007: Deny inbound UDP from 12.15.105.5/8 to
192.168.1.5/9 due to DNS flag

<HEADER
  id1="0001"
  id2="0001"
  content="%PIX-&lt;level&gt;-&lt;messageid&gt;: &lt;!payload&gt;" />
  .
  .
<MESSAGE
  level="2"
  eventcategory="1803000000"
  parse="1"
  parsedefvalue="1"
  tableid="9"
  id1="106007"
  id2="106007"
  summary="NIC_B_FW_ADDR_ACCOUNTING;fields=0,0;key=faddr;sumtype=ciscopix_d
eniedIncomingConn;|NIC_B_FW_PORT_ACCOUNTING;key=fport;sumtype=ciscopix_deniedIncom
ingConn;|NIC_B_DEVICES;sumtype=ciscopix_deniedIncomingConn;"
  content="&lt;@inout:*DIRCHK(faddr) &gt;&lt;@ntype:5&gt;&lt;@ntype:*ifSecur
ityDirOutBound(inout)&gt;&lt;@action:connection denied&gt;&lt;@reason:due to
DNS&gt; Deny inbound &lt;protocol&gt; from &lt;faddr&gt;/&lt;fport&gt; to
&lt;laddr&gt;/&lt;lport&gt; due to DNS &lt;reason&gt; " />
    
```

52

---

# Message ID

## Defining the Message ID in the Header

Identifying the Message ID is the most important part of designing an event source file. The Message ID should represent the focus of the message – what do you want to report on?

The Message ID component of the Header is the <messageid> variable.

The Message ID is the id2 parameter in each corresponding message definition that parses to the Header.

## Message ID Example

The slide shown on the lower left provides a further look at the relationship between the Header Message ID and the actual message tag. This is the same PIX event from the previous header example. Notice that the Message ID in the header (106007) becomes the id2 variable – Vendor Message ID - in a message tag.

It is not uncommon that both the id2 and the id1 fields have the same value, so the NIC Message ID for this is the same as Cisco's.

S  
E  
E  
T  
O  
N

## Device Discovery, Status and Analyze Event Source at 10.10.50.5

System Configuration - RSA enVision - Windows Internet Explorer

Overview Alerts Analysis Reports

Dashboard

System Performance

Best Practices

System Configuration

Devices

- Manage Monitored Devices
- Manage Device Group Filters
- Manage Device Attribute Definitions
- Import/Export Device Attributes
- Manage Device Types

Messages

- Manage Messages
- Manage Messages To Parse
- Manage Categories
- Manage Variables
- Set Up Undefined Label

Directories

- Users
- Services
- Dashboard Items

Use this window to display the list of devices being monitored.

**Manage Monitored Devices**

Filter: WHERE IP Address IN '10.10.50.5'

Delete	Operator	Attribute	Comparison	Criteria
<input type="checkbox"/>	WHERE	NIC Properties / IP Address	IN	10.10.50.5

Group By: None

Apply Add Delete

Filtered Devices: 1 Device found

Select	IP Address	Name	Device Type	Site/Node	Status
<input type="checkbox"/>	10.10.50.5	acomputer.rsa.com	UNIX Solaris	USPSAMDDDL1C / USPSAMDDDL1C	Active <input checked="" type="checkbox"/>

**RSA**  
The Security Division of EMC 53

## Details About UNIX Solaris at 10.10.50.5

System Configuration - RSA enVision - Windows Internet Explorer

Overview Alerts Analysis Reports

Dashboard

System Performance

Best Practices

System Configuration

Devices

- Manage Monitored Devices
- Manage Device Group Filters
- Manage Device Attribute Definitions
- Import/Export Device Attributes
- Manage Device Types

Messages

- Manage Messages
- Manage Messages To Parse
- Manage Categories
- Manage Variables
- Set Up Undefined Label

Directories

- Users
- Services
- Dashboard Items
- Watchlists
- Set Up System Performance

Vulnerabilities

Assets

Task Viewer

Use this window to modify a device's options and attributes or to add a new device.

**Manage Monitored Devices - Add/Modify Device**

Site: USPSAMDDDL1C IP address: 10.10.50.5

Node: USPSAMDDDL1C Device class: Host/Unix

Discovery: 2010-02-21 20:55:15.896 Device type: UNIX Solaris

Analyze:  Collection: Active

Multi device:  Has timestamp:

Remove relay headers:  Use timestamp:

Encoding: 65001 (UTF-8)

Properties: ResolvedName = acomputer.rsa.com

Location:

Organization:

Owner:

Physical:

Function:

Importance: Value = 1

Vulnerability: Value = 1

Zone:

SystemInformation:

Apply Cancel

54



# Event Source Discovery, Status, and Analysis

## Event Source

By parsing the log message contents against the header and message tags in the XML files, enVision discovers and identifies the event source where the log message came from. In the slide on the upper left, the IP address is taken from the IP packet and the Device Type is determined from the contents of the solarismsg.xml file.

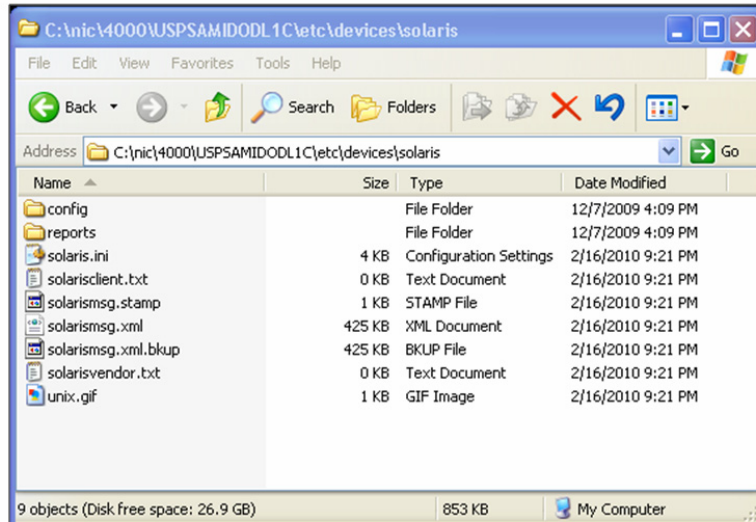
## Event Source Details

The slide shown on the lower left confirms some of the information we have already seen in other screens, including the Encoding (UTF-8) scheme mentioned in Unit 1.

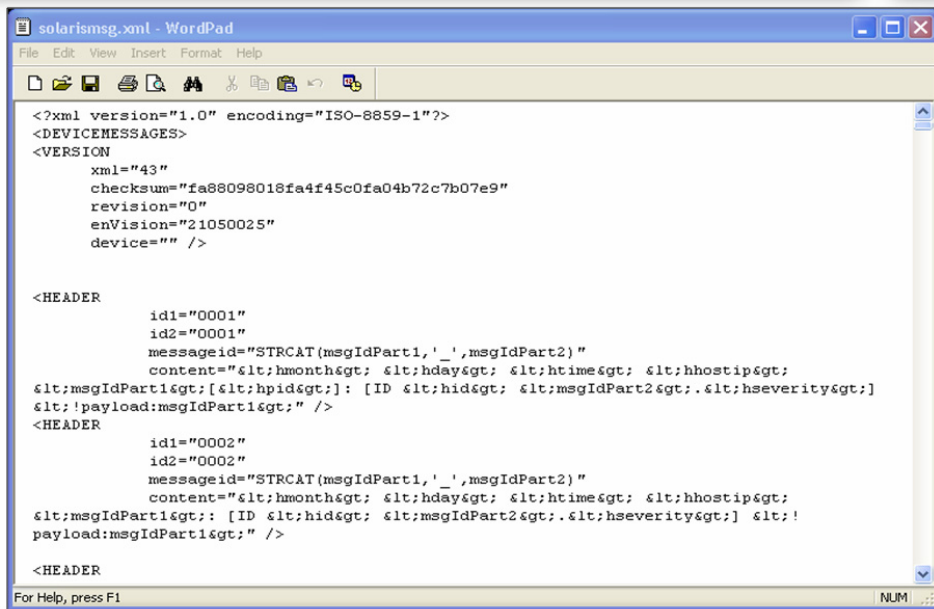
This screen should be familiar from the enVision Administration and Operations class. It is the result of clicking the IP address hyperlink in the previous window.

S  
E  
E  
T  
O  
N

## enVision Device Files *Solaris Directory and its Contents*



## Solaris XML File Detail



## Event Source Discovery, Status, and Analysis, continued

### Event Source Files





The slide on the upper left shows the directory and files referenced by enVision's parser to allow the event source to be discovered in enVision as seen on the previous two slides.


More specifically, the slide shown on the lower left is the XML file that was used by NIC DML to parse the messages.

The slide on the lower left depicts the result of highlighting the solarismsg.xml file and opening it with Wordpad.

SEFTON





### Device Class, Device Subclass and Event Source Class


- ▶ Pre-enVision 4.0:
  - Device class
    -  Host,  Network,  Security,  Storage
  - Device subclass
    - Examples: UNIX, Windows, Router, Database, Firewall, IDS
  - Nomenclature examples:
    - Host.Unix, Network.Router, Storage.Database, Security.Firewall, Security.IDS
- ▶ enVision 4.0 and beyond:
  - Event Source class
  - Nomenclature examples:
    - Host.Unix, Network.Router, Storage.Database, Security.Firewall, Security.IDS



The Security Division of EMC 57

### Event Source Classes

 <ul style="list-style-type: none"><li>Host.Application Servers</li><li>Host.Load Balancing</li><li>Host.Mail Servers</li><li>Host.Mainframe</li><li>Host.Midrange</li><li>Host.Unix</li><li>Host.Virtualization</li><li>Host.Web Logs</li><li>Host.Windows Hosts</li></ul>	 <ul style="list-style-type: none"><li>Security.Access Control</li><li>Security.Analysis</li><li>Security.Anti Virus</li><li>Security.Application Firewall</li><li>Security.DLP</li><li>Security.Firewall</li><li>Security.IDS</li><li>Security.IPS</li><li>Security.Physical Security</li><li>Security.VPN</li></ul>
 <ul style="list-style-type: none"><li>Network.Configuration Management</li><li>Network.Router</li><li>Network.Switch</li><li>Network.System</li><li>Network.Wireless Devices</li></ul>	 <ul style="list-style-type: none"><li>Storage.Database</li><li>Storage.Storage</li></ul>



The Security Division of EMC 58

---

# Classes and Subclasses

RSA enVision originally grouped all supported devices into device *classes* and device *subclasses*. The device classes represented the general function of the devices. This provided the framework for organizing devices and the information from devices. enVision assigned each device subclass to only one device class. If the classification was not explicit, enVision assigned the subclass to the Security device class.

Although the general function of such classes is ongoing, RSA is transitioning the terminology from device to the more descriptive event source, and this will be reflected in the user interface in subsequent releases, SPs and/or updates. The enVision functionality does not change, just the way it is referred to in documentation and the user interface changes.

In current and future practice, a particular event source will be assigned to an event source class, which is a concatenation of the old device class and device subclass.

## Event Source Classes

The slide on the left lists the event source classes that are used now. There may be additional categories added in the future through content updates and/or product releases.

# SEVENTH ON

## Identifying the Class of Your New Event Source

- ▶ Much about the event source (device) is known
  - Function: what it does on a high level, e.g., router, host, firewall
  - The IP address configured into it
- ▶ Choose an event source class that seems most appropriate
  - Each message represents a different event, so each one can be handled uniquely if necessary
- ▶ Simple

## Classes and Subclasses, continued

### Identifying the Class of a New Event Source

Choosing an event source class is not at all difficult. Although enVision may consider this an unknown, we know more about it than enVision does at this point. We can easily assign it to an event source class based on what it does.

The choice of class does not limit you to only using one Database Table, though, as each log message can be assigned to any table.

S  
E  
T  
O  
N

## enVision's Database Tables

- ▶ General or Vendor-specific IT infrastructure elements
  - General: Access Control, Antivirus, Configuration Management, Database, Firewall, IPS, Mainframe, Network, Router, Storage, Switch, UNIX, Virtualization, VPN, Web
  - Specific: Alerts (enVision), Check Point, DLP (RSA), Global (enVision), ISeries (IBM), NIC (enVision), Windows (Microsoft)
- ▶ Common themes:
  - Accounting, Security, System, Summary, etc.
- ▶ Total of 100 Database Tables (as of March 2010)
  - Examples: Access Control Accounting, CheckPoint Audit, Firewall System, IPS Intrusion, ISeries Audit, Global Summary, NIC Performance, Storage Activity, UNIX Authentication, UNIX Performance, VPN Tunnel Summary, Web Accounting, Windows Accounting



## Database Tables and the XML variables

- ▶ Choose a Database Table (or Tables) that seems to be appropriate for the characteristics and functions of the event source
- ▶ Creating the XML file requires parsing flags - variables - to be identified





---

## Database Tables

### enVision Database Tables

RSA enVision organizes the Database Tables into general topics such as Access Control, Antivirus, Configuration Management, Database, Firewall, IPS, Mainframe, Router, Storage, Switch, UNIX and VPN, as well as those topics that are more vendor-specific, such as Check Point, ISeries, NIC (that is, enVision), and Windows.

Under these groups, information might be found within the messages that share a common interest. Therefore, there is another level of commonly-occurring themes such as Accounting, Security, System, and so forth.

The results that will be included in any particular Database Table can be researched using the Query tool, under the Select table to query: menu at Create New Query. Each choice will create and display the Fields included in that particular Table.

As of March 2010, there are 100 Database Tables. This number will increase as new Tables are developed by the enVision engineering team. Any and all new Tables are released by way of the monthly Event Source Update process on RSA's SecurCare Online.

### Database Tables and XML Variables

As pointed out earlier, how log messages are further decoded depends on which Database Table you choose, as there are different variables used in different tables.

This process is similar to choosing an event source class in that there are general categories of Tables, and there are general topics of interest.

# SEARCH TIPS

## Selecting a Database Table *Various Methods*

- ▶ Research with the Query tool
  - Choose a likely Database Table
  - Look at Column Names, which may contain the desired information
  - Run the Query to confirm
- ▶ Use enVision Online Help
  - Search for: Database Tables
- ▶ The most up-to-date listing of all the enVision Database Tables is in the `..\etc\sqltbl` directory
  - These may be updated by the monthly event source Updates
  - But, these files are not easily interpreted (at first)
  - DO NOT MODIFY

## Database Tables, continued

### Ways to Select a Database Table

There are several methods that you can use to find an appropriate database table. Using the query or the online Help provides you with information that you can use to select a table. However, these resources may be limited due to the fact that RSA releases Event Source Updates on a monthly basis which may contain new Database Tables, that may not appear in the Help.

You can still use the Query tool to see what fields will be used, but a new Table may introduce a new field. This field is subsequently represented by a new variable in the XML file.

To find the internal reference from a Query Column to an XML variable, you can also look in the `..etc\sqltbl` directory, which includes the Database Table structures.

**IMPORTANT:** Do NOT alter the content of any of these files. Editing these files may adversely affect your enVision installation.

S  
E  
E  
T  
O  
N

### enVision Query Tool UNIX Security Database Table

The Security Division of EMC 63

### enVision Query Results UNIX Security Database Table

Review the Column Headers:

- ▶ These fields contain information about:
  - DeviceAddress, DeviceHostName, Date/Time, MessageID, ForeignPort, Username, Username1, Uid, Interface, Action, Agent, Reason
- ▶ Note the "MessageID" value

The Security Division of EMC 64

---

## Database Tables, continued

### enVision Query Tool Example

Returning to our example log message, we already know that the message shown in the slide on the upper left is from a UNIX Solaris system.

Now we want to see which Database Table would reveal information about user **tbrady**.

From the pulldown menu in Select table to query. There are only five choices related to UNIX. We could run a Query against all five, but we can probably eliminate UNIX System and perhaps UNIX Level, and run the other three.

The result of the query is shown in the slide on the lower left.

Notice that tbrady appears under UserName, so we know we have the right table.

For your new event source, none of the UNIX tables are likely to be used. Instead, you could use one of the general choices, such as Firewall, Network, Access Control, Configuration Management, etc.

The MessageID column will allow you to research this message in more detail, which will provide the foundation to creating the new XML file.

S  
E  
T  
O  
N

## Selecting a Database Table *General Guidelines*

- ▶ Duration or Byte Count  
> one of the Accounting tables
- ▶ Successful Connections or Connection attempts  
> one of the Authentication tables
- ▶ Unsuccessful Connection attempts, Denied or Rejected Connections  
> one of the Security tables
- ▶ Fan, Temperature, Failed Hard Drive, Power Cycles  
> one of the System tables



## Selecting a Database Table *General Guidelines, continued*

- ▶ Encryption Keys, Tunnel ID  
> one of the VPN tables
- ▶ IDS events  
> the Intrusion Detection System table
- ▶ IDPS events  
> one of the IPS tables



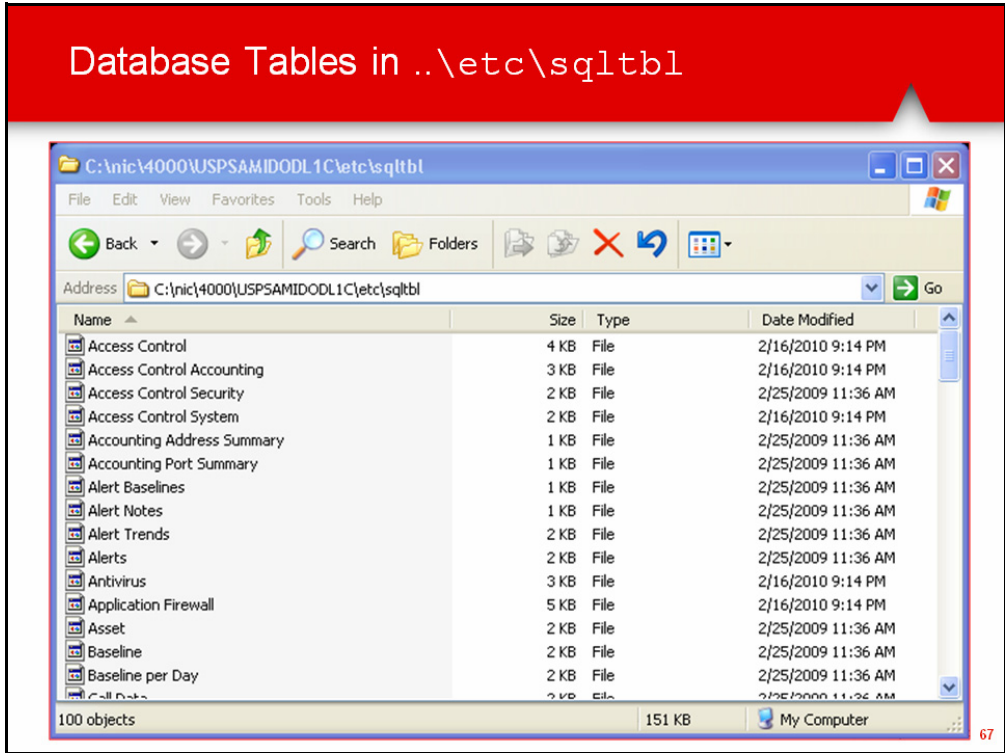
## Database Tables, continued

### Selecting a Database Table

The choice of which Table to use depends on what type of information you are looking for. For example, what events do you want to monitor, investigate and report on? Since we have a new event source, none of the vendor-specific tables would be under consideration, so the choice narrows to those general elements.

The slides shown on the left offer some suggestions on which tables to consider for specific needs.

SEFTON





## Database Tables, continued

### Database Tables in `\etc\sqli.tbl`

The slide on the left shows the `\etc\sqli.tbl` directory. You can highlight a file, and open it with a text editor to see what fields the table contains. Do not save the file in that directory.

S  
E  
E  
T  
O  
N

## Variables in a Particular Database Table Example: UNIX Security

```

Unix Security - WordPad
File Edit View Insert Format Help
unixsecurity, 50, TRUE, detail, device
paddr, DeviceAddress, TRUE, CHAR, 20, RESERVED, RESERVED, NULL
devicehostname, DeviceHostName, TRUE, CHAR, 128, RESERVED, RESERVED, NULL
stamp, Date/Time, TRUE, TIMESTAMP, 24, RESERVED, RESERVED, NULL
msg_id, MessageID, TRUE, CHAR, 64, RESERVED, RESERVED, NULL
faddr, ForeignAddress, TRUE, CHAR, 20, faddr, ADDR1, HOST1
fhost, ForeignHostName, TRUE, CHAR, 96, fhost, HOST1, NULL
fport, ForeignPort, TRUE, INT, 0, fport, PORT1, NAME1
fportname, ForeignPortName, TRUE, CHAR, 48, fportname, NAME1, NULL
username, UserName, TRUE, CHAR, 48, username, CHAR48_1, NULL
tbdstr1, UserName1, TRUE, CHAR, 48, username1, CHAR48_2, NULL
uid, Uid, TRUE, INT, 0, uid, INT1, NULL
type, Type, TRUE, INT, 0, ntype, TYPE, NULL
protocol, Protocol, TRUE, CHAR, 64, protocol, CHAR64_1, NULL
int_name, Interface, TRUE, CHAR, 48, interface, CHAR48_3, NULL
action, Action, TRUE, CHAR, 64, action, CHAR64_2, NULL
agent, Agent, TRUE, CHAR, 64, agent, CHAR64_3, NULL
reason, Reason, TRUE, CHAR, 64, reason, CHAR64_4, NULL
info, Info, TRUE, CHAR, 256, info, CHAR256_1, NULL
directory, WorkingDirectory, TRUE, CHAR, 64, directory, CHAR64_5, NULL
    
```



The Security Division of EMC 68

## Variables in a Particular Database Table UNIX Security formatted in Microsoft Excel

Displayed Database Table Name	Internal Table Name	Table ID	Table Type	Table Category	Internal Field Name	Displayed Field (Column) Name	FieldType	Field Length				
Unix Security	unixsecurity	50	TRUE	detail	device							
					paddr	DeviceAddress	TRUE	CHAR	20	RESERVED	RESERVED	NULL
					devicehostname	DeviceHostName	TRUE	CHAR	128	RESERVED	RESERVED	NULL
					stamp	Date/Time	TRUE	TIMESTAMP	24	RESERVED	RESERVED	NULL
					msg_id	MessageID	TRUE	CHAR	64	RESERVED	RESERVED	NULL
					faddr	ForeignAddress	TRUE	CHAR	20	faddr	ADDR1	HOST1
					fhost	ForeignHostName	TRUE	CHAR	96	fhost	HOST1	NULL
					fport	ForeignPort	TRUE	INT	0	fport	PORT1	NAME1
					fportname	ForeignPortName	TRUE	CHAR	48	fportname	NAME1	NULL
					username	UserName	TRUE	CHAR	48	username	CHAR48_1	NULL
					tbdstr1	UserName1	TRUE	CHAR	48	username1	CHAR48_2	NULL
					uid	Uid	TRUE	INT	0	uid	INT1	NULL
					type	Type	TRUE	INT	0	ntype	TYPE	NULL
					protocol	Protocol	TRUE	CHAR	64	protocol	CHAR64_1	NULL
					int_name	Interface	TRUE	CHAR	48	interface	CHAR48_3	NULL
					action	Action	TRUE	CHAR	64	action	CHAR64_2	NULL
					agent	Agent	TRUE	CHAR	64	agent	CHAR64_3	NULL
					reason	Reason	TRUE	CHAR	64	reason	CHAR64_4	NULL
					info	Info	TRUE	CHAR	256	info	CHAR256_1	NULL
					directory	WorkingDirectory	TRUE	CHAR	64	directory	CHAR64_5	NULL



The Security Division of EMC 69

---

## Database Tables, continued

### Variables in a Database Table

The slide on the upper left shows how the UNIX Security Database Table file looks like when opened in a text editor.

Each line in this file represents a variable that is encoded in the XML file. When creating a new XML file for a new event source, you will need to pick up these variables based on their appearance in the collect sample log message.

Scan down to roughly the middle of the file, to a line that begins with: **username**. The first string is the internal reference for the variable that is used to generate information under the column heading of UserName, which is listed as the next string.

The format shown in the slide at lower left was created by exporting the UNIX Security text file to Microsoft Excel. This format may be easier to work with, and also provides more information as to what might be expected in a given variable, such a length, type, etc. Refer to Appendix B for sample database tables, including the one on the slide.

To format the database table in Microsoft Excel:

1. Open the original file with MS Word.
2. Save it in a different directory (such as, desktop) as text only.
3. Rename the file by replacing the .txt suffix with .csv, (click Use .csv).
4. Open with MS Excel, re-format as needed.

S  
E  
E  
T  
O  
N

## The Remainder of the Log Message

- ▶ Known as: payload
- ▶ Depends on the variables in the Database Table chosen
- ▶ Variables
  - Identification of event data that is intended to be parsed
    - Will always be bound in < > characters in enVision
- ▶ Fixed text
  - Literal translation of event data that is not intended to be parsed
    - Character by character representation of the event
    - Explicit definition of all odd characters and whitespace is important!
- ▶ Example: MessageID 000296 in the UNIX Security Query



## Sample Payload *Solaris NIC Message ID 000296*

Use this window to display the processing information associated with message for a device or correlation class. **RSA enVision**

**Manage Messages for All Device Types**

Filter: WHERE Type IN 'UNIX Solaris' AND NIC message ID IN '000296'

Delete	Operator	Attribute	Comparison	Criteria
<input type="checkbox"/>	WHERE	Device / Type	IN	UNIX Solaris
<input type="checkbox"/>	AND	Event / NIC message ID	IN	000296

Group By: None

Apply Filter to monitored devices only

---

Filtered Messages: 1 message found

Select	Event Category	Message ID	Device	Message Text
<input type="checkbox"/>	User.Activity.Privileged Us e.Denied	<a href="#">000296</a>	UNIX Solaris	<@:*SYSVAL(\$MSGID,\$D1)><@action:su failure><@ntype:22 

71

# Payload

After the header is defined, the remainder of the message is the payload, which is synonymous with the message. The layout of the log message (payload) depends on the event source class and the database Table fields (variables).

The two main components of the payload are Variables and Fixed text.

- Variables are values that vary across similar types of events in the payload of the event.

Variables are surrounded by angle brackets (<>).

- Fixed text are literal translations of event data.

## Example

The slide on the left shows the result of the Message ID returned in the previous Query regarding user **tbrady**.

The Message Text column in the lower window displays the variables used and the order in which they appear in this message. The variables are enclosed in <> brackets, and fixed text appears literally. This is the fixed text that you observed in Exercise 2 when you browsed the log file.

This message text will apply to all log messages that have the same payload, and serves as the basis for the XML message definition. However, the brackets are not to be confused with the XML code itself, as they are special characters in XML and would be taboo characters in entries within a XML file.

Note that the use of the EventSource Integrator tool (almost) entirely eliminates the need to understand the syntax and idiosyncrasy of coding XML files.

SECTORS

### Message Details

*Solaris NIC message ID 000296*

System Configuration - RSA enVision - Windows Internet Explorer

Overview Alerts Analysis Reports

Dashboard

System Performance

Best Practices

System Configuration

Devices

Messages

Manage Messages

Manage Messages To Parse

Manage Categories

Manage Variables

Set Up Undefined Label

Directories

Users

Services

Manage Services

Manage Collector Service

Set Up DNS Resolver Service

Set Up DHCP Polling Service

Set Up Site Communication

Scheduler Service

Device Service

Asset Services

Universal Device Collection

Dashboard Items

Watchlists

Vulnerabilities

Assets

Use this window to add or modify a message in the message file for a device.

**Manage Messages - Add/Modify Message**

Vendor message ID: su\_auth

NIC message ID: 000296

Message text: <@\*SYSVAL(\$MSGD.\$ID1)><@action:su failure-<@ntype:22> <agent> [D <d> <facilityname> <severity>] 'su <username1>' failed for <username> on <interface>

Special processing:

SiteTrack on this event:

Parse this event:

Default parse setting:

Alert level: 2

Alert category: User Activity

NIC category: User

Event category: User\_Activity.Privileged Use.Denied

Data table: Unix Security

Vendor description: No description given

Vendor action: No action given

Company description: Keep an eye on this event

Company action: Refer any instances to Dave Amidon

Apply Cancel

72

## Payload, continued

### Message Details

Drilling down into the details of message 000296 by clicking on the Message ID hyperlink, reveals other items that have been decoded by the XML file.

A few examples: Vendor Message ID, Message text (again), Severity level, NIC category, Alert category, Event category and Data table used (but we knew that already, since that's where we found the Message ID in the first place). However, this is where you could find which table was used if you did not know.

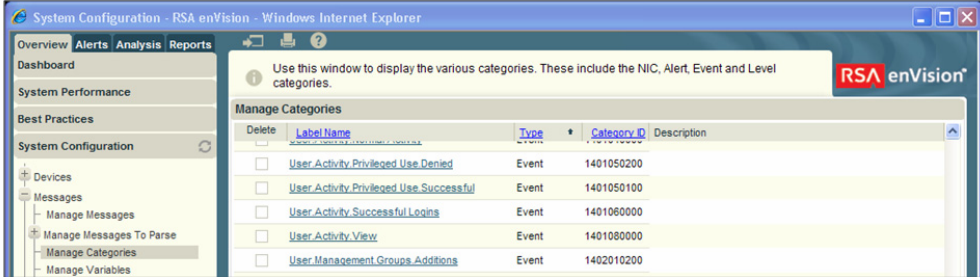
There are also Vendor and Company description and action entries, although in this example, they are empty. We have entered a Company description of "Keep an eye on this event" and a Company action of "Refer any instances to Dave Amidon." Therefore, if this message is used in a View and an Alert is triggered, the analyst has more information to help handle this incident.

S  
E  
C  
U  
R  
I  
T  
Y  
E  
N  
V  
I  
S  
I  
O  
N

## Event Categories - enVision Taxonomy

Descriptors of Messages

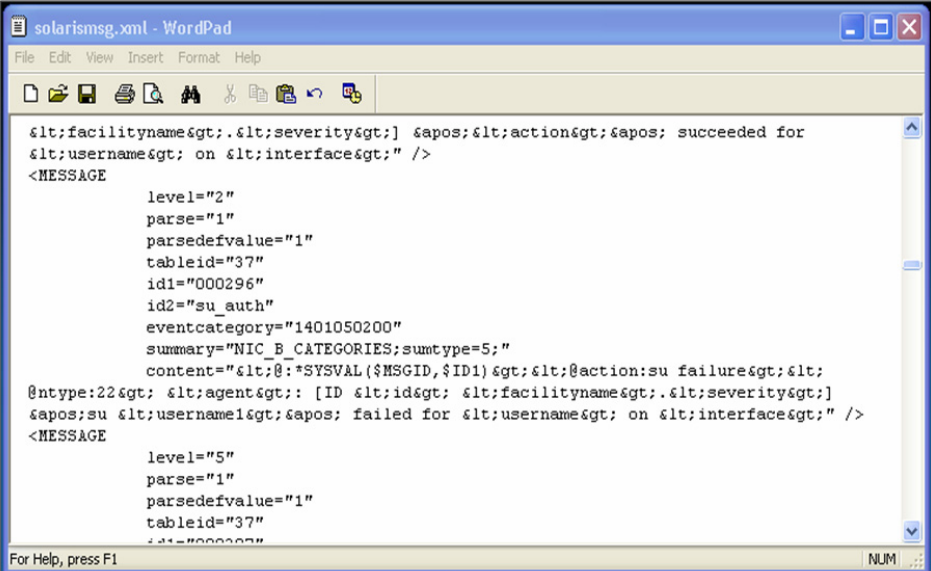
- ▶ Up to five Event Category levels
- ▶ Displayed as a dotted alphabetic string - Label Name
- ▶ Coded as a string of 10 decimal digits - Category ID



Delete	Label Name	Type	Category ID	Description
<input type="checkbox"/>	User Activity.Privileged Use.Denied	Event	1401050200	
<input type="checkbox"/>	User Activity.Privileged Use.Successful	Event	1401050100	
<input type="checkbox"/>	User Activity.Successful Logins	Event	1401060000	
<input type="checkbox"/>	User Activity.View	Event	1401080000	
<input type="checkbox"/>	User Management.Groups.Additions	Event	1402010200	

**RSA**  
The Security Division of EMC 73

## Solaris XML File Detail



```

<lt;facilityname>.<lt;severity>] &apos;<lt;action>&apos; succeeded for
<lt;username> on <lt;interface>" />
<MESSAGE
  level="2"
  parse="1"
  parsedefvalue="1"
  tableid="37"
  id1="000296"
  id2="su_auth"
  eventcategory="1401050200"
  summary="NIC_B_CATEGORIES;sumtype=5;"
  content="<lt;@: *SYSVAL ($MSGID, $ID1) &gt;&lt; @action:su failure&gt;&lt;
@ntype:22&gt; &lt; @agent&gt;: [ID &lt;id&gt; &lt;facilityname&gt;.<lt;severity&gt;]
&apos;su &lt;username1&gt;&apos; failed for <lt;username> on <lt;interface>" />
<MESSAGE
  level="5"
  parse="1"
  parsedefvalue="1"
  tableid="37"
  id1="000296"
  
```

The Security Division of EMC 74



# Event Categories

## enVision Taxonomy

The slide on the upper left reviews the elements of the enVision Taxonomy for classifying messages. Although we cannot drill down in the event category listed - User.Activity.Privileged Use.Denied - we can research enVision's Taxonomy at **System Configuration > Messages > Manage Categories**, under Label Name. This tool also provides the Category ID in a 10-digit decimal format.

When creating the XML file for your new event source, each log message will be assigned to at least a NIC and Alert category. If a log message can be categorized more precisely, then a deeper Event category may be assigned. Refer to Appendix A for a complete listing of the taxonomy, arranged alphabetically.

## Detail Example

The slide on the lower left shows the actual XML file that pulls all that information into a few lines of code. This is the Message tag in XML terminology, as evidenced by the < opening bracket and label of **MESSAGE**. It can be translated as follows:

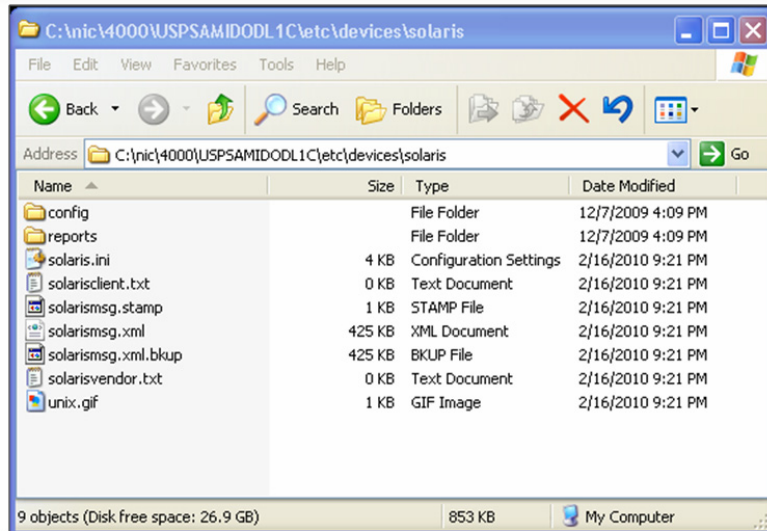
Message Element	Meaning
level=2	severity level 2 Alert
parse=1	1 indicates that this message will be parsed
parsedefvalue=1	the default state is 1 (must be 1 to parse)
tableid=37	the internal reference for the UNIX Security table is 37
id1=000296	enVision Message ID
id2=su_auth	Vendor (i.e., Sun Solaris) Message ID
eventcategory=	enVision Taxonomy (i.e., User.Activity.Privileged Use.Denied)
summary=	summary of category/type
content=	the enVision Message Text, with taboo characters substituted out. (This is where EventSource Integrator saves the frustration of working with XML.)

And the MESSAGE tag is closed with the /> characters.

Although EventSource Integrator does not require that you know the details of XML, you should still be familiar with enVision's variables.

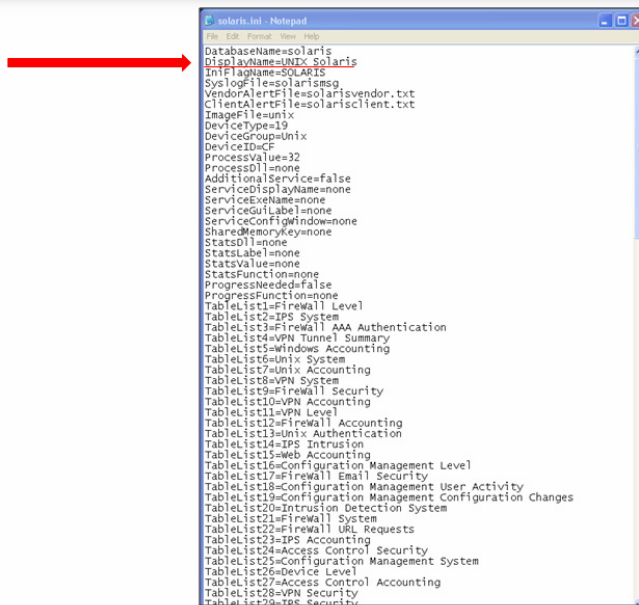
S  
E  
E  
T  
O  
N

## enVision Device Files Solaris Directory



The Security Division of EMC 75

## solaris.ini File



The Security Division of EMC 76

---

## enVision Event Source Files

In the Solaris example shown in the slide on the upper left, the entire support folder contains at least three files in addition to the \*.XML file: two \*.txt files and a \*.ini file.

Once you have completed the creation of the XML file for your new event source by using EventSource Integrator (ESI), you will use the ESI tool to create this folder and the three files for activation on your enVision site. Although we are getting ahead of ourselves, that process is referred to as deployment, and will be covered in the next unit.

### The .ini File

The solaris.ini file, shown in the slide on the lower left, contains information about the Solaris event source.

Typically, it is not to be edited. However, the DisplayName field should be edited to identify your new event source. When you first create an XML file in EventSource Integrator, you provide it a filename. That file will eventually appear in this file as DatabaseName. Notice here that the DisplayName has been edited so that it appears as “UNIX Solaris” throughout enVision’s interface.

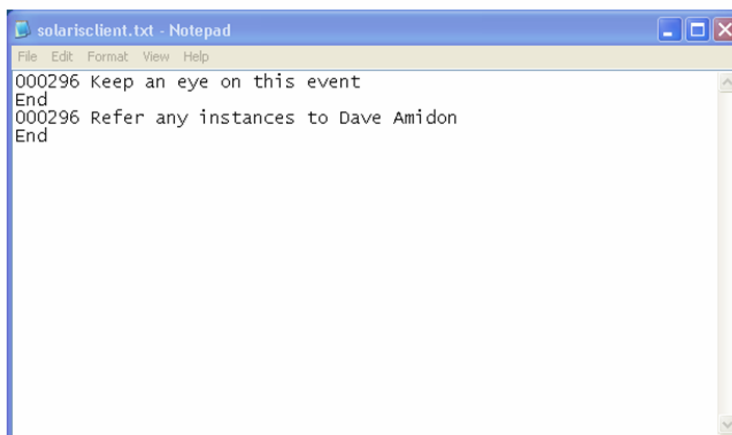
Notice, too, that there are many Database Tables to which various Solaris messages might be parsed.

Do not make any changes to this file beyond the DisplayName.

# STEP 10

## Vendor and Company Description and Action Notes

- ▶ Vendor text file is empty - solarisvendor.txt
- ▶ Company ("client") file - solarisclient.txt



```
solarisclient.txt - Notepad
File Edit Format View Help
000296 Keep an eye on this event
End
000296 Refer any instances to Dave Amidon
End
```

## enVision Device Files, continued

### The .txt Files

The slide on the left shows the contents (or lack thereof) of the two .txt files. Remember that we added information to the Company description and action fields. These are stored in the client txt file, as shown on the slide.

# SECTION

## Exercise 3 Determine the Table and Field Values

### The goal of this exercise is:

- To identify which fields to populate with message content (identified in the scenario)
- To identify the table that you want to parse the message to

### Scenario

Before you create the XML file to parse the message, you should determine:

- What information you want to get from the message.
- Where the information resides (in what table).
- Which fields you want to use from the table.

In the `syslogDogHouse.unx` log file, there are four types of messages:

```
TCP Connection Established: Connect
TCP Connection Request: Connect
TCP Connection Terminated: Abnormal
TCP Connection Terminated: Normal
```

The complete message includes this information:

```
Feb 11 04:20:16 [10.10.4.1] Socks5[747]: TCP
Connection Terminated: Normal (172.30.21.43:37469
to 172.30.32.96:443) for user root: 453 bytes
out, 2779 bytes in
```

First, determine the type of information you want from the message. In this scenario, we want to know the status of TCP connections, including local and foreign ip addresses, local and foreign port numbers and usernames. If the session was terminated, we want to know the number of bytes transferred.

Therefore, we need to find a table that includes the information identified above.

### Exercise 3, continued

The BullDog device is a Firewall, so we want to use a Firewall table to analyze the information from the message. We must determine which Firewall table to use.

Follow these steps to determine the appropriate Firewall table and fields.

Step	Action
1	In the enVision user interface, click the help icon.
2	Click <b>Search</b> in the left panel and enter <b>when to use each database table</b> and click the <b>Search</b> button.
3	Select <b>When to Use Each Database Table</b> in the left panel. <b>Result:</b> The When to Use Each Database Table help screen appears. Note that the window has three columns: <ul style="list-style-type: none"> <li>To report on device subclass...</li> <li>And report on this information...</li> <li>Select this table...</li> </ul>

To report on device subclass...	And report on this information...	Select this table...
all	<a href="#">Alert notes</a>	<a href="#">Alert Notes table</a>
	Messages generated from the <a href="#">Alerter service</a>	<a href="#">Alerts table</a>
	Number of messages received by <a href="#">alert category</a>	<a href="#">Baseline table</a>

4	Scroll down to Firewall in the first column. <b>Result:</b> There are several Firewall tables.
---	---

Firewall	Events related to logging in or logging out of a device or server	<a href="#">Firewall AAA Authentication table</a>
	Connection messages from which address, bandwidth, duration and port-specific information can be derived (for example, teardown messages)	<a href="#">Firewall Accounting table</a> <b>Note:</b> Reports take longer to run using this table because it includes all the information as opposed to the various Firewall Accounting Summary tables which already have information broken down.
	Overall status of connection messages related to address information	<a href="#">Firewall Accounting Address Summary table</a>
	Overall status of connection messages related to bytes	<a href="#">Firewall Accounting Byte Summary table</a>



## Exercise 3, continued

Step	Action
5	<p>To determine the appropriate table for your message, review the second column, <b>And report on this information...</b>, to see the type of information the table contains.</p> <p>Since we want to know the status of TCP connections, we should look for a table that contains connection information. The first table that includes connection information is the Firewall Accounting table.</p> <p>Click <b>Firewall Accounting table</b> in column 3.</p> <p><b>Result:</b> A list of fields for the table appears.</p>
6	<p>Review the list to see if all of the information that we want to parse is included in the table:</p> <ul style="list-style-type: none"> <li>• Local address (LocalAddress)</li> <li>• Local port (LocalPort)</li> <li>• Foreign address (ForeignAddress)</li> <li>• Foreign port (ForeignPort)</li> <li>• Username (UserName)</li> <li>• Sent bytes (SentBytes)</li> <li>• Received bytes (ReceivedBytes)</li> </ul> <p><b>Result:</b> All of the fields are included.</p> <p><b>Note:</b> Appendix B also includes the fields for the Firewall Accounting table.</p>
7	<p>Click the <b>back</b> button in the help menu.</p>
8	<p>(Optional) Click additional Firewall tables and view the fields.</p> <p><b>Note:</b> The Firewall Accounting table is the only Firewall table that includes all the fields that we want to parse. There are tables other than Firewall tables that might include all the fields, such as the Access Control table. We could parse the message to the Access Control table, but since this device is a Firewall, we want to view the Firewall information using a Firewall report.</p>

# SECTION

---

# Creating Support Files

---

## Objectives

Upon completion of this unit, you should be able to:

- Describe the EventSource Integrator
- Identify how headers and payloads are defined in ESI
- Create support files for an unknown event source
- Create and deploy the event source package
- Test the event source integration

S  
E  
C  
U  
R  
I  
T  
Y  
S  
O  
L  
U  
T  
I  
O  
N

## RSA enVision EventSource Integrator (ESI)

- ▶ ESI is an intuitive graphical interface that allows you to:
  - Integrate the event source easily
  - Create, update and validate the event source XML
- ▶ Enforces best practices for generating the event source XML



## ESI Features

- ▶ Create a new event source XML
- ▶ Modify an existing event source XML
  - Quick edit
  - Complete edit
- ▶ Parse status of the events in log file
- ▶ Validation - data pattern errors
- ▶ Report generation
- ▶ Event source package



# RSA enVision EventSource Integrator

RSA enVision EventSource Integrator enables you to integrate event sources with the RSA enVision platform.

Using EventSource Integrator, you can define how enVision should interpret the events from an event source. The definitions for the event source are stored as an XML file, called an event source XML file, which you deploy on the enVision platform. After you deploy the event source XML file, enVision will be able to interpret the events and monitor the event source.

EventSource Integrator (ESI) allows you to create a new event source XML file for an event source that is not supported by enVision and edit an existing event source XML file to add or modify the definitions in the event source XML file.

You can verify the header and message definitions in the event source XML by parsing the log file. After you define the header and message, you can verify if the events in a log file are parsed for the definitions in the event source XML file.

The status of each event is displayed in the Log View area as one of the following:

- Parsed
- Header parsed and message not parsed
- Not parsed

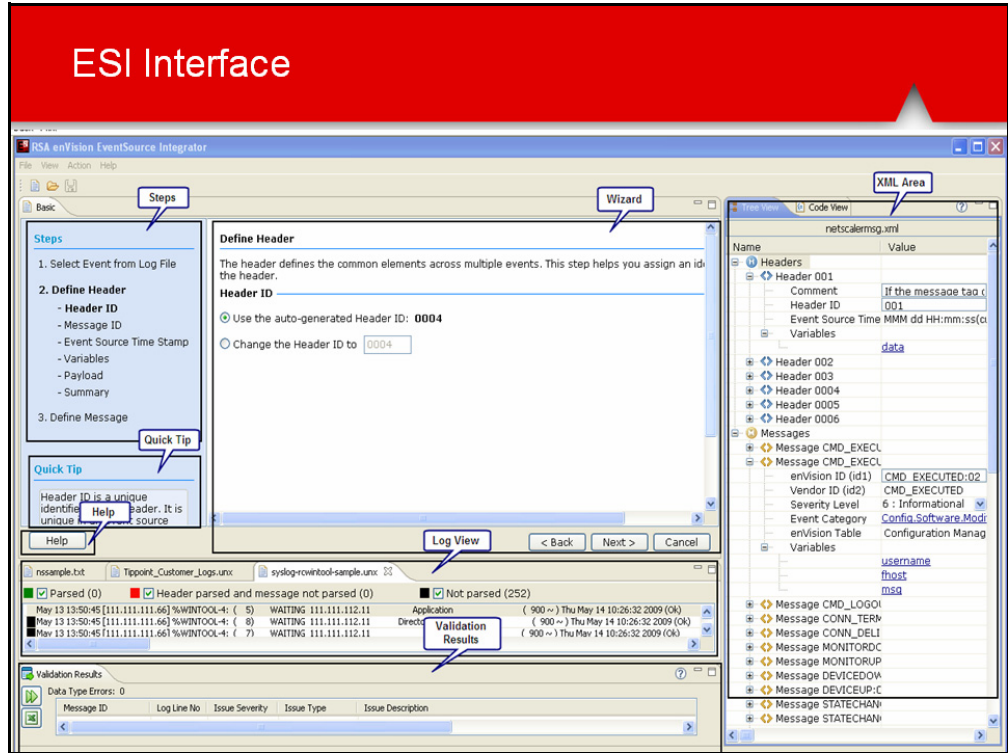
This status also provides a count of the events parsed in the selected log file based on the definitions in the event source XML file.

You can generate reports to analyze how the events in a log file are parsed by the header and message definitions in the event source XML file. Each selected event is parsed against each header definition and against each message definition that has the same vendor ID (id2) as the first completely parsed header definition in the event source XML file. The report provides a complete breakdown of the header and message definitions in the event source XML file. It shows the parsed values for the associated variables and static text for the events in the log file. The report also provides the parsed status for each header and message definition. You can also use a report to debug the definitions as the report provides details of how the events are parsed. You can generate reports on:

- Selected events in the log file
- All events in the log file

The event source package allows you to deploy the event source in enVision. When you deploy the package, you run a script that assigns a unique event source type ID to the event source. RSA enVision identifies the event source with the event source type ID, which is used for analysis and reporting.

S  
E  
T  
O  
N



---

## RSA enVision EventSource Integrator, continued

### Event Source Integrator Interface

The areas of the EventSource Integrator (ESI) user interface are as follows:

- **Wizard:** Displays the wizard pages that assist you in creating the message.
- **Steps:** Highlights the step that you are performing in the wizard.
- **Quick Tip:** Displays information about the page that you are performing in the wizard.
- **Help button:** Displays context-sensitive Help for the current page.
- **XML Area:** Displays the event source XML file. It has two tabs:
  - **Tree View tab:** Displays the event source XML file in tree format and allows you to perform a quick edit.
  - **Code View tab:** Displays the event source XML file in code format and allows you to perform a complete edit.
- **Log View:** Displays the log file that is opened and the parse status of the events in the log file.
- **Validation Results:** Displays the pattern validation errors found when validating an event source XML file.

## Defining the Header in ESI

Header includes:

- ▶ Message ID – Unique identifier for the message
- ▶ Event Source Time Stamp (optional)
- ▶ Variables (optional)
- ▶ Examples:

**The same header may be used for both messages, but different Message IDs**

```
Feb 11 04:20:16 [10.10.1.1] Socks5[3505]: TCP Connection Request:  
Connect(172.30.21.43:37425 to 172.30.32.93:80) for user root
```

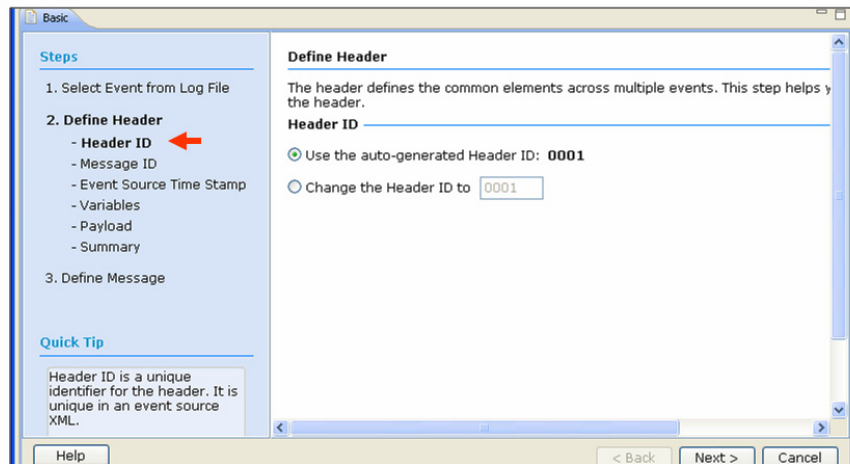
```
Feb 11 04:20:16 [10.10.1.1] Socks5[4921]: TCP Connection Established:  
Connect(172.30.21.43:37425 to 172.30.32.93:80) for user root
```

**Be aware of anything that varies within the header!**



The Security Division of EMC 84

## Defining the Header ID



The Security Division of EMC 85



---

# Defining the Header in ESI

## Overview

As stated earlier, the header portion of the message contains elements that are common across multiple events. The header includes the Message ID, and, optionally, the event source Time Stamp and variables. The Message ID is the unique identifier for the message. Typically, the vendor provides a unique identifier for each event generated by the event source. You can select this value as the Message ID. To determine the identifier for the event, see the vendor documentation.

If there is no text in the event that you can identify as a unique identifier or if you are unable to identify a unique identifier from the vendor documentation, you can create a Message ID from a set of strings in the event. To identify such strings, you must analyze the complete set of events in the log file and select only strings that are unique to that event.

EventSource Integrator allows you to define values in an event up to the end of the Message ID or Time Stamp, whichever appears later in the event, as the event header.

If you want the header to include more values from the event, you can extend the header definition. This functionality allows you to choose header variables from any part of the event including the payload. When you define header variables in the payload of the event, EventSource Integrator treats unselected text that precedes the last defined header variable as static text in the event header.

## Defining the Header ID

A header ID is a unique identifier in the event source XML file. RSA enVision EventSource Integrator generates a unique header ID for the selected event. You can either retain the header ID generated by EventSource Integrator or specify a different header ID for the event.

### To retain the generated header ID:

1. On the Define Header - Header ID page, select **Use the auto-generated Header ID**.
2. Click **Next**.

### To specify a header ID:

1. On the Define Header - Header ID page, select **Change the header ID to**.
2. In the Header ID field, enter a unique alphanumeric value from one to four characters. For example, 0001 or ab09.

---

Note: RSA recommends that you use sequential values based on any existing header IDs.

---

S  
E  
T  
O  
N

## Defining the Message ID in the Header



The Security Division of EMC 86

---

## Defining the Header in ESI, continued

### Defining the Message ID in the Header

You must define a unique Message ID for the event in the event source XML file. You either select a vendor-provided identifier as the Message ID or create a Message ID from unique strings in the event.

#### To specify the Message ID:

1. On the Define Header - Message ID page, select the identifier in the event.
2. Click **Next**.

#### To create the Message ID:

1. On the Define Header - Message ID page, under Are you able to identify the Message ID, select **No, I would like to create the Message ID**.
2. Select a unique string in the event.
3. Click **Add**.

Repeat steps 2 and 3 until you have selected all the necessary strings in the event that form a Message ID. You must select at least two strings to create a Message ID. The selected strings appear in the Message ID field in the order that you selected them.

If you want to change the strings that you have selected, do the following, use the Up and Down buttons.

To delete a string, select the string, and click **Delete** and click **Next**.

S  
W  
E  
E  
T  
O  
N

### Defining the Time Stamp in the Header

**Steps**

1. Select Event from Log File
- 2. Define Header**
  - Header ID
  - Message ID
  - **Event Source Time Stamp**
  - Variables
  - Payload
  - Summary
3. Define Message

**Define Header**

RSA enVision associates the time of collection as the default time stamp. This step enVision time stamp or override it with a time stamp in the event.

**Event Source Time Stamp**

If you want to use the enVision time stamp, click Next. To override the enVision time stamp and select the check box.

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

Override enVision time stamp

**Quick Tip**

Event source time stamp indicates the time of event generation. Some events may not have a time stamp.

Help < Back Next > Cancel

**RSA**  
The Security Division of EMC 87

## Defining the Header in ESI, continued

### Defining the Time Stamp in the Header

A Time Stamp contains information about the date and time at which the event message was generated by the event source. Some events may not contain a Time Stamp. You can define two types of Time Stamps in RSA enVision EventSource Integrator:

- **enVision Time Stamp:** The date and time that RSA enVision associates with the event at the time of event collection. You can define an enVision Time Stamp in either of the following situations:
  - The event does not contain a Time Stamp.
  - You do not want to use the Time Stamp contained in the event.
- **Event source Time Stamp:** If the event contains an event source Time Stamp, you can define a Time Stamp. EventSource Integrator identifies the format of the Time Stamp and assigns it to the variables.

To define an event source Time Stamp:

1. On the Define Header - Define Event Source Time Stamp page, select the Time Stamp in the event.
2. Select **Override enVision Time Stamp**.
3. Click **View Format**. Depending on the selected Time Stamp, EventSource Integrator displays the date and time format information.
4. If the Time Stamp format displayed by EventSource Integrator is not correct and you want to change the format, do the following:
  - Click **If the format is not correct, click here**.
  - In the 'Select the Format' pop-up window, the format codes of the selected Time Stamp are populated in the right-hand list box. You can rearrange the format codes or, remove the existing format codes from the right-hand list box and add new format codes from the left-hand list box.

S  
E  
E  
T  
O  
N

## Defining Variables in the Header

**Steps**

1. Select Event from Log File
- 2. Define Header**
  - Header ID
  - Message ID
  - Event Source Time Stamp
  - **Variables** ←
  - Payload
  - Summary
3. Define Message

**Quick Tip**

Header variables represent parts of the header that vary across events.

**Define Header**

This step helps you define the variables in the default header. Unselected parts are optional. You can also extend the header definition in which case the unselected are classified as static text.

**Variables**

1. Select the value in the event
2. Select or type a variable name and click Assign
3. Optionally, assign a null character before the value by selecting the Assign null v
4. Repeat the above steps till all the variables are defined

%CERT-1-101001

Variable name: level

Assign null value to the variable

Assign

Help < Back Next > Cancel

**RSA**  
The Security Division of EMC 88

## Defining the Payload in the Header

**Steps**

1. Select Event from Log File
- 2. Define Header**
  - Header ID
  - Message ID
  - Event Source Time Stamp
  - Variables
  - **Payload** ←
  - Summary
3. Define Message

**Quick Tip**

Payload is the key part of the event that contains details useful for analysis and reporting in RSA enVision.

**Define Header**

This step helps you define the payload.

**Payload**

Identify the payload in the event.

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

Define the payload as:

- Remaining part of the event
- Starting from the variable: --Select--
- Entire event

Highlight payload or select "Remaining part of the event"

Help < Back Next > Cancel

**RSA**  
The Security Division of EMC 89

## Defining the Header in ESI, continued

### Defining Variables in the Header

Header variables are elements that are subject to change in an event. The header variables are not used for analysis and reporting in RSA enVision. ESI defines the undefined part of the header in an event as static text.

#### Assigning Null Values to Header Variables

EventSource Integrator allows you to assign null values to header variables. When you assign a null value to a header variable, the variable acts as a placeholder to any value that will occur at that location in the event to which the variable is assigned. When you parse an event with a header variable that has a null value assigned for a particular field in an event, the parsed value of the header variable is the actual value of the event in that location.

You can assign a null value to a variable if you want to use the same header definition to parse two events that are similar to each other but have a variation in a value that appears in the same location in both events.

### Defining the Payload in the Header

The payload is the part of the event that contains the data used for analysis and reporting in RSA enVision.


To define the payload:

1. On the Define Header - Payload page, do one of the following:
  - To define the part of the event after the header definition, select **Remaining part of the event**.
  - To include a header variable as the starting point of the payload, select **Starting from the variable**, and, from the drop-down list, select the variable. The rest of the event, including the selected variable, is selected as the payload.
2. To define the whole event as the payload, select **Entire event**.

If you selected **Extend header definition** on the Define Header - Variables page, select **Custom Selection** to manually define the starting point of the payload. Use the slider control to select the starting point of the payload. You can move the slider to the right of the last defined variable. The unselected text in the event is treated as static text and is part of the event header.

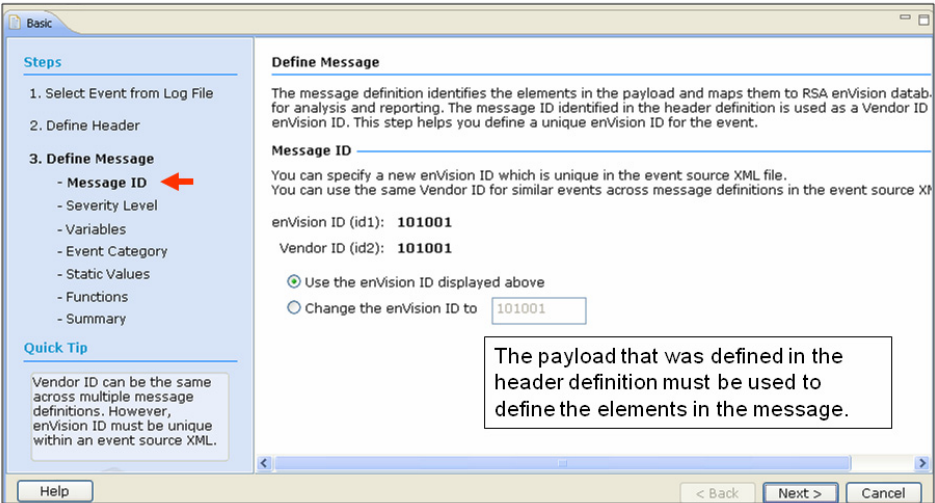
## Payload


- ▶ Same as the message in the event
- ▶ Includes:
  - Message variables
  - Static text
  - Message elements, including:
    - Vendor ID
    - enVision ID
    - Severity Level
    - Event Category
    - Static Values
    - Functions



The Security Division of EMC 90

## Defining the Message ID





The Security Division of EMC 91



## Defining the Message in ESI

### Payload

As stated earlier, the Payload is synonymous with the message portion of the event.

In the Payload, you define:

- Message variables
- Static text
- Message elements, such as:
  - The Vendor and enVision IDs
  - Severity Level
  - Event Category
  - Static Values
  - Functions

### Defining the Message ID

RSA enVision ESI generates two identifiers for each event:

- **Vendor ID.** This is an identifier of the message in the event. The vendor ID is always the Message ID selected in the header definition. Vendor ID may be the same across similar or different types of events. You cannot change the vendor ID.

The Vendor ID indicates the event identifier provided by the event source vendor and is automatically generated from the Message ID defined in the header definition.

- **enVision ID.** The unique identifier by which enVision identifies the event exclusively. The enVision ID can be defined in one of the following ways:
  - Same as vendor ID
  - A combination of the Message ID defined in the header definition and a unique variant
  - A brief description of the event to identify the event

The enVision ID is the identifier of the message in the event source XML file. In some events the vendor ID is not unique, in which case a unique enVision ID may be provided for clarity on the event. ESI generates a default enVision ID, based on the text selected for the Message ID in the header definition. The enVision ID is a combination of the Message ID defined in the header definition and a unique variant. For example, if the Message ID is 109801, the enVision ID can be defined as 109801:02.

The enVision ID can also be a brief description of the event to identify the event. For example, in the following event, the Message ID is 187698, the enVisionID can be defined as CableFailover.

```
Jan 01 11:06:39 [10.5.92.51] %PIX-1-101001:
(PRIORITY) Error reading failover cable status.
```

## Defining the Severity Level

**Define Message**

This step helps you assign the severity level for the event.

**Severity Level**

Severity level: 1 : Alert

The severity defines the severity of the event determines how enVision responds when the event occurs.

Quick Tip: Severity level defines the severity of the event and determines how RSA enVision responds when the event occurs.

Buttons: Help, < Back, Next >, Cancel

**RSA**  
The Security Division of EMC 92

## Defining the Message Variables

**Define Message**

This step helps you assign parts of the payload to variables in an RSA enVision table. Unselected payload are classified as static text.

**Variables**

1. Select the value in the event and assign it to a variable.  
2. Optionally, assign a null character before the value by selecting the Assign null value check box.  
3. Repeat the above steps till all the variables are defined.

(PRIORITY) Failover cable OK.

Assign the value to:  Variable in enVision

Assign null value to the variable

Enter the variable name:  Check Assign, else click [Search and Assign](#)

Variable name: priority

Assign

priority in Intrusion Detection Sy...  
priority in FireWall System  
priority in VPN System

1. Highlight value  
2. Select name  
3. Click Assign

Buttons: Help, < Back, Next >, Cancel

**RSA**  
The Security Division of EMC 93

## Defining the Message in ESI, continued

### Defining the Severity Level

The severity level defines the severity of the event and determines how RSA enVision responds when the event occurs. The severity level is required for analysis and reporting in enVision.

You can assign one of the following severity levels to the event.

0	Emergency	Emergency conditions in the system
1	Alert	Alert conditions in the system
2	Critical	Critical conditions in the system
3	Error	Errors in the system that need to be fixed
4	Warning	Conditions that should be investigated
5	Normal	Normal but significant conditions
6	Informational	Informational messages
7	Debug	Debugging messages

### Defining the Message Variables

A Message variable is a value in the payload that varies across similar types of events.

Define a message variable by selecting a part of the event and assigning it to one of the following types of variables:

- **Variables in an RSA enVision table.** You can assign a value in an event to a variable from an enVision table. When you define multiple message variables, ensure that you select variables from the same enVision table or from the enVision Global table.
- **Variables in the enVision Global table.** You can assign a value in the event to a variable from the enVision Global table. Typically, you do this if you cannot find a suitable variable in the enVision table from which you have selected variables. The Global table consists of commonly used variables from all the enVision tables. The report generated from the selected enVision table will not contain the global table variable. This global table variable is available only in the report generated using the global table.
- **Temporary variables.** To assign a value in the event to a dummy variable that is used only for parsing and not for reporting, assign the value to a temporary variable.

## Defining the Message Event Category

The screenshot shows the 'Define Message' wizard in the RSA enVision interface. The left sidebar lists the steps: 1. Select Event from Log File, 2. Define Header, and 3. Define Message. Under 'Define Message', 'Event Category' is selected and highlighted with a red arrow. The main area shows the 'Event Category' field with the text 'System.Normal Conditions' and a 'Search' button. A red arrow points from a text box to the 'Search' button. The text box contains the instruction: 'Use the enVision Taxonomy to determine the Event Category'. At the bottom right, the RSA logo and 'The Security Division of EMC 94' are visible.

## Defining the Message Static Values

The screenshot shows the 'Define Message' wizard in the RSA enVision interface. The left sidebar lists the steps: 1. Select Event from Log File, 2. Define Header, and 3. Define Message. Under 'Define Message', 'Static Values' is selected and highlighted with a red arrow. The main area shows the 'Static Values' section with instructions: '1. Enter the name of the variable, or click Search. 2. Enter the value and click Assign. 3. Repeat the steps to assign all the static values.' Below the instructions are input fields for 'Variable name:' and 'Value:', and an 'Assign' button. A red arrow points from a text box to the 'Assign' button. The text box contains the instruction: 'Static values are user-defined values assigned to variables from an RSA enVision table. This is done for improved analysis and reporting.' Below the input fields is a table with columns 'Variable', 'Value', and 'Actions'. At the bottom right, the RSA logo and 'The Security Division of EMC 95' are visible.

---

## Defining the Message in ESI, continued

### Defining the Message Event Category

Message categories are hierarchical. They can consist of up to five levels of categorization:

- NIC category
- Alert category
- Up to three levels of Event category

The Event category:

- Indicates the category to which the event belongs, based on the enVision taxonomy.
- Is required for reporting in enVision.

In the Event Category, a period separates the different levels of categorization. For example, in the category **User.Activity.Failed Logins**, User is the NIC category, Activity is the alert category, and Failed Login is the event category.

An event category must be specified to classify the event for analysis and reporting in RSA enVision.

### Defining the Message Static Values

Static values are user-defined values that are assigned to variables from an RSA enVision table.

Static values for variables are specified in an enVision table for better analysis and reporting in enVision. For example, to enhance reports with more descriptive information, you can add a static value to capture the reason for an event. Select reason as the variable and enter a detailed description of the event.

Variables can only be selected from the enVision table from which you selected message variables and only variables not previously selected.



## Defining the Message in ESI, continued

### Defining the Message Static Values, continued

#### To define static values:

On the Define Message - Static Value page, do one of the following:

- Enter the variable name as follows:
  - In the Variable name field, enter the name of the variable.
  - From the drop-down list of matching variable names, select the appropriate variable.
- OR
- Search for a variable, as follows:
  - Click **Search**.
  - In the Search Variable pop-up window, in the Search text field, enter the name or description of the variable.
  - In the Search within field, select whether you want to search for the text in the variable name or description or both.
  - Click **Search**.
- Select the variable from the list, and click **Select**.
 

**Note:** Variables that were selected while defining message variables are not listed.
- In the Value field, enter the static value for the variable.
- Click **Assign**.
- Repeat steps 1 through 3 until you define all the static values.
- Click **Next**.

### Defining Functions

A function defines an action to be performed on a variable in an event to generate a user-defined value. Multiple functions may be defined for a single event but a variable can only be used in one function. Functions are described in Unit 5.

### Static Text

Static text is anything that you want to appear as it appears in the event. In the example on the slide on the lower left, **(Primary) Link status** and will appear in the message as it appears here in the event, as will **on interface INTNAME**.

You do not have to explicitly define static text in ESI. If you do not select the text while creating the XML file in the ESI tool, it will appear as static text.

# SECTION



## Exercise 4.1 Create the Event Source File

### The goal of this exercise is:

- To create the XML and associated files for the new event source

This exercise consists of four parts:

- Select an event from the log file
- Define the Header
- Define the Message
- Validate the XML file

### Scenario

In this exercise, you will create an XML file using a common message that is representative of the logged messages from a Firewall proxy server such as the DogHouse. There are many ways to parse the message, but in this scenario we want to use the status (Established, Request or Terminated) as the Message ID since it is what we want to report on.

We will begin by selecting the Established message and we will parse the message as follows:

Component	Value
Message ID	Established
Variables	Local IP address Local port Foreign IP address Foreign port username
Static text	: TCP Connection

## Exercise 4.1, continued

### Select an Event

In this activity, you will install the EventSource Integrator, name the XML file, select an event source class (formerly called a category), import the log data and select an event.

Using the naming conventions of supported devices within enVision (for example, Cisco PIX, Check Point FW-1, UNIX Solaris, you will create an event with the name Bulldog DogHouse. The event source class of Firewall will be used because it matches most closely the type of data being received from the unknown syslog.

Follow these steps to name the XML file, select an event source class, import the log data, and select an event.

Step	Action
1	On the desktop, doubleclick the <b>RSA_enVision_ESI_1.1.0_setup.exe</b> to install the EventSource Integrator.
2	Click <b>Next</b> twice.
3	Select <b>I accept the terms in the license agreement</b> and click <b>Next</b> .
4	Click <b>Next</b> .
5	Click <b>Install</b> .
6	Leave <b>Launch RSA enVision EventSource Integrator</b> checked and click <b>Finish</b> .  <b>Result:</b> RSA enVision EventSource Integrator opens.
7	Click <b>Close</b> to close the popup window.
8	In the right panel (Tree View, Code View) click the <b>minimize</b> button to close the code view for now.
9	Ensure that the <b>Create XML</b> radio button is selected. In the Enter the XML filename field:, type: <b>BulldogDogHousemsg</b>
10	In the Select a folder to save the XML: field, click <b>Browse</b> and navigate to the desktop ( <b>c:\Documents and Settings\Administrator\Desktop</b> ) directory.
11	In the Select an event source class: pulldown menu under Security, select <b>Firewall</b> .
12	Click <b>Start</b> .

## Exercise 4.1, continued

### Select an Event, continued

Step	Action
13	In the Select a log file: field, click <b>Browse</b> and navigate to the <b>c:\nic\4000\<i>&lt;nodename&gt;</i>\bin\syslogDogHouse.unx</b>  <b>Result:</b> A message appears asking if this is the log file you exported using lsdata.
14	Click <b>Yes</b> .
15	Click <b>Select an Event</b> .  <b>Result:</b> A window opens displaying the events in the syslog file.
16	Select one of the <b>Established</b> events.
17	Click <b>Select</b> .  <b>Result:</b> The event appears in the Selected Event window.
18	Click <b>Next</b> .

## Exercise 4.1, continued

### Define the Header

The first thing you need to do is define a HEADER tag. The HEADER contains those elements that are common to large groups of messages. A Header consists of two main elements:

1. Header, consisting of Message ID, Time Stamp, and variables
2. Payload, containing detailed information about the event (synonymous with message)

Follow these steps to define the header.

Step	Action
1	Keep the default <b>Use the auto-generated Header ID: 0001</b> selected and click <b>Next</b> .
2	For Are you able to identify the Message ID? click <b>Yes</b> .
3	Click and drag to select <b>Established</b> and click <b>Next</b> .  <b>Note:</b> The Message ID is the most important part of the message. Identifying the Message ID requires evaluation of the logs to determine the type of information you want to report on.
4	Click <b>Next</b> .
5	Do NOT check the <b>Override enVision time stamp</b> .  <b>Result:</b> This uses the enVision Time Stamp, which associates the time of collection with the event.  <b>Note:</b> If Override enVision Time Stamp were checked, the enVision Time Stamp would not be associated with the event and you would select the event Time Stamp on this screen.
6	Click <b>Next</b> .  <b>Result:</b> The header variables window appears.
7	To define a variable for the session ID: <ul style="list-style-type: none"> <li>• Click and drag to select the <b>session ID</b> (this is the number in square brackets after Socks5), for example [4874]. Just select the number; do not select the brackets.</li> <li>• In the Variable name dropdown box, select the variable <b>hdata</b>.</li> <li>• Click <b>Assign</b>.</li> </ul> <b>Result:</b> The variable appears the table. Click <b>Next</b> .

## Exercise 4.1, continued

### Define the Header, continued

Step	Action
8	To define the payload of the event, click <b>Remaining part of event</b> .  <b>Result:</b> The remainder of the event is automatically selected.
9	Click <b>Next</b> .  <b>Result:</b> You should see a status of Header parsed successfully as well as the header definition.

**Define Header**

Review the summary of the header definition. Click Finish to add the header definition to the event source XML.

**Summary**

Status:

✔ **Header parsed successfully**

Selected Event:

Socks5[4874]: TCP Connection Established: Connect(172.30.21.43:37424 to 172.30.32.92:80) for user roc

Header Definition:

Header Element	Selected Value	Parsed Value
Header ID	0001	0001
Message ID	Established	Established
<b>Variables</b>		
hdata	4874	4874
Payload	Connect(172.30.21.43:37424 t...	Connect(172.30.21.43:37424 to 17...

10	Note that there is also a window at the bottom of the screen that shows the log file and which events have parsed. (You may have to move the bar or click the maximize icon to expand the window.) The status of this message is Not parsed (black).
----	--

unknownsyslog.unx

Parsed (0)    
  Header parsed and message not parsed (0)    
  Not parsed (193)

█ Socks5[4874]: TCP Connection Established: Connect(172.30.21.43:37424 to 172.30.32.92:80) for user root  
█ Socks5[4874]: TCP Connection Terminated: Normal(172.30.21.43:37424 to 172.30.32.92:80) for user root: 86 bytes out, 293 bytes in  
█ Socks5[4921]: TCP Connection Request: Connect(172.30.21.43:37425 to 172.30.32.93:80) for user root  
█ Socks5[4921]: TCP Connection Established: Connect(172.30.21.43:37425 to 172.30.32.93:80) for user root

11	Click <b>Finish</b> .  <b>Result:</b> A dialog box appears asking if you want to define the message.
12	Click <b>Yes</b> .

## Exercise 4.1, continued

### Define the Message

The message is the payload portion of the header and consists of message variables, static text and message elements including Vendor ID, enVision ID, Severity Level, Event Category and Functions.

Follow these steps to define the message using static text, variables, enVision ID, Severity Level and Event Category. Functions will be used in later exercises.

Step	Action
1	<p>The enVision ID and Vendor ID display as <b>Established</b>. This is the Message ID you created in the previous activity (Define the Header).</p> <p>Leave <b>Use the enVision ID displayed above</b> selected to use Established as the Message ID.</p>
2	Click <b>Next</b> .
3	<p>In the Severity Level: dropdown box, select <b>5 - Notice</b>.</p> <p>The severity level defines the severity of the event and determines how enVision responds when the event occurs. The severity level is required for analysis and reporting in enVision. For a list of Severity Levels, refer to the ESI online help.</p>
4	Click <b>Next</b> .
5	<p>To define the local address and local port variables:</p> <ul style="list-style-type: none"> <li>• Select the first ip address, for example, <b>172.30.21.43</b>.</li> <li>• Leave Assign the value to: <b>Variable in enVision table</b> selected.</li> <li>• Click the <b>Search and Assign</b> link.</li> <li>• Scroll down to the Firewall Accounting table.</li> <li>• Select <b>laddr</b> and click <b>Select</b>.</li> <li>• Select the first port number, for example, <b>37425</b>. Leave Assign the value to: <b>Variable in enVision table</b> selected.</li> <li>• Click the <b>Search and Assign</b> link. (Note that the Firewall Accounting table is the only table that appears in the list since you have already assigned a variable from this table.)</li> <li>• Select <b>lport</b> and click <b>Select</b>.</li> </ul>

## Exercise 4.1, continued

### Define the Message, continued

Step	Action
6	To define the foreign address and foreign port variables: <ul style="list-style-type: none"> <li>• Select the next IP address. for example, <b>172.30.32.93</b>.</li> <li>• Leave Assign the value to: <b>Variable in enVision table</b> selected.</li> <li>• Click the <b>Search and Assign</b> link.</li> <li>• Select <b>faddr</b> and click <b>Select</b>.</li> <li>• Select the next port number, for example <b>80</b>.</li> <li>• Leave Assign the value to: <b>Variable in enVision table</b> selected.</li> <li>• Click the <b>Search and Assign</b> link.</li> <li>• Select <b>fport</b> and click <b>Select</b>.</li> </ul>
7	To define the user variable: <ul style="list-style-type: none"> <li>• Select the user name, for example, <b>root</b>.</li> <li>• Leave Assign the value to: <b>Variable in enVision table</b> selected.</li> <li>• Click the <b>Search and Assign</b> link.</li> <li>• Select <b>username</b> and click <b>Select</b>.</li> </ul> <p><b>Result:</b> You should have five variables defined in the table.</p>
8	Click <b>Next</b> .
9	To select the event category for the message, click <b>Search</b> .
10	Since this message reports on a connection that was established, we will select a connection successful category as follows: <ul style="list-style-type: none"> <li>• Click the plus sign to expand <b>Network</b>.</li> <li>• Expand <b>Connections</b>.</li> <li>• Select <b>Successful</b>.</li> <li>• Click <b>Select</b>.</li> </ul> <p><b>Result:</b> Network.Connections.Successful appears in the Event category field.</p>
11	Click <b>Next</b> .

## Exercise 4.1, continued

### Define the Message, continued

Step	Action
12	<p>Click <b>Next</b> twice.</p> <p>We will not add any static values or functions in the exercise.</p> <p><b>Result:</b> You should see a status of Message parsed successfully as well as the header definition.</p>

✔ **Message parsed successfully**

Selected Event:

Socks5[4921]: TCP Connection Established: Connect(172.30.21.43:37425 to 172.30.32.93:80) for user root

Message Definition:

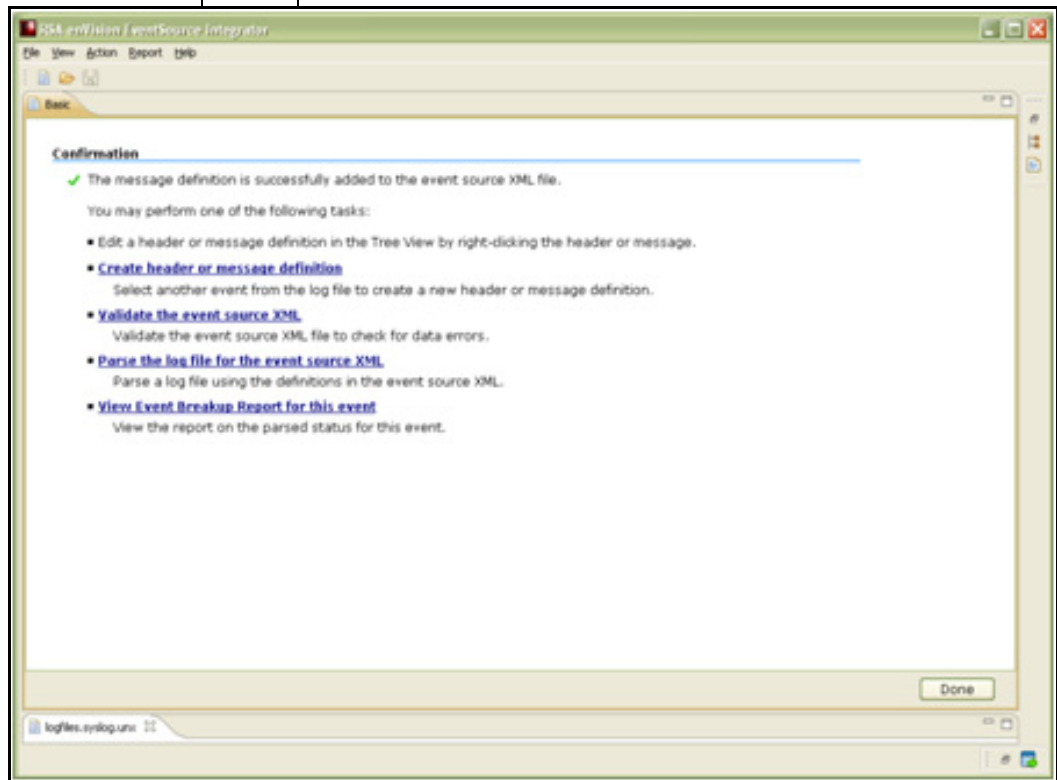
Message Elements	Selected Value	Parsed Value
enVision ID (id1)	Established	Established
Vendor ID (id2)	Established	Established
Severity Level	5 : Notice	5 : Notice
Event Category	Network.Connections.Successful	Network.Connections.Successful
enVision Table	FireWall Accounting	FireWall Accounting
<b>Variables</b>		
laddr	172.30.21.43	172.30.21.43
lport	37425	37425
faddr	172.30.32.93	172.30.32.93
fport	80	80
username	root	root



## Exercise 4.1, continued

### Define the Message, continued

Step	Action
13	<p>Click <b>Finish</b>.</p> <p><b>Result:</b> The confirmation screen appears where you can perform various tasks.</p>


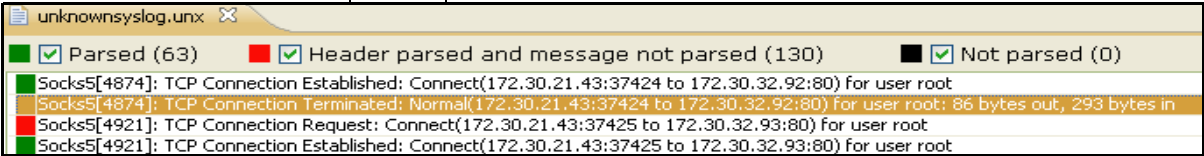


	Do <b>NOT</b> click Done.
--	---------------------------

## Exercise 4.1, continued

### Validate the XML File and Parse the Message

Follow these steps to validate the XML file and parse the message.

Step	Action
1	<p>From the Confirmation screen, click hyperlink: <b>Validate the event source xml</b>.</p> <p><b>Result:</b> If there are no errors, a message appears with No Data Errors.</p>
2	<p>Click <b>OK</b>.</p>
3	<p>Open the Parse log view if the view is not visible.</p> <p> If the view is closed, you should see an icon in the left or right panel. Click this icon.</p> <p>Note the status of the message. The message status is Not parsed.</p>
4	<p>Click the hyperlink: <b>Parse the log file for the event source XML</b>.</p> <p><b>Result:</b> The status of the message you just created changes from black to green indicating that it is now Parsed.</p>
 <p>The screenshot shows a window titled 'unknownsyslog.unx' with three filter buttons: 'Parsed (63)' (checked, green), 'Header parsed and message not parsed (130)' (checked, red), and 'Not parsed (0)' (unchecked, black). Below the filters, four log entries are visible: a green entry for 'Socks5[4874]: TCP Connection Established', a red entry for 'Socks5[4874]: TCP Connection Terminated', a red entry for 'Socks5[4921]: TCP Connection Request', and a green entry for 'Socks5[4921]: TCP Connection Established'.</p>	
5	<p><b>Uncheck the Header parsed and message not parsed (red) and Not parsed (black).</b></p> <p><b>Result:</b> The screen displays only parsed messages.</p>
6	<p>Click the <b>Header parsed and message not parsed (red)</b>.</p> <p><b>Result:</b> The Header for all messages are parsed because the messages contain common information: <b>Socks5</b>. If you had not created a variable for the process id, only the headers with the same process id would have parsed, for example, <b>Socks5:[4921]</b>.</p>
7	<p>Click <b>Done</b>.</p> <p><b>Result:</b> The Basic wizard appears where you can create a new XML file or open an existing XML file.</p>

# SWEET TATION

S  
E  
T  
T  
I  
N  
G

## Modifying an Existing Event Source XML File

- ▶ Event source XML files may be edited regardless of how they were created
  - All XML files are located in `..\etc\devices\`
  - Can use text editor to modify XML code, or use EventSource Integrator
- ▶ Do not change the order of tags in any existing XML file
  - Header followed by Message
  - The order of the elements within a Message
- ▶ The Header and Message definitions can be seen in EventSource Integrator in Tree View or Code View
  - All the Header definitions (Tree View) and tags (Code View), followed by all the Message definitions (Tree View) and tags (Code View)

---

# Modifying Event Source XML Files

You can edit event source XML files that were created by either the RSA enVision EventSource Integrator or any other source. When you edit an XML file that was created by another source, you cannot edit elements that are presently not supported by EventSource Integrator, such as value maps, regular expressions (REGX), summary buckets, data reduction commands, and functions that are not supported. However, you can edit these elements using the Code View of ESI.

---

**Note:** When you edit an event source XML file, ensure that you understand the impact of the changes that you make on event parsing. For example, when you change definitions, the number of events parsed may differ from the number of events previously parsed.

---

## Edit Modes

You can edit an event source XML file using one of the following features:

- **Quick Edit**

You can use Quick Edit to quickly change certain elements in the header or message definition. For example, if you want to change a header variable, you can select a new variable, or, if you want to change the enVision ID, you can enter a new enVision ID. The Quick Edit feature is available only in the Tree View tab, on the right-hand side of the EventSource Integrator window.

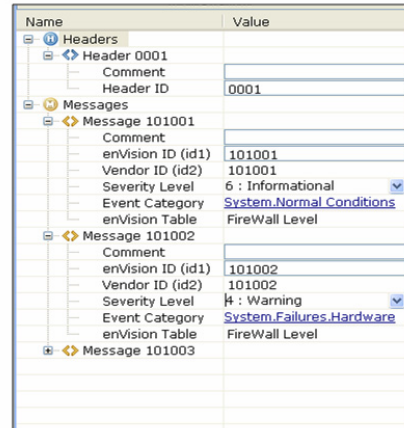
- **Complete Edit**

You can use the Complete Edit feature to edit all the elements defined for an event in the event source XML. For example, if the text selected in the event for the Message ID in the header definition is incorrect, you can select new text in the event and define it as the Message ID, or you can change the message variable definition by selecting a variable from a different enVision table.

S  
E  
E  
T  
O  
N

## EventSource Integrator - Tree View

- ▶ Tree View list the elements as definitions
- ▶ Listed by name, with their respective elements
- ▶ If an XML file contains syntax errors, it cannot be opened in Tree View
- ▶ You can edit some parts of a message by selecting in Tree View (Quick Edit)



## EventSource Integrator - Code View

Code View shows the definitions as XML tags

```

26
27 The following are the entity referen
28 &lt; &lt;(opening angle bracket)
29 &gt; &gt;(closing angle bracket)
30 &amp; &(ampersand)
31 &quot; &(double quotation mark)
32
33-->
34
35 <HEADER
36   id1="0001"
37   id2="0001"
38   content="%CERT-1-&lt;messag
39
40 <MESSAGE
41   level="6"
42   parse="1"
43   parsedefvalue="0"
44   tableid="1"
45   id1="101001"
46   id2="101001"
47   eventcategory="1605000000"
48   content="(PRIORITY) Failover c
49
50 <MESSAGE
51   level="4"
52   parse="1"
53   parsedefvalue="0"
54   tableid="1"
55   id1="101002"
56   id2="101002"
57   eventcategory="1601010000"
58   content="(PRIORITY) Bad failov

```



## Modifying Event Source XML Files, continued

### Complete Edit

When you use Complete Edit, you must select an event in the log file in addition to selecting the header or message definition in the event source XML file. Typically, you use Complete Edit to correct definitions in the event source XML file. Using Complete Edit, you can also make all the changes that you can make using Quick Edit. Once you save the XML file, the XML definitions are updated in both Tree and Code View tabs.

You can edit the following elements using Complete Edit:

- Header Elements
  - Header ID
  - Message ID
  - Event Source Time Stamp
  - Header Variables
  - Payload
  - Comment
- Message Elements
  - enVision ID
  - Severity Level
  - Message Variables
  - Event Category
  - Static Value
  - Functions
  - Comment

### Tree View

To edit in Tree View, right-click on the element and select Edit. You can edit the following elements using Quick Edit:

- Header Elements
  - Header ID
  - Header Variables
  - Comment
- Message Elements
  - enVision ID
  - Severity Level
  - Message Variables
  - Event Category
  - Static Value
  - Comment

### Code View

You can view the header and message definitions in the XML format that RSA enVision uses in the Code View tab. All the definitions are displayed as XML tags. You can change any of the definitions in the Code View tab. If you make changes to a header or message definition in the code view and save the changes, the tree view is automatically updated.

You can perform the following actions in the Code View tab:

- Edit header or message tags.
- Copy and paste a header or message definition to another position.
- Edit header and message elements.

**Note:** While creating or editing an XML definition using the wizard, you cannot edit the XML in the code view.

S  
E  
T  
T  
I  
N  
G

## Editing an Existing XML File

- ▶ RSA recommends making edits in the Tree View to avoid changes that do not conform to the event source XML schema, or that are in an incorrect sequence
  - Changing the order of messages may produce unexpected results
- ▶ The following cannot be edited in EventSource Integrator's Tree View (can only be edited in Code View):
  - Value maps
  - Regular expressions (REGX)
  - Summary buckets
  - Data reduction commands
  - Message ID





## Modifying Event Source XML Files, continued

### Editing an Existing XML File

RSA recommends that you:

- Make changes only in the Tree View tab to avoid changes that do not conform to the event source XML schema or that are in an incorrect sequence. These discrepancies would make the XML file invalid.
- Do not change the order of the header definitions followed by message definitions nor the order of the elements within a message definition because such changes may lead to incorrect definitions. (However, you can change the order within the list of message definitions.)

The order of message definitions is very important for precedence. RSA enVision reads the message definitions in the order that they appear in the XML file.

For example, an event source XML file contains the following messages:

```
Message 10123 < A B C >
```

where A, B, and C are elements in message

```
Message 10124 < A B C D >
```

where A, B, C, and D are elements in message

If the order of the message definitions is as shown and an A B C event from the event source is sent to enVision, the event is parsed using Message 10123. When an A B C D event is sent from the event source to enVision, the event is also parsed using the Message 10123, although it should be parsed against Message 10124. Therefore, you must ensure that Message 10124 appears before Message 10123 in the XML as shown below:

```
Message 10124 <A B C D>
```

```
Message 10123 < A B C >
```

# SECTION

## Cloning

- ▶ A clone of a header or message definition can be created in the Tree View and the elements edited to create a definition for another event
- ▶ When cloning, a copy of all the elements defined for the header or message is created
  - A unique Header ID must be specified to clone a header definition
  - A unique Message ID needs to be used to clone a message definition
- ▶ Elements within a header or message definition cannot be cloned
  - For example, you cannot clone a message variable in a message definition



---

## Modifying Event Source XML Files, continued

### Cloning

You can clone either a header definition or a message definition on the Tree View tab on the right-hand side of the window.

To clone a header:

- In the Tree View tab, select the **header** that you want to clone.
- Right-click the **header**, and click **Clone**.
- Specify the **header ID**. By default, a unique header ID is generated. If you want to change the header ID, enter a unique header ID in the Clone Header Definition pop-up window.
- Click **Save**.

Edit the header elements as required. To clone a message:

- In the Tree View tab, select the **message** that you want to clone.
- Right-click the **message**, and click **Clone**.
- Specify the **enVision ID**. By default, the enVision ID of the selected message is displayed. You must enter a unique enVision ID in the Clone Message Definition pop-up window.
- Click **Save**.
- Edit the message elements as required.

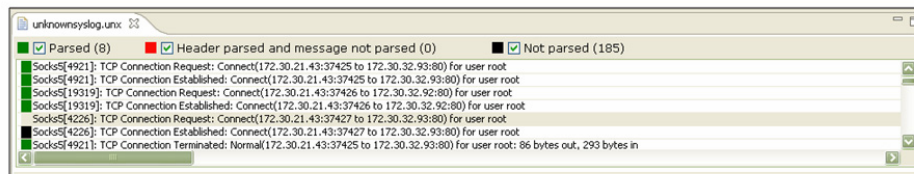
S  
W  
E  
E  
T  
O  
N

## Validation and Parsing

- ▶ You can validate the XML to ensure there are no errors



- ▶ You can parse the log to see if the header, message or both are parsed



# Validation and Parsing

## Validating an XML File

You can validate the message variables in the event source XML file for data pattern errors after defining the event source XML file. For example, the value specified for the username variable is validated not to contain special characters other than \*, +, or @. Another example is that the value specified for protocol is validated to contain only protocols such as icmp, tcp, ip, udp, http, https, ftp, telnet, or ntp.

You must validate the event source XML file using the associated log file. When you validate the event source XML file, the Validation Results tab displays any errors with details such as the Message ID that contains the error, the line number in the log file of the error, and a description of the error. This information can help resolve errors in the event source XML.

### Validating XML Errors in Code View

If an event source XML file contains errors, opening the file in code view will indicate the exact location of the error and allows you to edit the XML definitions and validate the edited XML file.

Errors are highlighted with an error marker on the left side of the code view and a red bar on the right side. Clicking an error marker launches a pop-up window displaying information about the error; clicking the red bar highlights the element tag that contains the error.

The following errors are shown in the Code View:

- **Syntax errors.** Syntax errors occur when an XML tag for an element is missing or incorrectly placed. For example, if the required closing tag of the message element is missing, a syntax error occurs. If there are multiple syntax errors in the same element tag, they are highlighted using the error marker and red bar one at a time. After resolving an error, the next error is highlighted.

**Note:** You can only open an event source XML file that contains syntax errors in code view. After you correct the errors, you can open the file in the wizard mode.

- **Semantic Errors.** Semantic errors occur when a mandatory element attribute in an element tag is missing or is incorrect. For example, if one of the required attributes for id1 and id2 is missing in a header element tag, a semantic error occurs. Semantic errors do not occur when optional attributes are missing or incorrect. If there are multiple semantic errors in the same element tag, all the errors are displayed.

SWEFTON

## Verifying the Event Source XML File Reporting

- ▶ Event Breakup Report – This report lists the parsed status and detailed break up of header and message definitions for each event
- ▶ Use the details of this report to identify incorrect or out of order definitions

XML file name: Maxfirewall.xml Export to: [Print](#) | [E](#)

Event log file name: Lab1syslog.unx


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

**Header Definition**

Sl.No	Parsed Status	HeaderID
1	Yes	0001
Header Element		Parsed Value
%CERT-1-		%CERT-1-
<messageid>		101001
:		:
<payload>		(PRIORITY) Failover cable OK.

**Message Definition**

Sl.No	Parsed Status	Vendor ID (id2)	enVision ID(id1)	enVision Table	Event Category
1	Yes	101001	101001	FireWall System	System.Normal Conditions
Message Element		Parsed Value			
(		(			
<priority>		PRIORITY			
) Failover cable OK.		) Failover cable OK.			



The Security Division of EMC 105

## Validation and Parsing, continued

### Parsing

You can parse the log file to see the status of the message. The Log View windows displays events in the following states:

- Not Parsed
- Header parsed and message not parsed
- Parsed

You can select specific states to view only those states.

To parse a log file using an event source XML file:

- Open the event source XML file.
- Click **File > Open**, and select the log file that you want to parse.
  - If you have more than one log file open, in the Log View area, click the tab of the log file that you want to parse.
- Click **Action > Parse Log**. Alternatively, press **F8**.
  - In the Log View area, you will see the parse status of the events displayed.

### Reporting

You can also generate reports to analyze how the log messages are parsed by the header and message definitions in the event source XML file.

Each selected event is parsed against each header definition and against each message definition that has the same vendor ID - id2 - as the first completely-parsed header definition.

The report provides a complete breakdown of the definitions, shows the parsed values for associated variables and static text, and provides the parsed status of each definition. This report can be used for debugging. Only message definitions that match the Vendor ID are displayed

Reports can be generated on:

- Selected events in the log file.
- All events in the log file.

# SECTION



## Exercise 4.2 Create Additional Messages

### The goal of this exercise is:

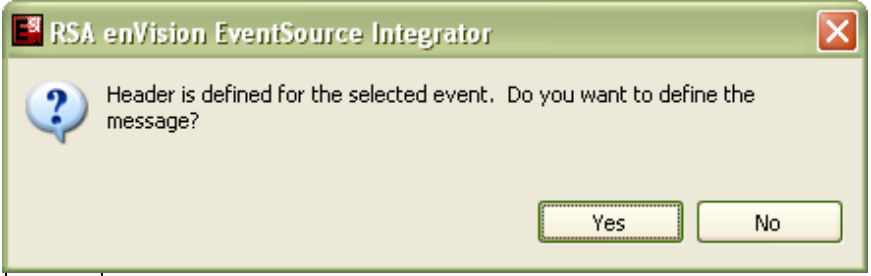
- Add three more messages to the existing BulldogDogHousemsg file

This exercise consists of four parts:

- Create a message for the Request action using the wizard
- Create a message for the Terminated Normal action using the wizard
- Edit the Terminated Normal message to add variables
- Create a message for the Terminated Abnormal action using cloning

### Create the Request Message

Follow these steps to create the message for the Request action.

Step	Action
1	<ul style="list-style-type: none"> <li>• From the Confirmation screen, click <b>Create header or message definition</b>, or</li> <li>• From the Introduction screen, click <b>Open XML</b>, browse to the desktop and open the XML you created in Exercise 4, <b>BulldogDogHousemsg.xml</b> (the file may already be open) and click <b>Start</b>.</li> </ul>
2	Click <b>Select an Event</b> .
3	Click on a <b>Request</b> event.
4	Click <b>Select</b> .
5	<p>Click <b>Next</b>.</p> <p><b>Result:</b> A dialog box appears reporting that the header is already defined.</p>
	
6	<p>Click <b>Yes</b>.</p> <p><b>Result:</b> The Define Message screen appears with the Message ID Request.</p>

## Exercise 4.2, continued

### Create the Request Message, continued

Step	Action
7	Leave <b>Use the enVision ID displayed above</b> selected to use Request as the Message ID.
8	Click <b>Next</b> .
9	In the Severity Level: dropdown box, select <b>5 - Notice</b> .
10	Click <b>Next</b> .
11	<p>To define the local address and port variables:</p> <ul style="list-style-type: none"> <li>• Select the first IP address, for example, <b>172.30.21.43</b>.</li> <li>• Leave Assign the value to: <b>Variable in enVision table</b> selected.</li> <li>• Click the <b>Search and Assign</b> link.</li> <li>• Scroll down to the <b>FireWall Accounting</b> table.</li> <li>• Select <b>laddr</b> and click <b>Select</b>.</li> <li>• Select the first port number, for example, <b>37425</b>.</li> <li>• Leave Assign the value to: <b>Variable in enVision table</b> selected.</li> <li>• Click the <b>Search and Assign</b> link.</li> <li>• Select <b>lport</b> and click <b>Select</b>.</li> </ul>
12	<p>To define the foreign address and port variables:</p> <ul style="list-style-type: none"> <li>• Select the next IP address, for example, <b>172.30.32.93</b>.</li> <li>• Leave Assign the value to: <b>Variable in enVision table</b> selected.</li> <li>• Click the <b>Search and Assign</b> link.</li> <li>• Select <b>faddr</b> and click <b>Select</b>.</li> <li>• Select the next port number, for example, <b>80</b>.</li> <li>• Leave Assign the value to: <b>Variable in enVision table</b> selected.</li> <li>• Click the <b>Search and Assign</b> link.</li> <li>• Select <b>fport</b> and click <b>Select</b>.</li> </ul>

## Exercise 4.2, continued

### Create the Request Message, continued

Step	Action
13	To define the user variable: <ul style="list-style-type: none"> <li>• Select the user name, for example, <b>root</b>.</li> <li>• Leave Assign the value to: <b>Variable in enVision table</b> selected.</li> <li>• Click the <b>Search and Assign</b> link.</li> <li>• Select <b>username</b> and click <b>Select</b>.</li> </ul> <b>Result:</b> You should have five variables defined in the table.
14	Click <b>Next</b> .
15	To select the event category for the message, click <b>Search</b> .
16	Since this message reports on a connection that was requested but not established or terminated, just select the connection category as follows: <ul style="list-style-type: none"> <li>• Click the plus sign to expand <b>Network</b>.</li> <li>• Select <b>Connections</b>.</li> <li>• Click <b>Select</b>.</li> </ul> <b>Result:</b> Network.Connections appears in the Event category field.
17	Click <b>Next</b> .
18	Click <b>Next</b> twice.  We will not add static values or functions in the exercise.  <b>Result:</b> You should see a status of Message parsed successfully as well as the header definition.
19	Click <b>Finish</b> .  <b>Result:</b> The confirmation screen appears.
20	Select <b>Parse the log file for the event source XML</b> and view the results in the parse log view.  <b>Result:</b> The Request messages are now green - Parsed.
21	Click <b>Done</b> .

## Exercise 4.2, continued

### Create the Terminated Normal Message

Follow these steps to create the message for the Terminated Normal action.

Step	Action
1	<ul style="list-style-type: none"> <li>• From the Confirmation screen, click <b>Create header or message definition</b>, or</li> <li>• From the Introduction screen, click <b>Open XML</b>, browse to the desktop and open the XML you created in Exercise 4, <b>BulldogDogHousemsg.xml</b> (the file may already be open) and click <b>Start</b>.</li> </ul>
2	Click <b>Select an Event</b> .
3	Click on a <b>Terminated Normal</b> event.
4	Click <b>Select</b> .
5	Click <b>Next</b> .  <b>Result:</b> A dialog box appears reporting that the header is already defined.
6	Click <b>Yes</b> .  <b>Result:</b> The Define Message screen appears with the Message ID Terminated.
7	Leave <b>Use the enVision ID displayed above</b> selected to use <b>Terminated</b> as the Message ID.
8	Click <b>Next</b> .
9	In the Severity Level: dropdown box, select <b>5 - Notice</b> .
10	Click <b>Next</b> .

## Exercise 4.2, continued

### Create the Terminated Normal Message, continued

Step	Action
11	<p>To define the local address and port variables:</p> <ul style="list-style-type: none"> <li>• Select the first IP address, for example, <b>172.30.21.43</b>.</li> <li>• Leave Assign the value to: <b>Variable in enVision table</b> selected.</li> <li>• Click the <b>Search and Assign</b> link.</li> <li>• Scroll down to the <b>FireWall Accounting</b> table.</li> <li>• Select <b>laddr</b> and click <b>Select</b>.</li> <li>• Select the first port number, for example, <b>37425</b>.</li> <li>• Leave Assign the value to: <b>Variable in enVision table</b> selected.</li> <li>• Click the <b>Search and Assign</b> link.</li> <li>• Select <b>lport</b> and click <b>Select</b>.</li> </ul>
12	<p>To define the foreign address and port variables:</p> <ul style="list-style-type: none"> <li>• Select the next IP address, for example, <b>172.30.32.93</b>.</li> <li>• Leave Assign the value to: <b>Variable in enVision table</b> selected.</li> <li>• Click the <b>Search and Assign</b> link.</li> <li>• Select <b>faddr</b> and click <b>Select</b>.</li> <li>• Select the next port number, for example, <b>80</b>.</li> <li>• Leave Assign the value to: <b>Variable in enVision table</b> selected.</li> <li>• Click the <b>Search and Assign</b> link.</li> <li>• Select <b>fport</b> and click <b>Select</b>.</li> </ul>
13	<p>To define the user variable:</p> <ul style="list-style-type: none"> <li>• Select the user name, for example, <b>root</b>.</li> <li>• Leave Assign the value to: <b>Variable in enVision table</b> selected.</li> <li>• Click the <b>Search and Assign</b> link.</li> <li>• Select <b>username</b> and click <b>Select</b>.</li> </ul> <p><b>Result:</b> You should have five variables defined in the table.</p>

## Exercise 4.2, continued

### Create the Terminated Normal Message, continued

Step	Action
14	Click <b>Next</b> .
15	To select the event category for the message, click <b>Search</b> .
16	<p>Since this message reports on a connection that was requested but not established or terminated, just select the connection category as follows:</p> <ul style="list-style-type: none"> <li>• Click the plus sign to expand <b>Network</b>.</li> <li>• Expand <b>Connections</b> and select <b>Terminations</b>.</li> <li>• Click <b>Select</b>.</li> </ul> <p><b>Result:</b> Network.Connections.Terminations appears in the Event category field.</p>
17	Click <b>Next</b> .
18	<p>Click <b>Next</b> twice.</p> <p>We will not add static values or functions in the exercise.</p> <p><b>Result:</b> You should see a status of Message parsed successfully as well as the header definition.</p>
19	<p>Click <b>Finish</b>.</p> <p><b>Result:</b> The confirmation screen appears.</p>
20	<p>Select <b>Parse the log file for the event source XML</b> and view the results in the parse log view.</p> <p><b>Result:</b> The Terminated messages did not parse because the bytes variables were not assigned.</p>
21	Click <b>Done</b> .

## Exercise 4.2, continued

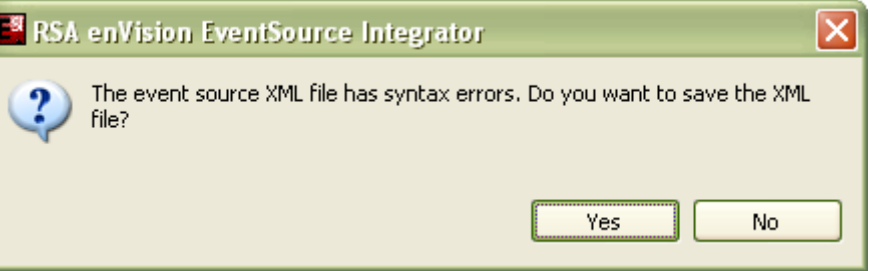

### Edit the Terminated Normal Message

Follow these steps to edit the Terminated Normal message to add the bytes variables.

Step	Action
1	In the Tree View, right-click <b>Message Terminated</b> .
2	Select <b>Edit</b> .
3	For Select a log file, click <b>Browse</b> and select <b>syslogDogHouse.unx</b> .
4	Click <b>Select an Event</b> .
5	Click the <b>Terminated</b> message you just created and click <b>Select</b> .
6	Click <b>Next</b> three times. <b>Result:</b> The Variables screen appears.
7	To define the sent and received bytes variables: <ul style="list-style-type: none"> <li>• Select <i>&lt;number of bytes&gt;</i> out.</li> <li>• Leave Assign the value to: <b>Variable in enVision table</b> selected.</li> <li>• Click the <b>Search and Assign</b> link.</li> <li>• Select <b>sbytes</b> and click <b>Select</b>. Select <i>&lt;number of bytes&gt;</i> in.</li> <li>• Leave Assign the value to: <b>Variable in enVision table</b> selected.</li> <li>• Click the <b>Search and Assign</b> link.</li> <li>• Select <b>rbytes</b> and click <b>Select</b>.</li> </ul>
8	Click <b>Next</b> four times.
9	Click <b>Finish</b> . <b>Result:</b> The confirmation screen appears.
10	Click <b>Done</b> .
11	Open the Code View window and scroll to the bottom to the Terminated Message.
12	Scroll the way to the right in the content section and see if the sbytes and rbytes variables were added.
13	Remove the last bracket from the end of the line (>).

## Exercise 4.2, continued

### Edit the Terminated Message

Step	Action
14	Select <b>Save</b> from the File menu.  <b>Result:</b> A dialog box appears saying that the XML file has errors.
	
15	Click <b>Yes</b> .
16	Open the Code View  .
17	Scroll to the bottom of the file and notice the red marks indicating the error.
18	Add the bracket (>) to the end of the line.
19	Select <b>Save</b> from the File menu.  <b>Note:</b> The message does not appear. There are no errors.
20	From the Action menu, select <b>Parse log</b> .
21	Review the parse log view. to see if the messages parsed.  <b>Result:</b> The Terminated: Normal message should be parsed.
22	From the Report menu, select <b>Event Analysis &gt; Selected Events</b> .
23	Select the events that were parsed and one that was not parsed.
24	Click <b>Generate Report</b> .
25	Click <b>Continue</b> .
26	Review the report.





## Exercise 4.2, continued

### Create the Terminated Abnormal Message

In a previous activity you added a message for the Terminated Normal action. In this activity, you will add a message for the Terminated Abnormal action using the cloning function to save time.

Follow these steps to create a message for the Terminated Abnormal action using cloning.

Step	Action
1	Click the  icon in the right panel to open the tree view.  Notice the three messages that you just created: Established, Request, Terminated.
2	Expand the messages by clicking the plus sign to see the information contained in the message.
3	Right-click the <b>Message Terminated</b> .
4	Select <b>Clone</b> .
5	For the enVision ID, enter <b>Terminated:01</b> .
6	Click <b>Save</b> .  <b>Result:</b> The new Message appears in the tree view.
7	Click the plus sign (+) next to <b>Message Terminated:01</b> .  <b>Result:</b> The Message elements appear.
8	Since this action is Terminated Abnormal, change the Severity Level by clicking the down arrow and selecting <b>4 - Warning</b> .
9	Select the <b>Code View</b> tab or the Code View icon  .
10	Scroll down to the last message (Terminated:01).
11	In the content= section, change “Normal” to “ <b>Abnormal</b> ”.
12	From the File menu, select <b>Save</b> to save the XML code.  <b>Note:</b> You must save the XML for the change to appear in the Tree View.
13	Select <b>Parse the log file for the event source XML</b> and view the results in the parse log view.  <b>Result:</b> The Abnormal messages should be parsed.
14	Click <b>Done</b> .

# SEVENTH ON

## Event Source Package

- ▶ The Event Source Package contains the following files:
  - eventsourcename.ini
  - eventsourcenameclient.txt.
  - eventsourcenamevendor.txt
  - eventsourcenamemsg.xml
  - UpdateESType.vbs



---

## Event Source Package

The Event Source Package contains a group of files that are needed to support the new device on enVision. These files are created as a result of the deployment process in ESI.

EventSource Integrator creates the support files (.ini and two .txt files). ESI then compresses these files along with the .XML file and a VBScript file, creating an *<eventsource>*.zip file, known as the package.

The package (.zip file) is then copied and deployed on the enVision machine to allow for discovery of the event source.

An earlier slide featured the Solaris support folder that included solaris.ini, solarisvendor.txt and solarisclient.txt files, along with a few others. These are the files that comprise the package.

S  
E  
T  
O  
N

## Deploying the Event Source Files

- ▶ Deployment is the process of finalizing and installing all support files for the new event source
- ▶ Create the event source package
  - Open the event source XML file that you created using RSA enVision EventSource Integrator.
  - Click **File > Create Event Source Package**.
  - In the Create Event Source Package pop-up window, enter or browse to the path where you want to save the package.
  - Click **Create**.
- ▶ Result: event source package = <eventsourcename>.zip



## Deploying the Event Source Files, continued

Deploy the event source package:

1. Create a new folder in every `..\etc\devices` directory
  - Just the <eventsourcename>
2. Copy the zip file (event source package) into this new folder
3. Expand the zip file
4. Run the VBscript file, by double-clicking
5. Restart the NIC Service Manager Service

**Note:** The event source package must be deployed on every enVision appliance



---

## Deploying the Event Source Files

Deployment is the process of finalizing and installing all support files for the new event source.

To create the event source package:

- Open the event source XML file that you created using RSA enVision EventSource Integrator.
- Click **File > Create Event Source Package**.
- In the Create Event Source Package pop-up window, enter or browse to the path where you want to save the package.
- Click **Create**.

The result will be an event source package created with the name *<eventsourcename>.zip*.

To deploy the event source package:

- Create a new folder in the `.\etc\devices` directory (on every D-SRV) with the name *<eventsourcename>*.
- Copy the zip file (event source package) into this new folder on the first D-SRV.
- Expand the zip file.
- Run the VBscript file, by double-clicking the script file.
  - The script, `UpdateESType.vbs`, assigns an event source type ID to the event source.
  - The time the script file takes to run depends on the number of event source XML files that need to be verified.
- Edit the `.ini` file to add the name of the event source in the `DisplayName` field.
- Save the `.ini` file.
- Copy the `.ini` file to the D-SRV on each enVision site.
- Restart the NIC Service Manager Service.

---

Note: It is important that you run the script on the first machine and then copy the `.ini` file to the other machines in the site. Otherwise, there is a risk of different ids being created on different machines.

---

# SECTION

## Exercise 4.3 Create, Deploy, and Test the Event Source Package

### The goal of this exercise is:

- Create the package for the XML support files
- Deploy the files
- Test the event source integration

This exercise consists of two parts:

- Create and deploy the event source
- Test the event source integration

### Create and Deploy the Event Source

To get the event source files into enVision, you must create a package (eventsourcefilename.zip) in ESI that includes the following support files:

- eventsourcename.ini
- eventsourcenameclient.txt.
- eventsourcenamevendor.txt
- eventsourcenamemsg.xml
- UpdateESType.vbs

Follow these steps to create and deploy the package for the Bulldog DogHouse event source.

Step	Action
1	From the File menu, select <b>Create Event Source Package</b> .  <b>Result:</b> The Create Event Source Package dialog box appears.
2	In the event source package path, click <b>Browse</b> .
3	Select: <b>c:\Documents and Settings\Administrator\Desktop\BulldogDogHousemsg</b>
4	Click <b>Create</b> .  <b>Result:</b> A dialog box appears stating that the package was successfully created with a link to the directory.  Do not click the directory link since we will move the file manually.

## Exercise 4.3, continued

### Create and Deploy the Event Source, continued

Step	Action
5	<p>Create a folder called BulldogDogHousemsg in the directory:  <b>..\nic\4000\&lt;nodename&gt;\etc\devices\BulldogDogHouse</b></p> <p><b>Note:</b> The package MUST be in a folder with the same name as the event source package (in this example, BulldogDogHousemsg).</p>
6	<p>Copy the BulldogDogHousemsg.zip file to the directory created in step 5:</p> <p><b>c:\nic\4000\&lt;nodename&gt;\bin\etc\devices\BulldogDogHouse\BulldogDogHousemsg</b></p>
7	<p>Unzip the BulldogDogHousemsg.zip file.</p> <p><b>Note:</b> Make sure you are unzipping to the correct folder.</p> <p><b>Result:</b> The event source files appear in the directory.</p>
8	<p>Doubleclick the <b>UpdateESType.vbs</b> to run the script to assign an event source type ID to the event source.</p> <p><b>Note:</b> The time the script file takes to run depends on the number of event source XML files that need to be verified.</p> <p><b>Result:</b> An event source type is returned.</p>
9	<p>Open the .ini file in a text editor.</p>
10	<p>Change the DisplayName to <b>Bulldog DogHouse</b>.</p>
11	<p>Save the .ini file.</p>
12	<p>Restart the NIC Service Manager Service.</p>
13	<p>Click <b>OK</b> to close the ESI dialog box.</p>



## Exercise 4.3, continued

### Test the Event Source Integration

To make sure that enVision discovers and analyzes the event source under its proper and recognizable name of BulldogDogHouse (not 'unknown'), you need to inject data from your syslogDogHouse.unx file.

But first, you should remove the unknown event sources to make sure there is no confusion as to what's new and what's old. Then you will view the event sources in enVision.

Follow these steps to test the integration.

Step	Action
1	In the enVision user interface, select <b>System Configuration &gt; Devices &gt; Manage Monitored Devices</b> .
2	Select all ' <b>Unknown</b> ' Device Types then click <b>Delete</b> .
3	Log in to enVision and go to the <b>System Configuration &gt; Devices &gt; Manage Device Types</b> . Note the Bulldog DogHouse device.
4	Log out of enVision.
5	Using Windows Explorer, delete the following directories and/or files: <b>c:\nic\4000\&lt;nodename&gt;\tmp\nuggets\unknown</b> <b>c:\nic\lsnode\data\&lt;nodename&gt;\unknown</b>
6	Restart the NIC Service Manager.
7	Run the injector utility as follows: <b>c:\class\injector\esisysloginjector.bat</b>  <b>Result:</b> As events are collected, enVision will list the data coming from the device type Bulldog DogHouse. You should have more than one device listed of this type because there are different source IP addresses contained in the syslog file.
8	Log in to enVision, and select <b>System Configuration &gt; Devices &gt; Manage Devices</b> to confirm that a Bulldog DogHouse has been discovered and is being analyzed  You will also see BulldogDogHouse directories created within nic\4000\<nodename>\tmp\nuggets as well as under the directory nic\lsnode\data\<nodename>.

## Exercise 4.3, continued

### Test the Event Source Integration, continued

Step	Action
9	<p>Select <b>Analysis &gt; Event Viewer &gt; Message View</b> to see messages received from Bulldog DogHouse.</p> <p>You may also filter displayed events by setting Bulldog DogHouse as the Device Type.</p>
10	<p>Create a Query and Report using the Firewall Accounting table with the following fields:</p> <ul style="list-style-type: none"> <li>• Device Address</li> <li>• Date/Time</li> <li>• Message ID</li> <li>• Local Address</li> <li>• Local Port</li> <li>• Local Port Name</li> <li>• Foreign Address</li> <li>• Foreign Port</li> <li>• Foreign Port Name</li> <li>• Sent Bytes</li> <li>• Received Bytes</li> </ul>
11	<p>List the fields that are empty.</p> <p>_____</p> <p>_____</p> <p>_____</p> <p>_____</p> <p>Why are these fields empty?</p>

---

# Advanced Topics

---

## Objectives

Upon completion of this unit, you should be able to:

- Describe the ESI supported functions
- Describe conditional variables
- Add a function to an XML file
- Add a value map to an XML file

SECTIONS

## Functions

- ▶ Definition: An action to be performed on a variable in an event to generate user-defined values.
- ▶ ESI supported functions:
  - PARMVAL
  - SYSVAL
  - HDR
  - CALC
  - SUM, MIN, and MAX
  - EVNTTIME



## PARMVAL Function

- ▶ Used to assign the value of a message variable to another variable
- ▶ The following actions can be performed using the PARMVAL function:
  - Assign the value of one variable to another variable
  - Assign the payload to msg variable



# Functions

A function defines an action to be performed on a variable to generate a user-defined value. You can define multiple functions for a single event but you can use a variable only in one function.

The functions that are supported within the ESI tool are:

- **PARMVAL** - assigns the value of a message variable to another variable.
- **SYSVAL** - populates the value of a message variable.
- **HDR** - assigns the value of a header variable to a variable from an enVision table.
- **CALC** - performs a calculation on values and variables in the event.
- **SUM, MIN, and MAX** - calculates the total, minimum, and maximum of values and variables in the event.
- **EVNTTIME** - assigns the date and time information in the event to a time stamp variable.

These functions can be added using the ESI wizard.

You can add other functions to the XML. However, you need to add them by editing the Code View.

## PARMVAL Function

Use the PARMVAL function to assign the value of a message variable to another variable.

For example, if you have assigned the value 47 to the lport variable from the Firewall Accounting table, you can use the PARMVAL function to assign the same value to the sport variable from the same table.

You can perform the following actions using the PARMVAL function:

- Assign the value of one variable to another variable.
  - You can assign the value of one of the message variables that you have defined to a variable from the same table that has not been previously selected. For example, you can assign the value of the dport variable from the Intrusion Detection System table to the sport variable of the same table.
- Assign the payload to a msg variable.
  - You can assign the complete payload to a msg variable from any of the RSA enVision tables. The payload will be available for generating reports and alerting in RSA enVision.

**Note:** This option is available only if the table you selected while defining the message variables contains a msg variable.

S  
E  
T  
O  
N

## SYSVAL Function

- ▶ Populates a variable with a message element.
- ▶ The following actions can be performed using the SYSVAL function:
  - Use the enVision ID instead of vendor ID
  - Assign the severity level attribute value to level variable in enVision table



## CALC Function

- ▶ Performs arithmetic calculations on the numeric values and variables that have already been defined
- ▶ For example, to add the incoming bytes and the outgoing bytes from all the events in an event source, use the "+" operation
  - No order of precedence is followed and values are evaluated from left to right.
  - This function does not support parentheses



## Functions, continued

### SYSVAL Function

Use the SYSVAL function to populate a variable with a message element. You can perform the following actions using the SYSVAL function:

- Use the enVision ID instead of the vendor ID.
  - In RSA enVision reports, vendor ID is available for selection. If you want to use the enVision ID (id1) in the reports generated by enVision, you must use this function. You might use the enVision ID because it has a better description of the event than the vendor ID or because the enVision ID differentiates between multiple events that have the same vendor ID.
- Assign the severity level attribute value to the level variable in an enVision table.
  - An event coming from a event source might be assigned a severity level based on the standards of a different organization. This standard may be different from the enVision standard of defining the severity level. Use this function to normalize the severity level as per enVision standards. After you define the value, the level variable is included in the message definition of the event source XML file.

**Note:** This option is available only if the table you selected when you defined message variables contains a level variable.

### CALC Function

Use the CALC function to perform arithmetic calculations on the numeric values and variables that you have already defined.

The following table describes the operators that the CALC function supports.

Operator	Description
+	Arithmetic addition
-	Arithmetic subtraction
*	Arithmetic multiplication
/	Arithmetic division
%	Arithmetic modulo
>>	Shift right
<<	Shift left

S  
E  
T  
O  
N

## HDR Function

Assigns the value of a header variable to another variable

- ▶ Header variables are not stored in enVision and are not available for alerting and reporting
- ▶ To retain the value of any specific header variable, use the HDR function to assign the value of a header variable to another variable in the enVision table



## SUM, MIN, MAX Functions

Perform arithmetic operations on the values and variables that you have already defined:

- ▶ SUM
  - Assigns the total of all variable values to a variable from the enVision table
- ▶ MIN
  - Assigns the lowest value of the selected variable values to a variable from the enVision table
- ▶ MAX
  - Assigns the highest value of the selected variable values to a variable from the enVision table





---

## Functions, continued

### HDR Function

Use the HDR function to assign the value of a header variable to another variable. Header variables are not stored in enVision and are not available for alerting and reporting. If you want to retain the value of any specific header variable, you can use the HDR function to assign the value of a header variable to another variable in the enVision table.

### SUM, MIN, Max Functions

Use the SUM, MIN, or MAX function to perform arithmetic operations on the values and variables that you have already defined:

- **SUM** assigns the total of all values and variables in the specified variables to a variable from the enVision table.
  - For example, if lport and fport are variables from the enVision table, you can assign the total value by entering SUM(lport,fport) in the expression.
- **MIN** assigns the lowest value in the selected variables and values to a variable from the enVision table.
  - For example, if lport and fport are variables from the enVision table, you can assign the lower value by entering MIN(lport,fport) in the expression.
- **MAX** assigns the highest value in the selected variables to a variable from the enVision table.
  - For example, if lport and fport are variables from the enVision table, you can assign the higher value by entering MAX(lport,fport) in the expression.

S  
E  
T  
O  
N

## EVNTTIME Function

- ▶ EVNTTIME function is used to:
  - Create a Time Stamp value of the event for reporting and querying
  - Construct a Time Stamp format that is acceptable to enVision
  - Normalize the Time Stamp of an event to the YYYY MM DD HH:mm:ss format and assigns it to any timestamp variable from enVision table
- ▶ Example:
  - Time Stamp is in the DD MMM YY HH:mm:ss format.
  - The EVNTTIME function normalizes to YYYY MM DD HH:mm:ss and shows it as "2009-01-03 11:06:39" in the XML file

```
Event Source Time Stamp
↑
03 Jan 09 11:06:39 [10.5.92.51] *PIX-1-105037:
The primary and standby units are switching back and forth as active units.
```



---

## Functions, continued

### EVNTTIME Function

If you want to use the event source time stamp of the event for reporting and querying, you can use the EVNTTIME function to construct a time stamp in a standard and normalized format. This function normalizes the time stamp of an event to the MM DD YYYY HH:mm:ss format and assigns it to any timestamp variable from the enVision table.

For example, in the event shown on the slide, the time stamp "03 Jan 09 11:06:39" is in the DD MMM YY HH:mm:ss format. The EVNTTIME function normalizes this time stamp to the YYYY MM DD HH:mm:ss format and shows it as "2009-01-03 11:06:39" in the event source XML file.

---

**Note:** The EVNTTIME function can be used only if the event has a time stamp value. Before adding an EVNTTIME function, you must define time-related variables either in the header or message definition.

---

SECTORS

## Conditional Variables

- ▶ Methods of manipulating the stored contents of variables
- ▶ Can be advantageous to force a variable to contain a value other than what was directly parsed (to “re-map” the original value)
- ▶ Can use conditional variables to add more data to an event as it is parsed into the database
- ▶ Available in three flavors:
  - Value maps
  - Regular expressions
  - Functions
- ▶ Conditional variables are not required!



## Conditional Variable Syntax

- ▶ Conditional variables need to be invoked from the content parameter of a message tag
- ▶ The usage syntax for all conditional variables in a message definition is:

```
<targetVariableName:*conditionalVariableName  
(sourceVariable or sourceVariableList)>
```
- ▶ Value maps and regular expressions also require special tags `<VALUEMAP />` `<REGX />` to be placed between the `HEADER` and `MESSAGE` tag sections



---

# Conditional Variables

Conditional Variables are methods of manipulating the stored contents of variables.

There are three categories of Conditional Variables:

- Value Maps
- Regular Expressions
- Functions

Conditional Variables are not required in ESI. An XML file can exist and function properly without them.

## Conditional Variable Syntax

Conditional Variables need to be invoked from the content parameter of a message tag.

The usage syntax for all Conditional Variables within message definitions is the following:

```
<targetVariableName:*conditionalVariableName  
(sourceVariable or sourceVariableList)>
```

Value Maps and Regular Expressions will also require a special tag to be placed below the header section and above the message section.

S  
E  
E  
T  
O  
N

## Value Map

- ▶ Sometimes known as table lookup
- ▶ Provides the ability to substitute a DML <parm> value for an alternate value
- ▶ Requires three definitions:
  - Name
  - Defaultvalue
  - Keyvaluepairs



## Value Map Example

```

<VALUEMAP
  name="portconvert"
  default="unknown"
  keyvaluepairs="80='HTTP'|25='SMTP'|110='POP3'" />

<MESSAGE
  .
  .
  .
  content="There is some traffic on port
  &lt;dport&gt; &lt;dportname:*portconvert(dport)&gt; " />
    
```

portconvert	Conditional expression name – it references the Value Map with that same name parameter
dportname	Target variable for the replacement value
dport	Source variable



---

## Conditional Variables, continued

### Value Map

Value maps, sometimes known as table lookups, allow you to substitute a DML value for an alternative value.

Value maps require three definitions:

- **Name** is used to reference the map in the XML event source definitions.
- **Defaultvalue** is the value associated with any key that is not found in the map.
- **Keyvaluepairs** are a set of hash key and associated value pair.

In the example shown in the slide on the left, the source variable, **dport**, value will be passed to the Value Map named **portconvert**.

If the dport value is a 80, 25 or 110, then the target variable, **dportname**, will be populated with the appropriate string from the keyvaluepairs parameter, HTTP, SMTP or POP3.

If the dport value does not match anything in the keyvaluepairs parameter, then dportname will populate with the default string, unknown.

S  
 W  
 T  
 O  
 N

## Regular Expressions

- ▶ Are often called RegX or REGX for shorthand - Note the tag used is REGX
- ▶ Are special ways of manipulating variable contents
- ▶ Can pass results back to an entirely different target variable
- ▶ Should be placed after all header tags in the XML file, below Value Map tags
- ▶ Have the following structure:

```

<REGX
  name="outsideAddr"
  parms="parml"
  default="INSIDE_ADDR"
  expr="&lt;parml&gt;(^192\..*) ? 'OUTSIDE_ADDR' " />
    
```



## REGX Example

```

<REGX
  name="hostnames"
  parms="parml"
  default="NOTME"
  expr="&lt;parml&gt;(^10\.10\.30\.103*) ? 'ME' " />

<MESSAGE
  .
  .
  .
  content="HOST ID &lt;username:*hostnames(laddr)&gt; at addr
&lt;laddr&gt;" />
    
```

username      Target variable for the replacement value

hostnames     Conditional expression name – it references the REGX with that same name parameter

laddr          Source variable





---

## Conditional Variables, continued

### Regular Expressions

A regular expression (REGX) is a compact way of describing complex patterns in texts.

You can use regular expressions to search for patterns and, once found, to modify the patterns. Regular expressions can also be used to launch programmatic actions that depend on patterns.

In the example shown in the slide on the left, the source variable, **laddr**, value will be passed to the REGX tag called hostnames.

If the laddr string starts with the sequence 10.10.30.103, then the target variable, username, will be populated with the string, ME (this is what the '?' operator does in REGX).

If the laddr value does not start with the sequence 10.10.30.103, then username will populate with the default string, NOTME.

S  
E  
T  
O  
N

## Examples of Advanced Topics

The `ciscopixmsg.xml` file:

- ▶ Includes examples of value maps, regular expressions and other advanced functions
- ▶ Is located in:  
`..\etc\devices\ciscopix`



## Removing an Event Source

- ▶ Two reasons for removing an event source:
  - Start from scratch when starting XML development
  - No longer need to support the device
- ▶ Remove the event source from all enVision appliances
  - Managed Monitored Devices
    - Select and Delete
- ▶ To remove the event source:
  - Remove all discovered event sources that use the XML
  - Delete the `..\etc\devices\<eventsourcename>` folder and all of its contents



---

## Additional Information

### Advanced Topics Examples

The enVision directory `..\bin\devices\ciscopix` contains an XML file `ciscopixmsg.xml`. This file includes examples of value maps, regular expressions, and other advanced functions.

### Removing an Event Source

There are two reasons for removing an event source from a system.

- You are working on XML development and want to start from scratch.
- You do not need the event source support anymore.

To remove an event source:

- Remove all discovered event sources that use the XML
- Delete the `..\etc\devices\<eventsourcename>` folder and all of its contents from all enVision appliances

### Reporting a Problem

If you believe you have found a problem with the ESI tool, contact RSA support.

Be sure to include the following information:

- The contents of the `\envision\etc\devices\<devicename>` folder that was created
- The `\envision\etc\ver.dat` file
- The log file from which you are trying to develop (sample data is required to duplicate an issue)
- A clear description of the problem

S  
W  
T  
O  
N

## RSA enVision Event Source Updates

- ▶ Monthly content update
- ▶ Run on the A-SRV – changes will be replicated to all nodes

Attribute Name	Preserved	Not Preserved
level	X	
parse	X	
parsedefvalue		X
tableid		X
id1		X
id2		X
sitetrack	X	
eventcategory	X	
summary		X
content*		X

\* The content attribute consists of the detailed parsing logic for a given event. The logic is a combination of static text, variables, and UDS (Universal Device Support) functions.



The Security Division of EMC 128

# Event Source Updates

## How the enVision Event Source Update Works

You only run the enVision Event Source executable from the A-SRV running the NIC App server. The executable replicates the update to all nodes in your enVision domain, then it executes the update on each node.

The enVision Event Source Update executable monitors the execution of the update on the appliance it is running on. You must still use the enVision Event Viewer to monitor the update progress on all other appliances in the site and the enVision domain.

### How the Update Applies Each Patch

Each patch in the update package migrates the event source XML for an event source cited in the package.

It merges customer (your existing version of the msg.xml file) and factory (the new enVision version of the msg.xml file) data. In addition, the update creates a msg.xml.bkup file that is identical to the newly merged msg.xml. Finally, the update creates an msg.xml.mod file that contains the customer changes and tells you whether or not they are preserved in the newly merged msg.xml file.

The table lists the attributes the update preserves from the customer changes.

### Enhancement Requests

To request a new event source, refer to the Enhancement Portal at the following URL:

[http://www.rsa.com/go/partners/suggest\\_new.asp](http://www.rsa.com/go/partners/suggest_new.asp)

# SESSION

## Exercise 5 Add Advanced Functions (Optional)

### The goal of this exercise is:

- Add the CALC function to total the number of bytes
- Assign the Foreign port name using a value map

### Add the CALC Function

Follow these steps to create a message and add a function that combines bytes sent and bytes received to populate total bytes.

Step	Action
1	Select a Terminated message from the Tree View and select Edit.
2	Move to the Function screen.
3	In the Function name: field, select <b>CALC</b>
4	In the Expression field, enter: <b>sbytes+rbytes</b>
5	Click <b>Next</b> .
6	Click <b>Finish</b> .
7	Select the other Terminated message and repeat steps 1 through 6.
8	Restart the NIC Server service.
9	Run another query/report. The bytes field should be populated. It should equal the sum of bytes in and bytes out.

Note: If you cannot edit a Terminated message, create a new message or create a new XML file and add the parsed and you cannot select a message, create a new XMLfile with the laddr, lport, faddr, fport, username, sbytes and rbytes variables.

## Exercise 5, continued

### Add a Value Map

The FireWall Accounting table contains a field called Foreign Port Name. This field is unpopulated, whereas the Foreign Port field is populated. You can assign the Port Name using a value map.

Value Maps are not supporting through the ESI interface. However, we can edit the XML code using the interface Code View.

Follow these steps to create a value map for the BulldogDogHousemsg.xml file.

Step	Action
1	Look at the variables in the FireWall Accounting table either using the Help or the Appendix.  Note that the table contains a variable for fportname.
2	Run a query on the BulldogDogHouse.  Note that the fportname field is empty.
3	To populate the foreign port name, click the <b>Code View</b> tab in the ESI tool.
4	AFTER the Header tag (and before the Message tag), add a Valuemap tag as follows:  <b>&lt;VALUEMAP name="portconvert" default="unknown" keyvaluepairs="80='HTTP' 443='HTTPS'"/&gt;</b>
5	Add the following at the end of the Request, Established and Terminated messages (insert before the final "  <b>&amp;lt;@fportname:*portconvert(fport)&amp;gt;"/&gt;</b>
6	From the File menu, select <b>Save</b> to save the code.
7	From the Action menu, select <b>Validate XML</b> .  Check the Code View for errors. If there are errors, correct them according to Steps 4 and 5.
8	Restart the NIC Server service.
9	Run a report/query and check that the port name is assigned.



## Exercise 5, continued

The start of the file should look like this:

```
<HEADER
  id1="0001"
  id2="0001"
content="Socks5[&lt;pid&gt;]: TCP Connection
&lt;messageid&gt;: &lt;!payload&gt;" />
<VALUEMAP
  name="portconvert"
  default="unknown"
  keyvaluepairs="80='HTTP'|443='HTTPS'"/>
<MESSAGE
  level="7"
  parse="1"
  parsedefvalue="1"
  tableid="12"
  id1="Request"
  id2="Request"
  eventcategory="1801000000"
content="Connect (&lt;laddr&gt;:&lt;lport&gt; to
&lt;faddr&gt;:&lt;fport&gt;) for user
&lt;username&gt;&lt;@fportname:*portconvert (fport
)&gt;" />
```

# SESSION

---

# enVision Taxonomy

---

# SWEET TATION

## RSA enVision Event Taxonomy

XX00000000	NIC Category				
XXXX000000	Alert Category				
XXXXXX00000	Event Category (3-item )				
XXXXXXXXX00	Event Category (4-item )				
XXXXXXXXXXXX	Event Category (5-item )				
1000000000	Attacks				
1001000000	Attacks.	Access			
1001030000	Attacks.	Access.	Informational		
1001030100	Attacks.	Access.	Informational.	Eavesdropping	
1001030200	Attacks.	Access.	Informational.	Host Based	
1001030201	Attacks.	Access.	Informational.	Host Based.	ODBC
1001030202	Attacks.	Access.	Informational.	Host Based.	SQL
1001030203	Attacks.	Access.	Informational.	Host Based.	TFTP
1001030300	Attacks.	Access.	Informational.	Network Based	
1001030305	Attacks.	Access.	Informational.	Network Based.	HTTP
1001030301	Attacks.	Access.	Informational.	Network Based.	NetBIOS
1001030302	Attacks.	Access.	Informational.	Network Based.	NFS
1001030303	Attacks.	Access.	Informational.	Network Based.	NNTP
1001030304	Attacks.	Access.	Informational.	Network Based.	TELNET
1001030400	Attacks.	Access.	Informational.	Possibly benign	
1001030500	Attacks.	Access.	Informational.	TCP/IP	
1001010000	Attacks.	Access.	Interception		
1001020000	Attacks.	Access.	Modification		
1001020200	Attacks.	Access.	Modification.	Host Based	
1001020205	Attacks.	Access.	Modification.	Host Based.	FTP
1001020201	Attacks.	Access.	Modification.	Host Based.	ODBC
1001020206	Attacks.	Access.	Modification.	Host Based.	Overflow
1001020204	Attacks.	Access.	Modification.	Host Based.	RPC
1001020202	Attacks.	Access.	Modification.	Host Based.	SQL
1001020203	Attacks.	Access.	Modification.	Host Based.	TFTP
1001020300	Attacks.	Access.	Modification.	Network Based	

## RSA enVision Event Taxonomy

1001020305	Attacks.	Access.	Modification.	Network Based.	HTTP
1001020309	Attacks.	Access.	Modification.	Network Based.	ICMP
1001020301	Attacks.	Access.	Modification.	Network Based.	NetBIOS
1001020302	Attacks.	Access.	Modification.	Network Based.	NFS
1001020303	Attacks.	Access.	Modification.	Network Based.	NNTP
1001020307	Attacks.	Access.	Modification.	Network Based.	POP3
1001020306	Attacks.	Access.	Modification.	Network Based.	SMTP
1001020308	Attacks.	Access.	Modification.	Network Based.	SNMP
1001020304	Attacks.	Access.	Modification.	Network Based.	TELNET
1001020100	Attacks.	Access.	Modification.	TCP/IP	
1002000000	Attacks.	Denial of Service			
1002010000	Attacks.	Denial of Service.	Bandwidth consumption		
1002010100	Attacks.	Denial of Service.	Bandwidth consumption.	Distributed	
1002010200	Attacks.	Denial of Service.	Bandwidth consumption.	Non-Distributed	
1002006000	Attacks.	Denial of Service.	DNS		
1002002000	Attacks.	Denial of Service.	Generic attacks		
1002030000	Attacks.	Denial of Service.	Program flaws		
1002040000	Attacks.	Denial of Service.	Resource stravation		
1002050000	Attacks.	Denial of Service.	Routing		
1003000000	Attacks.	Malicious Code			
1003040000	Attacks.	Malicious Code.	P2P		
1003050000	Attacks.	Malicious Code.	Spyware		
1003030000	Attacks.	Malicious Code.	Trojan Horse/Backdoor		
1003010000	Attacks.	Malicious Code.	Virus		
1003010500	Attacks.	Malicious Code.	Virus.	Armored	
1003010900	Attacks.	Malicious Code.	Virus.	Camouflage	
1003010700	Attacks.	Malicious Code.	Virus.	Cavity (Spacefiller)	
1003010300	Attacks.	Malicious Code.	Virus.	Fast and Slow Infectors	

### RSA enVision Event Taxonomy

1003010600	Attacks.	Malicious Code.	Virus.	Multipartite
1003011000	Attacks.	Malicious Code.	Virus.	NTFS ADS
1003010100	Attacks.	Malicious Code.	Virus.	Polymorphic
1003010400	Attacks.	Malicious Code.	Virus.	Sparse Infectors
1003010200	Attacks.	Malicious Code.	Virus.	Stealth
1003010800	Attacks.	Malicious Code.	Virus.	Tunneling
1003020000	Attacks.	Malicious Code.	Worm	
1300000000	Auth			
1303000000	Auth.	Errors		
1301000000	Auth.	Failures		
1301000000	Auth.	Failures.	Administrative Settings	
1301000000	Auth.	Failures.	User Errors	
1302000000	Auth.	Successful		
1302010000	Auth.	Successful.	Methods	
1302010100	Auth.	Successful.	Methods.	RADIUS
1302010200	Auth.	Successful.	Methods.	SSH
1302010300	Auth.	Successful.	Methods.	TACACS
1302010400	Auth.	Successful.	Methods.	Windows
1700000000	Config			
1701000000	Config.	Changes		
1701010000	Config.	Changes.	Add	
1701030000	Config.	Changes.	Delete	
1701070000	Config.	Changes.	Disable	
1701060000	Config.	Changes.	Enable	
1701050000	Config.	Changes.	Export	
1701040000	Config.	Changes.	Import	
1701020000	Config.	Changes.	Modify	
1701080000	Config.	Changes.	Move	

### RSA enVision Event Taxonomy

1703000000	Config.	Errors			
1703010000	Config.	Errors.	Auditing		
1703020000	Config.	Errors.	Uploading		
1705000000	Config.	OS			
1705010000	Config.	OS.	Service Pack		
1705010200	Config.	OS.	Service Pack.	Installed	
1705010300	Config.	OS.	Service Pack.	Uninstalled	
1704000000	Config.	Software			
1704010000	Config.	Software.	Installed		
1704030000	Config.	Software.	Modified		
1704020000	Config.	Software.	Uninstalled		
1702000000	Config.	Versions			
1702020000	Config.	Versions.	Deleted Versions		
1702030000	Config.	Versions.	Inconsistent States		
1702010000	Config.	Versions.	New Version		
1200000000	Content				
1203000000	Content.	Bad Data			
1207000000	Content.	Email			
1207010000	Content.	Email.	Delivery		
1207010200	Content.	Email.	Delivery.	Error	
1207010201	Content.	Email.	Delivery.	Error.	Nondelivery Receipt
1207010100	Content.	Email.	Delivery.	Success	
1207030000	Content.	Email.	Message Received		
1207030100	Content.	Email.	Message Received.	Attachment	
1207020000	Content.	Email.	Message Sent		
1207020100	Content.	Email.	Message Sent.	Attachment	
1207040000	Content.	Email.	Spam		
1207040200	Content.	Email.	Spam.	Blocked	



### RSA enVision Event Taxonomy

1207040100	Content.	Email.	Spam.	Suspect
1201000000	Content.	Illegal URL		
1201010000	Content.	Illegal URL.	Porn	
1206000000	Content.	Object		
1206020000	Content.	Object.	Delete	
1206010000	Content.	Object.	Open	
1205000000	Content.	Process		
1205010000	Content.	Process.	Create	
1205020000	Content.	Process.	Exit	
1202000000	Content.	SiteTrack		
1204000000	Content.	Web Traffic		
1204020000	Content.	Web Traffic.	Denied	
1204020400	Content.	Web Traffic.	Denied.	Applications
1204020200	Content.	Web Traffic.	Denied.	Audio/Video
1204020300	Content.	Web Traffic.	Denied.	Images
1204020100	Content.	Web Traffic.	Denied.	Text
1204010000	Content.	Web Traffic.	Successful	
1204010400	Content.	Web Traffic.	Successful.	Applications
1204010200	Content.	Web Traffic.	Successful.	Audio/Video
1204010300	Content.	Web Traffic.	Successful.	Images
1204010100	Content.	Web Traffic.	Successful.	Text
1800000000	Network			
1801000000	Network.	Connections		
1801010000	Network.	Connections.	Errors	
1801010100	Network.	Connections.	Errors.	VPN
1801020000	Network.	Connections.	Successful	
1801020100	Network.	Connections.	Successful.	VPN
1801030000	Network.	Connections.	Terminations	

### RSA enVision Event Taxonomy

1801030100	Network.	Connections.	Terminations.	VPN
1803000000	Network.	Denied Connections		
1803010000	Network.	Denied Connections.	Address	
1803020000	Network.	Denied Connections.	Protocol	
1803030000	Network.	Denied Connections.	Username	
1804000000	Network.	Devices		
1804010000	Network.	Devices.	Additions	
1804020000	Network.	Devices.	Removals	
1805000000	Network.	Routing		
1805020000	Network.	Routing.	Changes	
1805010000	Network.	Routing.	Errors	
1805010100	Network.	Routing.	Errors.	Collisions
1802000000	Network.	Usage		
1901000000	Other.	Default		
1500000000	Policies			
1501000000	Policies.	ACL		
1501010000	Policies.	ACL.	Accepted	
1501020000	Policies.	ACL.	Added	
1501030000	Policies.	ACL.	Deleted	
1501040000	Policies.	ACL.	Denied	
1501040100	Policies.	ACL.	Denied.	Groups
1501050000	Policies.	ACL.	Errors	
1501050100	Policies.	ACL.	Errors.	Config
1503000000	Policies.	Rights		
1503010100	Policies.	Rights.	Failed.	Privileged Use
1503020200	Policies.	Rights.	Successful.	Privileged Use
1502000000	Policies.	Rules		
1502030000	Policies.	Rules.	Added	

### RSA enVision Event Taxonomy

1502040000	Policies.	Rules.	Deleted	
1502050000	Policies.	Rules.	Modofied	
1502010000	Policies.	Rules.	Rejects	
1502020000	Policies.	Rules.	Successful	
1100000000	Recon			
1101000000	Recon.	Brute Force		
1101010000	Recon.	Brute Force.	Account Related	
1101020000	Recon.	Brute Force.	Authorization	
1102000000	Recon.	DNS		
1103000000	Recon.	Scans		
1103010000	Recon.	Scans.	ARP	
1103020000	Recon.	Scans.	Enumeration	
1103030000	Recon.	Scans.	Finger	
1104000000	Recon.	Sweeps		
1600000000	System			
1602000000	System.	Accounting		
1602010000	System.	Accounting.	Errors	
1602010100	System.	Accounting.	Errors.	Hardware
1602020000	System.	Accounting.	Successful	
1609000000	System.	Alerts		
1612000000	System.	Audit		
1612010000	System.	Audit.	Failure	
1613030000	System.	Crypto.	Configuration	
1613030100	System.	Crypto.	Configuration.	Error
1613060200	System.	Crypto.	Decryption.	Failure
1613060100	System.	Crypto.	Decryption.	Success
1613020000	System.	Crypto.	Disabled	
1613010000	System.	Crypto.	Enabled	

### RSA enVision Event Taxonomy

1613050200	System.	Crypto.	Encryption.	Failure	
1613050201	System.	Crypto.	Encryption.	Failure.	Invalid Method
1613050100	System.	Crypto.	Encryption.	Success	
1613040100	System.	Crypto.	Key.	Created	
1613040200	System.	Crypto.	Key.	Manipulation	
1603000000	System.	Errors			
1603050000	System.	Errors.	Command Failures		
1603060000	System.	Errors.	Config		
1603080000	System.	Errors.	CPU		
1603100000	System.	Errors.	Environmentals		
1603010000	System.	Errors.	Hardware		
1603010100	System.	Errors.	Hardware.	Disk	
1603030000	System.	Errors.	Interfaces		
1603020000	System.	Errors.	Memory		
1603070000	System.	Errors.	Peripherals		
1603090000	System.	Errors.	Resources		
1603110000	System.	Errors.	Services		
1603040000	System.	Errors.	Software		
1601000000	System.	Failures			
1601010000	System.	Failures.	Hardware		
1601020000	System.	Failures.	Software		
1604000000	System.	Heartbeats			
1604010000	System.	Heartbeats.	Errors		
1608000000	System.	License			
1608010000	System.	License.	Violations		
1605000000	System.	Normal Conditions			
1605030000	System.	Normal Conditions.	Config		
1605010000	System.	Normal Conditions.	Daemons		

### RSA enVision Event Taxonomy

1605020000	System.	Normal Conditions.	Services	
1606000000	System.	Reboots		
1611000000	System.	Shutdown		
1610000000	System.	Startup		
1607000000	System.	Unusual Activity		
1614000000	System.	VA Scan		
1614020000	System.	VA Scan.	Completed	
1614030000	System.	VA Scan.	Error	
1614010000	System.	VA Scan.	Started	
1400000000	User			
1401000000	User.	Activity		
1401080000	User.	Activity.	Export	
1401030000	User.	Activity.	Failed Logins	
1401010000	User.	Activity.	File Access	
1401020000	User.	Activity.	Known Bad Commands	
1401060100	User.	Activity.	Login.	Workstation Unlock
1401070000	User.	Activity.	Logoff	
1401070100	User.	Activity.	Logoff.	Workstation Lock
1401040000	User.	Activity.	Normal Activity	
1401050200	User.	Activity.	Privileged Use.	Denied
1401050100	User.	Activity.	Privileged Use.	Successful
1401010000	User.	Activity.	Successful Logins	
1401080000	User.	Activity.	View	
1402000000	User.	Management		
1402010200	User.	Management.	Groups.	Additions
1402010100	User.	Management.	Groups.	Deletions
1402010300	User.	Management.	Groups.	Modifications
1402010301	User.	Management.	Groups.	Modifications. User Added

### RSA enVision Event Taxonomy

1402010302	User.	Management.	Groups.	Modifications.	User Removed
1402040200	User.	Management.	Password.	Expiration	
1402040100	User.	Management.	Password.	Modifications	
1402040101	User.	Management.	Password.	Modifications.	Failed
1402030000	User.	Management.	Permissions		
1402020200	User.	Management.	Users.	Additions	
1402020100	User.	Management.	Users.	Deletions	
1402020400	User.	Management.	Users.	Disabled	
1402020300	User.	Management.	Users.	Modifications	

---

# Sample Database Tables

---

# SESSION



Unix Security Database Table (.. etc\sqltbl\Unix Security)

Displayed Database Table Name	Internal Table Name	Table ID	Table Type	Table Cat	Internal Field Name	Displayed Field (Column) Name	FieldType	Field Len			
Unix Security	unixsecurity	50	TRUE	detail	device						
					paddr	DeviceAddress	CHAR	20	RESERVED	RESERVED	NULL
					devicehostname	DeviceHostName	CHAR	128	RESERVED	RESERVED	NULL
					stamp	Date/Time	TRUE	TIMESTAMP	24	RESERVED	RESERVED
					msg_id	MessageID	TRUE	CHAR	64	RESERVED	RESERVED
					faddr	ForeignAddress	TRUE	CHAR	20	faddr	ADDR1
					fhost	ForeignHostName	TRUE	CHAR	96	fhost	HOST1
					fport	ForeignPort	TRUE	INT	0	fport	PORT1
					fportname	ForeignPortName	TRUE	CHAR	48	fportname	NAME1
					username	UserName	TRUE	CHAR	48	username	CHAR48_1
					tbdstr1	UserName1	TRUE	CHAR	48	username1	CHAR48_2
					uid	Uid	TRUE	INT	0	uid	INT1
					type	Type	TRUE	INT	0	ntype	TYPE
					protocol	Protocol	TRUE	CHAR	64	protocol	CHAR64_1
					int_name	Interface	TRUE	CHAR	48	interface	CHAR48_3
					action	Action	TRUE	CHAR	64	action	CHAR64_2
					agent	Agent	TRUE	CHAR	64	agent	CHAR64_3
					reason	Reason	TRUE	CHAR	64	reason	CHAR64_4
					info	Info	TRUE	CHAR	256	info	CHAR256_1
					directory	WorkingDirectory	TRUE	CHAR	64	directory	CHAR64_5

Firewall Accounting Database Table (.. etc\sqltbl\Firewall Accounting)

Displayed Database Table Name	Internal Table Name	Table ID	Table Type	Table Cat	Internal Field Name	Displayed Field (Column) Name	FieldType	Field Len		
FireWall Accounting	accounting	12	TRUE	detail	device					
					paddr	DeviceAddress	TRUE	CHAR	20	RESERVED
					devicehostname	DeviceHostName	TRUE	CHAR	128	RESERVED
					stamp	Date/Time	TRUE	TIMESTAMP	24	RESERVED
					msg_id	MessageID	TRUE	CHAR	64	RESERVED
					laddr	LocalAddress	TRUE	CHAR	20	RESERVED
					lhost	LocalHostName	TRUE	CHAR	96	HOST2
					lport	LocalPort	TRUE	INT	0	PORT2
					lportname	LocalPortName	TRUE	CHAR	48	lportname
					gaddr	TranslateAddress	TRUE	CHAR	20	gaddr
					faddr	ForeignAddress	TRUE	CHAR	20	faddr
					fhost	ForeignHostName	TRUE	CHAR	96	fhost
					fport	ForeignPort	TRUE	INT	0	fport
					fportname	ForeignPortName	TRUE	CHAR	48	fportname
					logip	SyslogIp	TRUE	CHAR	24	logip
					origin	SourceOrigin	TRUE	CHAR	48	origin
					haddr	HostAddress	TRUE	CHAR	24	haddr
					host	HostName	TRUE	CHAR	64	host
					username	UserName	TRUE	CHAR	48	username
					bytes	Bytes	TRUE	BIGINT	0	bytes
					duration	Duration	TRUE	INT	0	duration
					accountid	AccountID	TRUE	INT	0	accountid
					type	Connect/Translate	TRUE	INT	0	nType
					in_out	Inbound/Outbound	TRUE	INT	0	inout
					src_zone	SourceZone	TRUE	CHAR	32	src_zone
					dst_zone	DestZone	TRUE	CHAR	32	dst_zone
					event_time	EventTime	TRUE	TIMESTAMP	24	event_time
					iassoc1	Department1	TRUE	CHAR	64	RESERVED
					iassoc2	Department2	TRUE	CHAR	64	RESERVED
					iassoc3	Department3	TRUE	CHAR	64	RESERVED
					eassoc1	Category1	TRUE	CHAR	64	RESERVED
					eassoc2	Category2	TRUE	CHAR	64	RESERVED
					eassoc3	Category3	TRUE	CHAR	64	RESERVED
					sent_bytes	SentBytes	TRUE	BIGINT	0	sbytes
					recv_bytes	ReceivedBytes	TRUE	BIGINT	0	rbytes
					policy_id	PolicyID	TRUE	CHAR	48	policy_id
					action	Action	TRUE	CHAR	64	action
					rule	Rule	TRUE	CHAR	64	rule
					protocol	Protocol	TRUE	CHAR	64	protocol
					service	Service	TRUE	CHAR	64	service
					int_name	Interface	TRUE	CHAR	48	interface
					tbdstr1	ConnectionID	TRUE	CHAR	64	misc_id
					tbdstr3	VirtualName	TRUE	CHAR	64	vsys
					finterface	ForeignInterface	TRUE	CHAR	32	finterface
					linterface	LocalInterface	TRUE	CHAR	32	linterface
					stransaddr	TranslatedSourceAddress	TRUE	CHAR	128	stransaddr
					stransport	TranslatedSourcePort	TRUE	CHAR	16	stransport
					dtransaddr	TranslatedDestAddress	TRUE	CHAR	128	dtransaddr
					dtransport	TranslatedDestPort	TRUE	CHAR	16	dtransport
					category	EventCategory	TRUE	CHAR	32	RESERVED

Access Control Database Table (.. etc\sqltbl1\Access Control)

Displayed Database Table Name	Internal Table Name	Table ID	Table Type	Table Cat	Internal Field Name	Displayed Field (Column) Name	FieldType	Field Len			
Access Control	accesscontrol	11	TRUE	detail	device						
	paddr				DeviceAddress	DeviceAddress	CHAR	20	RESERVED	RESERVED	NULL
	devicehostname				DeviceHostName	DeviceHostName	CHAR	128	RESERVED	RESERVED	NULL
	stamp				Date/Time	Date/Time	TRUE	TIMESTAMP	24	RESERVED	RESERVED
	msg_id				MessageID	MessageID	TRUE	CHAR	64	RESERVED	RESERVED
	severity				Severity	Severity	TRUE	CHAR	64	RESERVED	RESERVED
	event_source				EventSource	EventSource	TRUE	CHAR	64	event_source	CHAR64_11
	hostname				HostName	HostName	TRUE	CHAR	96	hostname	CHAR96_1
	hostip				IPAddress	IPAddress	TRUE	CHAR	64	hostip	CHAR64_14
	macaddr				MacAddress	MacAddress	TRUE	CHAR	24	macaddr	CHAR24_8
	node				NodeName	NodeName	TRUE	CHAR	64	node	CHAR64_10
	clustermembers				ClusterMembers	ClusterMembers	TRUE	CHAR	128	cluster_members	CHAR128_8
	saddr				SourceAddress	SourceAddress	TRUE	CHAR	32	saddr	ADDR1
	shost				SourceHostName	SourceHostName	TRUE	CHAR	96	shost	HOST1
	smacaddr				SourceMacAddress	SourceMacAddress	TRUE	CHAR	32	smacaddr	CHAR32_4
	daddr				DestinationAddress	DestinationAddress	TRUE	CHAR	32	daddr	ADDR2
	dhost				DestinationHostName	DestinationHostName	TRUE	CHAR	96	dhost	HOST2
	dmacaddr				DestinationMacAddress	DestinationMacAddress	TRUE	CHAR	32	dmacaddr	CHAR32_5
	userid				UserID	UserID	TRUE	CHAR	48	username	CHAR48_1
	realm				Realm	Realm	TRUE	CHAR	64	realm	CHAR64_9
	authmethod				AuthenticationMethod	AuthenticationMethod	TRUE	CHAR	16	authmethod	CHAR16_1
	c_username				ClientUserName	ClientUserName	TRUE	CHAR	128	c_username	CHAR128_6
	errcode				ErrorCode	ErrorCode	TRUE	INT	0	errcode	INT1
	sessionid				SessionID	SessionID	TRUE	CHAR	60	event_type	CHAR64_1
	session				Session	Session	TRUE	CHAR	64	sessionid	CHAR64_5
	deviceid				DeviceID	DeviceID	TRUE	CHAR	48	device	CHAR48_2
	accountid				AccountID	AccountID	TRUE	CHAR	48	connection_id/connection	CHAR48_3
	failreason				Fail Reason	Fail Reason	TRUE	CHAR	128	reason	CHAR128_1
	discreason				Disconnect Reason	Disconnect Reason	TRUE	CHAR	128	discreason	CHAR128_2
	disposition				Disposition	Disposition	TRUE	CHAR	64	disposition	CHAR64_12
	policyname				PolicyName	PolicyName	TRUE	CHAR	128	policyname	CHAR128_7
	policy_value				PolicyValue	PolicyValue	TRUE	CHAR	256	policy_value	CHAR256_3
	poolid				PoolID	PoolID	TRUE	CHAR	48	poolid	CHAR48_4
	portno				PortNO	PortNO	TRUE	CHAR	48	port	CHAR48_5
	socketno				SocketNO	SocketNO	TRUE	CHAR	48	socketno	CHAR48_6
	success				Success/Denial/Failure	Success/Denial/Failure	TRUE	CHAR	32	success	CHAR32_1
	groupid				Group	Group	TRUE	CHAR	48	group	CHAR64_2
	category				Category	Category	TRUE	CHAR	256	category	CHAR256_2
	user_org				UserOrg	UserOrg	TRUE	CHAR	64	user_org	CHAR64_6
	user_fullname				UserFullName	UserFullName	TRUE	CHAR	128	user_fullname	CHAR128_4
	event_time				EventTime	EventTime	TRUE	TIMESTAMP	24	event_time	EVENTTIME
	event_queue_time				EventQueueTime	EventQueueTime	TRUE	TIMESTAMP	24	event_queue_time	CHAR24_5
	expiration_time				ExpirationTime	ExpirationTime	TRUE	TIMESTAMP	24	expiration_time	CHAR24_7
	recorded_time				RecordedTime	RecordedTime	TRUE	TIMESTAMP	24	recorded_time	CHAR24_6
	netdevicegroup				NetworkDeviceGroup	NetworkDeviceGroup	TRUE	CHAR	64	group1	CHAR64_3
	app_token				ApplicationPostureToken	ApplicationPostureToken	TRUE	CHAR	128	status/result	CHAR128_3
	result_code				ResultCode	ResultCode	TRUE	INT	0	resultcode	INT3
	system_token				SystemPostureToken	SystemPostureToken	TRUE	CHAR	24	state	CHAR24_1
	objectname				ObjectName	ObjectName	TRUE	CHAR	32	name	CHAR32_2

Access Control Database Table (.. etc\sqltbl1\Access Control)

version	Version	TRUE	CHAR	64	version	CHAR64_13	NULL
action	Action	TRUE	CHAR	64	action	CHAR64_4	NULL
privilege	Privileges	TRUE	CHAR	64	privilege	CHAR64_7	NULL
risk_num	Risk	TRUE	INT	0	risk_num	INT2	NULL
process_id	ProcessID	TRUE	INT	0	process_id	INT4	NULL
process	Process	TRUE	CHAR	64	process	CHAR64_15	NULL
param1	Parameter1	TRUE	CHAR	24	param1	CHAR24_2	NULL
param2	Parameter2	TRUE	CHAR	24	param2	CHAR24_3	NULL
param3	Parameter3	TRUE	CHAR	24	param3	CHAR24_4	NULL
param4	Parameter4	TRUE	CHAR	24	param4	CHAR32_3	NULL
info	Information	TRUE	CHAR	128	Info	CHAR128_5	NULL
event_description	EventDescription	TRUE	CHAR	256	event_description	CHAR256_1	NULL
eventcatname	EventCategoryName	TRUE	CHAR	128	RESERVED	RESERVED	NULL
msg	Message	TRUE	CHAR	512	msg	MSG	NULL