Sentinel™ 5

Reference Guide v5.1.1

- Linux
- Solaris
- Windows

Volume IV of V

March 2006
www.esecurity.net

e-SECURITY

Preface

The e-Security Technical documentation is general-purpose operation and reference guide. This documentation is intended for Information Security Professionals. The text in this documentation is designed to serve as a source of reference about e-Security's Enterprise Security Management System. There is additional documentation available on the e-Security web portal.

e-Security Technical documentation is broken down into five different volumes. They are:

- Volume I – Sentinel™ 5 Install Guide
- Volume II – Sentinel™ 5 User's Guide
- Volume III – Sentinel™ 5 Wizard User's Guide
- Volume IV – Sentinel™ User's Reference Guide
- Volume V – Sentinel™ 3rd Party Integration

Volume I – Sentinel Install Guide

This guide explains how to install:

- Sentinel Server
- Sentinel Console
- Sentinel Correlation Engine
- Sentinel Crystal Reports
- Wizard Agent Builder
- Wizard Agent Manager
- Advisor

Volume II – Sentinel User's Guide

This guide discusses:

- Sentinel Console Operation
- Sentinel Features
- Sentinel Architecture
- Sentinel Communication
- Shutdown/Startup of Sentinel
- Vulnerability assessment
- Event monitoring
- Event filtering
- Event correlation
- Sentinel Data Manager
- Event Configuration for Business Relevance
- Mapping Service
- Historical reporting
- Wizard Host Management
- Incidents
- Cases
- User management
- Workflow

Volume III – Wizard User's Guide

This guide discusses:

- Wizard Agent Builder Operation
- Wizard Agent Manager
- Agents
- Wizard Host Management
- Building and maintaining agents

Volume IV - Sentinel User's Reference Guide

This guide discusses:

- Wizard scripting language
- Wizard parsing commands
- Wizard administrator functions
- Wizard and Sentinel meta-tags
- User Permissions
- Sentinel correlation engine
- Correlation command line options
- e-Security database schema

Volume V - Sentinel 3rd Party Integration Guide

- Remedy
- HP OpenView Operations
- HP Service Desk

Chapter 1 – Sentinel™ 5 User Reference Introduction................................1-1
Contents..1-1
Conventions Used..1-1
Notes and Cautions..1-1
Commands...1-1
Other e-Security References...1-1
Contacting e-Security..1-2
Chapter 2 – Wizard Scripting Language 2-1
Decide Strings...2-1
Manipulating the Rx Buffer (Receive Buffer) Pointer...................................2-1
Format...2-1
Parameter Names..2-2
Hierarchy of Operations in a Decide String..2-2
Receive Buffer Pointer Rules...2-2
Checking for an Empty Receive Buffer..2-3
Decide String Evaluations and Results Example..2-3
Regular Expressions...2-3
Summary of Special Characters for Regular Expressions............................2-4
White space in Regular Expressions..2-4
Parsing Commands...2-5
Simple Data Types...2-5
Derived Aggregate Data Types...2-6
Special Rules for Variables..2-7
Chapter 3 – Wizard Parsing Commands 3-1
Command Format and Using Arrays..3-3
Commands..3-3
ALERT...3-3
APPEND 3-4
BITFIELD 3-7
BREAKPOINT 3-9
BYTEFIELD...3-9
CLEAR 3-11
CLEARTAGS 3-12
COMMENT...3-13
COMPARE 3-14
CONSTANTTAGS...3-15
CONVERT...3-16
COPY 3-17
CRC 3-19
DATE..3-20
DATETIME 3-21
DBCLOSE...3-22
DBDELETE 3-22
DBGETROW..3-23
DBINSERT 3-23
DBOPEN...3-24
DBSELECT 3-25
DEC..3-26
DECODE...3-27
DELETE 3-27

DISPLAY 3-28
ELSE 3-29
ENCODE 3-30
ENDFOR 3-30
ENDIF 3-31
ENDWHILE 3-31
ESNMPGET 3-31
ESNMPSET 3-32
EVENT 3-33
EXPORTVAR 3-36
FILEA 3-37
FILEL 3-38
FILER 3-38
FILEW 3-39
FOR 3-40
GETCONFIG 3-41
GETENV 3-42
HEXTONUM 3-42
IF 3-43
INC 3-45
INDICATOR 3-45
INFO_CLEARTAGS 3-46
INFO_CLOSE 3-46
INFO_CONSTANTTAGS 3-47
INFO_CREATE 3-47
INFO_DUMP 3-48
INFO_PUSH 3-48
INFO_SEND 3-48
INFO_SETTAG 3-49
INFO_* Command Example 3-51
IPTONUM 3-53
LENGTH or LENGTH-OPTION2 3-54
LOOKUP 3-54
NEGSEARCH 3-56
NUMTOHEX 3-57
NUMTOIP 3-57
PARSER_ATTACHVARIABLE 3-58
PARSER_CREATEBASIC 3-59
PARSER_NEXT 3-60
PARSER_PARSESTRING 3-61
PAUSE 3-61
POPUP 3-62
PRINTF 3-62
REGEXPREPLACE 3-64
REGEXPSEARCH, REGEXPSEARCH_EXPLICIT or REGEXPSEARCH_STRING 3-66
REPLACE 3-69
RESET 3-70
RXBUFF 3-70
SEARCH 3-71
SET 3-72

    SETBYTES 3-73
    SETCONFIG 3-73
    SHELL 3-74
    SKIP 3-75
    SKIPWORD 3-77
    SNMPGET 3-78
    SNMPSET 3-79
    SOCKETW 3-80
    STONUM 3-81
    STRIP or STRIP-ASCII-RANGE 3-82
    TBOSSETCOMMAND 3-83
    TBOSSETREQUEST 3-85
    TIME 3-87
    TOKENIZE 3-87
    TOLOWER 3-89
    TOUPPER 3-89
    TRANSLATE 3-90
    TRIM 3-92
    WHILE 3-92

Chapter 4 – Wizard Administrator Functions 4-1

  Wizard Utilities and Applications 4-1
    Agent Builder 4-1
    Agent Manager 4-1
    Agent Engine 4-1
    popup.exe 4-1
    popup.cfg 4-2
  Wizard Directory Structure 4-2

Chapter 5 – Wizard and Sentinel Meta-tags 5-1

Chapter 6 – Sentinel Control Center User Permissions 6-1

  Default Users 6-1
  General 6-1
    General – Public Filters 6-2
    General – Private Filters 6-2
    General – Integration Actions 6-2
  Active Views 6-2
    Active Views – Menu Items 6-2
    Active Views – Summary Displays 6-3
  iTRAC 6-4
    Template Management 6-4
    Process Management 6-4
  Incidents 6-4
  Agent Management 6-4
  Analysis 6-5
  Advisor 6-5
  Administration 6-5
    Administration – Correlation 6-5
    Administration – Global Filters 6-6
    Administration – Menu Configuration 6-6
    Administration - DAS Statistics 6-6
    Administration – Event File Information 6-6

Administration – Server Views (Linux only) 6-6
Administration – User Management 6-6
Administration – User Session Management 6-7
Administration – iTRAC Role Management 6-7
Chapter 7 – Sentinel Correlation Engine 7-1
Correlation Filter Types 7-2
Pattern Type Correlation Filter 7-2
Filter Manager Type Correlation Filter 7-2
Builder Type Correlation Filter 7-3
Correlation Rule Definition 7-5
Watchlist 7-5
Basic Correlation 7-5
Advanced Correlation 7-5
Free Form RuleLg Correlation 7-6
Creating a Watchlist Rule 7-6
Creating a Basic Correlation Rule 7-9
Creating an Advanced Correlation Rule 7-13
Creating a Free Form RuleLg Correlation Rule 7-17
Filter Operation 7-18
Window Operation 7-19
Trigger Operation 7-21
Operators That Combine With Operations to Form Rules 7-22
Example Correlation Rules 7-23
Buffer Overflow Attack and Service Stoppage 7-24
Denial of Service Attack and Service Stoppage 7-24
Virus Outbreak Detection 7-25
Worm Outbreak Detection 7-25
Trojan Horse Detection 7-26
Multiple Backdoor Attempts from a Single Source 7-26
Multiple Backdoor Attempts from Different Sources 7-26
Multiple Login Failures from Any Source to Any Destination 7-27
Multiple Login Failures from Same Source to Same Destination 7-27
Buffer Overflow Attack from Same Source to Same Target 7-28
Brute Force Attack Success where Source and Target Are the Same 7-28
Microsoft - Internet Information Services Attacks (IIS) Verification 7-29
Microsoft - Microsoft Data Access Connector (MDAC) Attack -- Remote Data Services Attack Verification 7-29
Microsoft– SQL Server Attacks – SQL Server Attack Verification 7-29
Microsoft - NETBIOS - Unprotected Windows Network Shares Attack Verification 7-30
Microsoft - Anonymous Login - Null Sessions Attack Verification 7-30
Microsoft - LAN Manager (LM) Authentication - Weak LM Hashing Attacks Verification 7-30
Microsoft - General Windows Authentication Attack Verification 7-31
Microsoft - Internet Explorer (IE) Attacks Verification 7-31
Microsoft - Remote Registry Access Attack Verification 7-31
Microsoft - Windows Scripting Attack Verification 7-32
UNIX - Remote Procedure Call (RPC) Attack Verification 7-32
UNIX - Apache Web Server Attack Verification 7-32
UNIX - Secure Shell Attack Verification 7-33
UNIX – Simple Network Management Protocol (SNMP) Attack Verification 7-33

UNIX - File Transfer Protocol (FTP) Attack Verification 7-33
UNIX - Remote Services Attack Verification 7-34
UNIX - Line Printer Daemon Attack Verification...7-34
UNIX - Sendmail Attack Verification...7-34
UNIX - BIND/DNS Attack Verification...7-35
UNIX - General UNIX Authentication Attack Verified 7-35
Taxonomy Tables...7-35
NIDS Taxonomy Table 7-35
HIDS & OS Taxonomy Table 7-39
Correlation Output...7-43
Correlation Rule Output Structure 7-43
Script Parameters Passed 7-43

Chapter 8 – Sentinel Correlation Command Line Options 8-1

Chapter 9 – Sentinel Data Access Service 9-1

DAS Container Files..9-1
Reconfiguring Database Connection Properties 9-2
DAS Configuration Files..9-3
Native DB Connectors for Event insertion..9-4

Chapter 10 – Changing Default User Passwords.......................................10-1

Changing Default User Passwords for Oracle and MS SQL Authentication 10-1
Changing esecadm Password 10-1
Changing esecapp Password 10-1
Changing esecdba Password 10-2
Changing esecrpt Password 10-2
Changing Default User Passwords for Windows Authentication............................10-3
Changing the e-Security Administrator Password..10-3
Changing the e-Security DB Administrator Password...10-3
Changing the e-Security Application DB Administrator Password 10-4
Changing the e-Security Report User Password..10-5

Chapter 11 - e-Security Database Views for Oracle 11-1

Views..11-1
ADV_ALERT_CVE_RPT_V 11-1
ADV_ALERT_PRODUCT_RPT_V 11-1
ADV_ALERT_RPT_V..11-1
ADV_ATTACK_ALERT_RPT_V...11-2
ADV_ATTACK_CVE_RPT_V...11-2
ADV_ATTACK_MAP_RPT_V 11-2
ADV_ATTACK_PLUGIN_RPT_V...11-3
ADV_ATTACK_RPT_V 11-3
ADV_CREDIBILITY_RPT_V 11-4
ADV_FEED_RPT_V..11-4
ADV_PRODUCT_RPT_V..11-4
ADV_PRODUCT_SERVICE_PACK_RPT_V 11-5
ADV_PRODUCT_VERSION_RPT_V 11-5
ADV_SEVERITY_RPT_V..11-5
ADV_SUBALERT_RPT_V 11-6
ADV_URGENCY_RPT_V 11-6
ADV_VENDOR_RPT_V...11-6
ADV_VULN_PRODUCT_RPT_V...11-7
ANNOTATIONS_RPT_V..11-7

ASSET_CTGRY_RPT_V 11-7
ASSET_HOSTNAME_RPT_V...11-8
ASSET_IP_RPT_V 11-8
ASSET_LOCATION_RPT_V...11-8
ASSET_RPT_V...11-9
ASSET_VALUE_RPT_V 11-9
ASSET_X_ENTITY_X_ROLE_RPT_V...11-9
ASSOCIATIONS_RPT_V...11-10
ATTACHMENTS_RPT_V...11-10
CONFIGS_RPT_V 11-10
CONTACTS_RPT_V..11-11
CORRELATED_EVENTS_RPT_V..11-11
CORRELATED_EVENTS_RPT_V1..11-11
CRITICALITY_RPT_V...11-12
CUST_RPT_V...11-12
ENTITY_TYPE_RPT_V...11-12
ENV_IDENTITY_RPT_V..11-13
ESEC_DISPLAY_RPT_V...11-13
ESEC_PORT_REFERENCE_RPT_V..11-14
ESEC_PROTOCOL_REFERENCE_RPT_V.......................................11-14
ESEC_SEQUENCE _RPT_V...11-15
EVENTS_ALL_RPT_V (Provided for backward compatibility purpose)............11-15
EVENTS_ALL_RPT_V1 (Provided for backward compatibility purpose)...........11-19
EVENTS_RPT_V (Provided for backward compatibility purpose) 11-20
EVENTS_RPT_V1 (Provided for backward compatibility purpose) 11-20
EVENTS_RPT_V2 (All new Sentinel 5 reports should use this view)...............11-20
EVT_AGENT_RPT_V 11-24
EVT_ASSET_RPT_V...11-24
EVT_DEST_EVT_NAME_SMRY_1_RPT_V 11-25
EVT_DEST_SMRY_1_RPT_V..11-26
EVT_DEST_TXNMY_SMRY_1_RPT_V 11-26
EVT_NAME_RPT_V 11-26
EVT_PORT_SMRY_1_RPT_V 11-27
EVT_PRTCL_RPT_V...11-27
EVT_RSRC_RPT_V 11-27
EVT_SEV_SMRY_1_RPT_V 11-27
EVT_SRC_SMRY_1_RPT_V...11-28
EVT_TXNMY_RPT_V 11-28
EVT_USR_RPT_V...11-28
EXTERNAL_DATA_RPT_V 11-29
HIST_ EVENTS_RPT_V 11-29
HIST_ INCIDENTS_RPT_V 11-29
IMAGES_RPT_V...11-29
INCIDENTS_ASSETS_RPT_V 11-29
INCIDENTS_EVENTS_RPT_V..11-30
INCIDENTS_RPT_V 11-30
INCIDENTS_VULN_RPT_V..11-31
L_STAT_RPT_V 11-31
LOGS_RPT_V...11-31
NETWORK_IDENTITY_RPT_V...11-31
ORGANIZATION_RPT_V 11-32

PERSON_RPT_V 11-32
PHYSICAL_ASSET_RPT_V 11-32
PRODUCT_RPT_V 11-32
ROLE_RPT_V 11-33
SENSITIVITY_RPT_V 11-33
STATES_RPT_V 11-33
UNASSIGNED_INCIDENTS_RPT_V View 11-34
USERS_RPT_V 11-34
VENDOR_RPT_V 11-34
VULN_CALC_SEVERITY_RPT_V 11-35
VULN_CODE_RPT_V 11-35
VULN_INFO_RPT_V 11-35
VULN_RPT_V 11-36
VULN_RSRC_RPT_V 11-36
VULN_RSRC_SCAN_RPT_V 11-37
VULN_SCAN_RPT_V 11-37
VULN_SCAN_VULN_RPT_V 11-37
VULN_SCANNER_RPT_V 11-38
Chapter 12 - e-Security Database Views for Microsoft SQL Server 12-1
Views 12-1
ADV_ALERT_CVE_RPT_V 12-1
ADV_ALERT_PRODUCT_RPT_V 12-1
ADV_ALERT_RPT_V 12-1
ADV_ATTACK_ALERT_RPT_V 12-2
ADV_ATTACK_CVE_RPT_V 12-2
ADV_ATTACK_MAP_RPT_V 12-2
ADV_ATTACK_PLUGIN_RPT_V 12-3
ADV_ATTACK_RPT_V 12-3
ADV_CREDIBILITY_RPT_V 12-4
ADV_FEED_RPT_V 12-4
ADV_PRODUCT_RPT_V 12-4
ADV_PRODUCT_SERVICE_PACK_RPT_V 12-5
ADV_PRODUCT_VERSION_RPT_V 12-5
ADV_SEVERITY_RPT_V 12-5
ADV_SUBALERT_RPT_V 12-6
ADV_URGENCY_RPT_V 12-6
ADV_VENDOR_RPT_V 12-6
ADV_VULN_PRODUCT_RPT_V 12-7
ANNOTATIONS_RPT_V 12-7
ASSET_CTGRY_RPT_V 12-8
ASSET_HOSTNAME_RPT_V 12-8
ASSET_IP_RPT_V 12-8
ASSET_LOCATION_RPT_V 12-8
ASSET_RPT_V 12-9
ASSET_VALUE_RPT_V 12-9
ASSET_X_ENTITY_X_ROLE_RPT_V 12-9
ASSOCIATIONS_RPT_V 12-10
ATTACHMENTS_RPT_V 12-10
CONFIGS_RPT_V 12-10
CONTACTS_RPT_V 12-11

CORRELATED_EVENTS_RPT_V....................12-11
CORRELATED_EVENTS_RPT_V1....................12-11
CRITICALITY_RPT_V....................12-12
CUST_RPT_V....................12-12
ENTITY_TYPE_RPT_V....................12-12
ENV_IDENTITY_RPT_V....................12-13
ESEC_DISPLAY_RPT_V....................12-13
ESEC_PORT_REFERENCE_RPT_V....................12-13
ESEC_PROTOCOL_REFERENCE_RPT_V....................12-14
ESEC_SEQUENCE _RPT_V....................12-15
EVENTS_ALL_RPT_V (Provided for backward compatibility purpose)....................12-15
EVENTS_ALL_RPT_V1 (Provided for backward compatibility purpose)....................12-19
EVENTS_RPT_V (Provided for backward compatibility purpose)....................12-19
EVENTS_RPT_V1 (Provided for backward compatibility purpose)....................12-20
EVENTS_RPT_V2 (Provided for backward compatibility purpose)....................12-20
EVT_AGENT_RPT_V....................12-24
EVT_ASSET_RPT_V....................12-24
EVT_DEST_EVT_NAME_SMRY_1_RPT_V....................12-25
EVT_DEST_SMRY_1_RPT_V....................12-25
EVT_DEST_TXNMY_SMRY_1_RPT_V....................12-26
EVT_NAME_RPT_V....................12-26
EVT_PORT_SMRY_1_RPT_V....................12-26
EVT_PRTCL_RPT_V....................12-27
EVT_RSRC_RPT_V....................12-27
EVT_SEV_SMRY_1_RPT_V....................12-27
EVT_SRC_SMRY_1_RPT_V....................12-27
EVT_TXNMY_RPT_V....................12-28
EVT_USR_RPT_V....................12-28
EXTERNAL_DATA_RPT_V....................12-28
HIST_EVENTS_RPT_V....................12-29
HIST_INCIDENTS_RPT_V....................12-29
IMAGES_RPT_V....................12-29
INCIDENTS_ASSETS_RPT_V....................12-29
INCIDENTS_EVENTS_RPT_V....................12-29
INCIDENTS_RPT_V....................12-30
INCIDENTS_VULN_RPT_V....................12-30
L_STAT_RPT_V....................12-31
LOGS_RPT_V....................12-31
NETWORK_IDENTITY_RPT_V....................12-31
ORGANIZATION_RPT_V....................12-31
PERSON_RPT_V....................12-32
PHYSICAL_ASSET_RPT_V....................12-32
PRODUCT_RPT_V....................12-32
ROLE_RPT_V....................12-32
SENSITIVITY_RPT_V....................12-33
STATES_RPT_V....................12-33
UNASSIGNED_INCIDENTS_RPT_V View....................12-33
USERS_RPT_V....................12-34
VENDOR_RPT_V....................12-34
VULN_CALC_SEVERITY_RPT_V....................12-34
VULN_CODE_RPT_V....................12-35

VULN_INFO_RPT_V......12-35
VULN_RPT_V......12-35
VULN_RSRC_RPT_V......12-36
VULN_RSRC_SCAN_RPT_V......12-37
VULN_SCAN_RPT_V......12-37
VULN_SCAN_VULN_RPT_V......12-37
VULN_SCANNER_RPT_V......12-37
Appendix A – Sentinel Copyright Information......A-1

Chapter 1 – Sentinel™ 5 User Reference Introduction

The e-Security User Reference Guide is your reference for:

- Wizard scripting language
- Wizard parsing commands
- Wizard administrator functions
- Wizard and Sentinel meta tags
- Sentinel console user permissions
- Sentinel correlation engine
- Sentinel command line options
- Sentinel server database schema

This guide assumes that you are familiar with Network Security, Data Base Administration and UNIX operating systems.

Contents

This guide contains the following chapters:

- Chapter 1 – e-Security User Reference Introduction
- Chapter 2 – Wizard Scripting Language
- Chapter 3 – Wizard Parsing Commands
- Chapter 4 – Wizard Administrator Functions
- Chapter 5 – Wizard and Sentinel Meta-tags
- Chapter 6 – Sentinel Control Center User Permissions
- Chapter 7 – Sentinel Correlation Engine
- Chapter 8 – Sentinel Correlation Command Line Options
- Chapter 9 – Sentinel Data Access Service
- Chapter 10 - Changing Default User Passwords
- Chapter 11 – e-Security Database Views for Oracle
- Chapter 12 - e-Security Database Views for Microsoft SQL Server
- Appendix A – Copyright information

Conventions Used

Notes and Cautions

NOTE: Notes provide additional information that may be useful.

CAUTION: Cautions provide additional information that may keep you from performing damage or loss of data to your system.

Commands

Commands appear in courier font. For example:

```
useradd –g dba –d /export/home/oracle –m –s /bin/csh
  oracle
```

Other e-Security References

The following manuals are available with the e-Security install CDs.

- Sentinel™ 5 Install Guide
- Sentinel™ 5 User’s Guide
- Sentinel™ 5 Wizard User’s Guide
- Sentinel™ 5 User’s Reference Guide

- Sentinel™5 3rd Party Integration Guide
- Release Notes

Contacting e-Security

- For Technical Support, support@esecurity.net
- For information, info@esecurity.net
- Website http://www.esecurity.net
- For 24x7 support, call Technical Support directly at 800-474-3131

Chapter 2 – Wizard Scripting Language

This chapter and the following chapter discuss how to use the Wizard scripting language to build scripts. Operators in the various strings and parsing commands that are used in agent building are covered.

The following is discussed:

- Decide Strings
- Regular Expressions

Decide Strings

Strings are case-sensitive.

As agents are being polled, various information is collected in the internal receive buffer. Decide type strings specify that a decision will be made concerning the data received and stored in the internal buffer. A decide string is evaluated to be either true or false. If there is a syntax error or if the Decide Type box is left blank, the decision is false.

The decide string is only evaluated if the Decide Type is set to string or data.

Manipulating the Rx Buffer (Receive Buffer) Pointer

Each port in Wizard has its own Receive Buffer pointer. The Receive Buffer pointer points to data bytes in the Receive Buffer. Prior to each evaluated decide string, the Receive Buffer pointer is reset to its held value (normally zero, unless it is modified by a decision that used the (:) search operator).

- 0 does not point to any byte in the receive buffer
- 1 points to the first data byte, 2 points to the second data byte and so on

Format

A decide string takes the form of a sequence of logical operators (LO) and regular expressions.

Logical operators and strings operators do not have to be present in each sequence. Some rules regarding their use are:

- Logical operators build boolean (true or false) expressions within the decide string and are evaluated based on the following precedence:
  `~ Not`

  `& And`
- A string operator specifies a string of characters to search in the receive buffer. Using the strings operator performs a search byte-for-byte from the Receive Buffer pointer position forward.

NOTE: Because the Decide Type box is cut off at the last printable character, the hex equivalent of a space must be used. The : operator cannot be used with the NULL operator.

Parameter Names

To specify a parameter in a decide string, the parameter name must be enclosed in curly braces ({ }). When the script is built, the parameter name and curly braces are replaced by the value of the parameter.

If the parameter name specified does not exist in the parameter file from which the script is built, the parameter name expression and curly braces remain in the decide string data.

Parameter name expressions can occur anywhere in the decide string. They cannot, however, be nested (include another parameter name expression within itself).

Hierarchy of Operations in a Decide String

Each operation in a decide string is evaluated as either true (1) or false (0). Operations in a decide string are always followed in the order governed by the logical operator syntax.

- When more than one operation is used, string evaluations are performed in order from left to right.
- When parentheses are used, the logical operator within each set of parentheses is evaluated first.
- The next logical operations to be evaluated are not (~), and (&).

An order of operation is also followed when using the string operator syntax:

- The reset Rx buffer pointer is evaluated first.
- All other syntax characters have equal precedence and are evaluated in order, from left to right.

Receive Buffer Pointer Rules

The following rules govern the value of the Receive Buffer pointer:

- When the search for a string of characters is successful, the search is considered to be true and the Receive Buffer pointer is positioned at the first byte in the string that was found.

  Decide String: DE

  ```
   A BCDE F GH
  ^
   A BCDE F GH
       ^
  ```

- When the search for a string of characters is unsuccessful, the search is considered to be false and the Receive Buffer pointer is returned to the hold value.

  Decide String: DEJ

  ```
   A BCDE F GH
  ^
   A BCDE F GH
  ^
  ```

Checking for an Empty Receive Buffer

To check for an empty receive buffer use the following decide string:

```
NULL
```

Decide String Evaluations and Results Example

Alphanumeric Decide Strings

The following are alphanumeric Decide Strings for a sample Receive Buffer:

```
ABCDEFGHIJKLMNO (line feed) YZ<[&
```

Decide String	Logical Expression	Result
A	1	1
P	0	0
\41\(HEX for A)	1	1
AB	1	1
\4142\(HEX for AB)	1	1
ABD	0	0
A&B	1 & 1	1
A&P	1 & 0	0
A+P	1 + 0	1
A\42\(HEX for B)	1	1
A&BC	1 & 1	1
DEF&ABC	1 & 0	0
ABC&DEF	1 & 1	1
ABC&BCD	1 & 1	1
ABC&ABC	1 & 0	0
\OA\(HEX for line feed)	1	1
NULL *	0	0

If no characters are found in the Receive Buffer, the result is TRUE.

HEX Decide Strings

The following are HEX Decide Strings for a sample Receive Buffer (HEX):

```
02 0A 10 FF 1F 2E 3C 03
```

Decide String	Logical Expression	Result
\020A\&\FF\	1 & 1	1
\02\	0	0
\02\&\03\	1 & 1	1
\03\&\02\	1 & 0	0

Regular Expressions

Special characters and sequences of characters are used in writing patterns for regular expressions.

Sentinel uses a POSIX (Portable Operating System Interface for UNIX)-compliant library for regular expressions. POSIX is a set of IEEE and ISO standards that help assure compatibility between POSIX-compliant operating systems, which includes most varieties of UNIX.

Summary of Special Characters for Regular Expressions

The following table summarizes the special characters that may be used in regular expressions for the SEARCH and REPLACE functions.

Character	Usage/Example
\	Marks the next character as special. n matches the character "n." The sequence \n matches a line feed or newline (end of line) character, but in order to pass the "\" through the parser, you must precede it with the exception character "/"; therefore, to pass a \n, you must use /\n.
^	Matches the start of the input or line.
$	Matches the end of the input or line.
*	Matches the preceding character zero or more times. go* matches either "g" or "goo."
+	Matches the preceding character one or more times. go+ matches "goo" but not "g."
?	Matches the preceding character zero or one time. a?te? matches the "te" in "eater."
.	Matches any single character except a newline (end of line) character.
x|y	Matches either x or y. z|good? matches "goo" or "good" or “z”.
{n}	n is a nonnegative integer. Matches exactly n times. e{3} does not match the "e" in "Ted," but matches the first three e’s in "greeeeeed."
{n,}	n is a nonnegative integer. Matches at least n times. e{3,} does not match the "e" in "Ted" and matches all the e’s in “greeeeeed." e{1,} is equivalent to e+.
{n,m}	m and n are nonnegative integers. Matches at least n and at most m times. e{1,3} matches the first three e’s in "greeeeeed."
[xyz]	A character set. Matches any one of the enclosed characters. [xyz] matches the "y" in "play."
[^xyz]	A negative character set. Matches any character not enclosed. [^xyz]/ matches the "v" in "vain."
[0-9]	Matches a digit character.
[^0-9]	Matches a nondigit character.
[A-Za-z0-9_]	Matches any word character, including an underscore.
[^A-Za-z0-9_]	Matches any nonword character.
/n/	Matches n, where n is an octal, hexadecimal, or decimal escape value. Allows embedding of ASCII codes into regular expressions.

White space in Regular Expressions

In regular expressions, white space consists of one or more blanks, which may be any of the following characters:

Symbolic Name	UCS	Description
<tab>	<U0009>	CHARACTER TABULATION (HT)
<carriage-return>	<U000D>	CARRIAGE RETURN (CR)
<newline>	<U000A>	LINE FEED (LF)
<vertical-tab>	<U000B>	LINE TABULATION (VT)

Symbolic Name	UCS	Description
<form-feed>	<U000C>	FORM FEED (FF)
<space>	<U0020>	SPACE

Parsing Commands

The Wizard parsing language is function-oriented. Most of the parsing functions enable you to manipulate Wizard variables and their contents. The Wizard parsing language supports four variable types:

- Integer (the variable name begins with i)
- Float (the variable name begins with f)
- Variable length strings (the variable name begins with anything other than an i or f
- Arrays of variables (the variable name ends with [ ]). Array variable types can be arrays of integers, floats, or strings

These variables are local to each Wizard port and are not shared globally across all Wizard ports. Parsing commands enable you to copy data from the receive buffer into string variables.

The receive buffer contains the data that was received from the Wizard communication port, socket port, file or process.

The length of bytes to copy, as well as the position to copy the bytes from, can be controlled using the following parsing commands:

- SEARCH()
- SKIP()
- SKIPWORD()
- NEGSEARCH()
- RESET()
- COPY()

Data from the receive buffer can be appended to a string variable with the APPEND() command. The Wizard parsing language also enables you to copy or append data from string variables into other string variables.

Simple Data Types

number

Numerals can only be preceded by a + or - in the case of the SKIP Command, SKIPWORD Command, and SET Command. For example:

```
0, 10, 2.5
```

ivar (Integer variables)

Integer variables are 32-bit signed numbers. The variable name must begin with an I or i. For example:

```
i_count, I_severity, i, i[55], i[index]
```

The integer variable, i[55], is the 55th index into the integer array, i[]. Also, the index into an array may be an integer variable.

fvar (Float variables)

Float variables are 32-bit floating point numbers. The variable name must begin with an F or f. For example:

```
f_rate, F_queue, f, f[1], f[index]
```

svar (String variables)

String variables contain variable length strings. String variable names cannot begin with an I, i, F or f. For example:

```
resource, date, _message, string[1000], string[i_sev]
```

array (Variable arrays)

Variable arrays can represent arrays of variables of type ivar, fvar and svar. For example:

```
i_bits[], F_values[], s_resources[]
```

Arrays can be indexed with any numeric index with no wasted memory space. Accessing ivar[1000] does not mean that memory is allocated for 1,000 integer variables.

An indexed array variable is treated as any other variable (ivar, svar and fvar)

For example, the following would be legal syntax for the POPUP command:

```
POPUP(xterm_display[4], data[i_count])
```

Quoted Data

Quoted data is scanned and parsed as follows:

- /=Exception character: include byte following the / without regard to any special meaning; to use one of the special characters in the string, / must be placed in front of the character. For example, corp/\router is used for corp\router
- \xx x xx\=Hex data (can be one or two characters per byte): \0ad\, \0a0d\, \a d\,\0a 0d\, and \0a d\ all mean line feed/carriage return

All other characters are specified directly.

Derived Aggregate Data Types

The following table list derived aggregate data types:

Type	Description
all	number, ivar, fvar, svar, quotes
numeric	number, ivar, fvar, ivar[index], fvar[index]
string	svar, svar[index], quotes
variable	ivar, fvar, svar, ivar[index], fvar[index], svar[index]
numvar	ivar, fvar, ivar[index], fvar[index]
array	ivar[], fvar[], svar[]
numvar array	ivar[], fvar[]
string variable array	svar[]

Special Rules for Variables

The following are special rules for variables.

- Variable names are case-sensitive
- When a numvar is used for the first time, except in the cases where it is having its value set, it is set to zero
- When an svar is used for the first time, except in the cases where it is having its value set, it is set to null ("")
- An indexed array is treated like any other variable of its type, ivar, fvar or svar
- To comment out one or more parsing commands, or to place comments into the parsing text, enclose the comments in /* */

For example:

```
/* this is a comment */
/* these are commented out parsing commands
COPY(s: "test")
DISPLAY()
*/
```

Chapter 3 – Wizard Parsing Commands

This chapter lists the Wizard Parsing Commands used in agent building in alphabetical order. Below is a listing of the Parsing Commands by Function.

Function	Parsing Command
Database Interaction	DBCLOSE DBDELETE DBGETROW DBINSERT DBOPEN DBSELECT
Debugging	BREAKPOINT DISPLAY POPUP
File Interaction	FILEA FILEL FILER FILEW

Logical Operations	COMPARE
	ELSE
	ENDFOR
	ENDIF
	ENDWHILE
	FOR
	IF
	LOOKUP
	WHILE

Network Interaction	ESNMPGET ESNMPSET SNMPGET SNMPSET SOCKETW
Notification	ALERT CLEARTAGS CONSTANTTAGS EVENT INDICATOR PAUSE
Raw Data Manipulation	BITFIELD BYTEFIELD CONVERT CRC DECODE ENCODE HEXTONUM NUMTOHEX SETBYTES STRIP STRIP-ASCII-RANGE

Function	Parsing Command
String Manipulation	APPEND COPY COPY-FROM-RX-BUFF-UNTIL-SEARCH COPY-FROM-RX-BUFF COPY-FROM-STRING-TO-STRING-UNTIL-SEARCH COPY-STRING-TO-STRING LENGTH LENGTH-OPTION2 NEGSEARCH PARSER_ATTACHVARIABLE PARSER_CREATEBASIC PARSER_NEXT PARSER_PARSESTRING PRINTF REGEXPREPLACE REGEXPSEARCH REGEXPSEARCH_EXPLICIT REGEXPSEARCH_STRING REPLACE SEARCH SKIP SKIPWORD STONUM TOKENIZE TOLOWER TOUPPER TOKENIZE TRANSLATE
Utility	DATE DATETIME PAUSE SHELL TBOSSETCOMMAND TBOSSETREQUEST TIME
Variable Handling	CLEAR DELETE EXPORTVAR GETCONFIG GETENV INC RESET RXBUFF SET SETCONFIG
Vulnerability Scanning	INFO_CLEARTAGS INFO_CLOSE INFO_CONSTANTTAGS INFO_CREATE

Function	Parsing Command
	INFO_DUMP INFO_PUSH INFO_SEND INFO_SETTAG

Command Format and Using Arrays

Parsing command formats use certain symbols to convey specific meanings. The following are examples of those symbols:

Example of Symbol in Use	Example of Symbol's Meaning
`[parameter]`	Straight brackets indicate optional parameters.
`<parameter>`	Angled brackets indicate required parameters you supply.
`a`	a must literally be typed here
`a|b`	use exactly either a or b, but not both
`<item> ::= <definition>`	item can be replaced by definition

`<varList>`
where:
`<varList> ::= var`
`[, <varList>]`
	used for recursive definitions to describe a list of variables in which at least one variable is required

`...`	Repetition of the preceding parameter(s) is allowed.
`/`	The forward slash is used as an "escape" to enable the use of special characters such as the backslash (\).

Arrays are allowed in expressions, for example:

Given	The following are equivalent
`SET(i_var = 2)`	`i_arr[3]`

`SET(i_arr[3]=2`	`i_arr[i_var]`
	`i_arr[1+2]`
	`i_arr[1+1_var]`
	`i_arr[i_arr[3]]`

Commands

ALERT

The ALERT command forwards event messages to Sentinel.

- The first required parameter defines the resource name
- The second required parameter defines the event message text
- The third required parameter defines the event severity
- The date and time of the event message can be defined as optional parameters.
    - The date parameter may be used by itself
    - The time parameter must be paired with the date parameter

Format

```
ALERT(resource, message, iseverity)
```

or

```
ALERT(resource, message, iseverity[, date[, time]])
```

You cannot use the date parameter unless it is paired with the time parameter.

> **NOTE:** Use the STONUM Command to convert iseverity from a string to an integer.

Data Types

Argument	Type	Description
resource	string (INPUT)	The resource and optionally, the sub-resource, to send an event to (for example: `xterm:tcp_retransmits`).
message string	(INPUT)	The message text for the event message.
iseverity	numeric (INPUT)	The numeric representation of the priority of this event message (0 -5). 0 = Informational 1 = Advisory 2 = Warning 3 = Minor 4 = Major 5 = Critical
date string	(INPUT) [OPTIONAL]	Sets the date of the event message in the format MM-DD-YYYY (for example: “12-01-2002”) (default = today’s date).
time string	(INPUT) [OPTIONAL]	Sets the time of the event message in the format HH:MM:SS (for example: “15:14:34”) (default = the current time); must be used with the date parameter.

For example:

```
ALERT("xterm:tcp_retransmits", mesg_txt,ivar[3])
ALERT("router_subnet_15", msg_txt, "c")
ALERT(resource, "Server not responding", iseverity)
ALERT("Mux184:card1", "C1 not funct. properly.", 4)
ALERT("Firewall", "Connection lost to Firewall.", 5)
ALERT("CB5", "Channel Bank 5 being serviced", "Maint")
ALERT(resource, message, isev, thedate, thetime)
ALERT("Switch3", oos_msg, 5, "07-30-1997", "07:03:23")
```

APPEND

The APPEND command adds data from the receive buffer, a string variable or a quoted string to a string variable. The following apply:

- Every APPEND parameter is optional except the destination parameter
- The destination for the data (string variable) can be specified with the APPEND parameters
- An offset into source can be specified to control where data is copied from the source data
- The number of bytes to be appended to the destination variable can be specified with the length parameter (ilen), or the length will default to the length of the source data
- In addition to specifying a numeric length parameter, a string can be used to define the length
- If a string is used as the length parameter, the source parameter must either be the receive buffer or an svar
- By using a string as the length parameter, the Agent Engine appends bytes from the source data (starting at offset) into the destination variable up to, but not including, the first character of the string (if found) (if the string is not found, no bytes are appended)
- If the offset or length parameters are specified out of the range of the source variable, then as many bytes as possible are appended, up to the end of the source data
- If the offset is greater than or equal to the length of the source data, no bytes are appended into the destination variable (if an offset is not specified, the offset defaults to zero)

Format

```
APPEND(<dest>: [source] [, [search] [, [ilen] [,
   [ioffset] ]]])
APPEND(<dest>: [source] [, [ilen] [, [ioffset] ]])
APPEND(<dest>: [ilen] [, [offset]])
```

Data Type

Argument	**Type**	**Description**
dest	svar (OUTPUT)	The data string variable to which bytes are appended.
source	string (INPUT) [OPTIONAL] or svar	The string where source bytes are located that will be appended to the destination string. (default = Receive Buffer) If the search parameter is used.
search	string (INPUT) [OPTIONAL]	A string used to specify: copy up to the bytes to search for in the source string.
ilen	numeric (INPUT) [OPTIONAL]	The number of bytes to append from the source to the destination.
ioffset	numeric (INPUT) [OPTIONAL]	The offset into the source at which to start appending data.

The following examples append bytes from the receive buffer to a destination svar (dest). The Rx buffer pointer position is added to the offset value to specify the first position of the data to be appended. The ^ symbol indicates the Rx buffer pointer position.

```
APPEND(svar:ilen)
APPEND(svar:3)
APPEND(svar:,ioffset)
APPEND(source:ilen,ioffset)
APPEND(svar: 10, 12)
```

The above example was made with the following assumptions.

```
rxbuff="receive buffer"
   ^ (Rx buffer pointer position)
dest="A destination string"
source="A source string"
ilen=3
ioffset=3
```

Enter the following:

```
APPEND(dest:)
```

Result:

```
dest = "A destination stringreceive buffer"
```

Or if you have entered:

```
APPEND(dest:ilen)
```

Result:

```
dest = "A destination stringrec"
```

Or if you have entered:

```
APPEND(dest:,ioffset)
```

Result:

```
dest = "A destination stringreceive buffer"
```

The following examples append bytes from the receive buffer up to, but not including, the search string to a destination svar (dest). If the search string is not found in the receive buffer (after the Rx buffer pointer + offset position), no bytes are appended.

Enter the following:

```
APPEND(dest:,"buffer")
```

Result:

```
dest = "A destination stringreceive "
```

Enter the following:

```
APPEND(dest:,"buffer", 9)
```

Result:

```
dest = "A destination string"
```

The following examples are to to append a substring from the receive buffer with the assumption that:

```
Rx Buffer = "Minor Alarm Firewall A"
```

Enter the following:

```
COPY(message:"Resource Name is: ")
APPEND(message:,6)
```

Result:

```
message = "Resource Name is: Alarm Firewall A"
```

BITFIELD

The BITFIELD command converts bytes into bits. This command converts each byte in a string of arbitrary length into 8 bits (0 or 1) by putting them into an integer array, float array or string.

> **CAUTION:** The output is 8 times larger than the input, so the bitfield parsing command could be very memory intensive if used improperly. For example, using input strings that have a very large number of bytes in them.

Format

```
BITFIELD(s_bytes, dest_var)
```

Data Types

Argument	Type	Description
s_bytes	string (INPUT)	Any number of ASCII or hex bytes in a string.

Argument	Type	Description

dest_var	numvar array (OUTPUT)
	Or
	svar (OUTPUT)
		Array of integers (set to 0 or 1). The number of bits equals the number of bytes in s_bytes times 8. For each 8-bit set, the bits are placed from Most Significant Bit (MSB) to Least Significant Bit (LSB). For example:
		idest_var[0] = MSB of Byte 1
		idest_var[1] = Next MSB of Byte 1
		idest_var[2] = Next MSB of Byte 1
		idest_var[3] = Next MSB of Byte 1
		idest_var[4] = Next MSB of Byte 1
		idest_var[5] = Next MSB of Byte 1
		idest_var[6] = Next MSB of Byte 1
		idest_var[7] = LSB of Byte 1
		idest_var[8] = MSB of Byte 2
		idest_var[9] = Next MSB of Byte 2
		...
		idest_var[n * 8 - 1] = LSB of Byte n
		A string that contains a multiple of 8 bytes where each byte represents a bit in the input bytes. The bytes in this string will always be set to an ASCII 0 or 1.
		For each consecutive 8 bits represented in each string, the ASCII (0s and 1s) are placed from MSB to LSB. For example:
		If s_bytes = “\5AFE\”
		Then,
		dest_var= "0101101011111110"

NOTE: The second parameter to bitfield (`dest_var`) must be a string (e.g., `ivar[]` or `fvar[]`).

For example:

```
BITFIELD("\00\", f_bit_array[])
BITFIELD(s_bytes, i_bit_array[])
BITFIELD(s_byte, string_out)
BITFIELD("This will work", i_bit_array[])
BITFIELD("\563F\", string_out)
```

In the following example, the string sbyte is set to a hex byte and sent to the BITFIELD command twice (once for an integer array and once for a string).

```
COPY(sbyte:"\AE\")
BITFIELD(sbyte, ibits[])
BITFIELD(sbyte, sbits)
```

Current Output Variables' Contents

```
ibits[0] = 1
ibits[1] = 0
ibits[2] = 1
ibits[3] = 0
ibits[4] = 1
ibits[5] = 1
ibits[6] = 1
ibits[7] = 0
sbits = "10101110"
```

BREAKPOINT

The BREAKPOINT command halts the execution of a parsing script. When the Wizard script debugger is running, the breakpoint command stops the parser pending user intervention. For example, from Wizard's Debugger panel, select the Go or Step button to resume the debugging process.

Format

```
BREAKPOINT()
```

BYTEFIELD

The BYTEFIELD command takes a bit (0 or 1) representation of byte(s) and puts the bytes into a string variable.

The input may be a:

- string
- integer array
- float array

The output is always a string variable.

Format

> **CAUTION:** If the first parameter is an integer or float array, do not use values greater than 100 for i_num_bytes, since the array will be initialized to that many entries (this could be memory intensive with large values of i_num_bytes).

```
BYTEFIELD(source_var, s_bytes[, i_num_bytes])
```

> **NOTE:** The first parameter to BYTEFIELD (`source_var`) must be `svar, ivar[], or fvar[].`

Data Types

Argument	Type	Description
source_var	numvar array (INPUT)	Array of integers (set to 0 or 1). The number of bits equals the number of bytes in s_bytes times 8. For each 8-bit set, the bits are placed from Most Significant Bit (MSB) to Least Significant Bit (LSB) (see examples located below this table).
	svar (INPUT)	A string that contains a multiple of 8 bytes where each byte represents a bit in the input bytes. The bytes in this string should always be set to an ASCII 0 or 1. For each consecutive 8 bits represented in each string, the ASCII (0s and 1s) should be placed from MSB to LSB. For example: If source_var = "0101101011111110", and i_num_bytes = 2, Then, s_bytes = “\5AFE\”
s_bytes	string (OUTPUT)	Any number of bytes of hex or ASCII data in a string.
i_num_bytes	numeric (INPUT) [OPTIONAL]	The number of bytes to place into the _bytes. Since it is optional, the default is 1 unless it is used when the input is of type STRING. If the input is of type STRING, then the default is the size of the string divided by 8.

Examples specific to source_var are:

```
ISOURCE_VAR[0] = MSB of Byte 1
ISOURCE_VAR[1] = Next MSB of Byte 1
ISOURCE_VAR[2] = Next MSB of Byte 1
ISOURCE_VAR[3] = Next MSB of Byte 1
ISOURCE_VAR[4] = Next MSB of Byte 1
ISOURCE_VAR[5] = Next MSB of Byte 1
ISOURCE_VAR[6] = Next MSB of Byte 1
ISOURCE_VAR[7] = LSB of Byte 1
ISOURCE_VAR[8] = MSB of Byte 2
ISOURCE_VAR[9] = Next MSB of Byte 2
...
ISOURCE_VAR[n * 8 - 1] = LSB of Byte n
```

Some BYTEFIELD examples:

```
BYTEFIELD(i_bit_array[], s_bytes)
BYTEFIELD(string_bits_in, s_bytes)
BYTEFIELD(f_bit_array[], string_bytes, 2)
BYTEFIELD(i_bit_array[], string_bytes, i_num_bytes)
```

In the following example, the string, sbyte and the integer array ivar are set to a bit representation of a hex byte and sent to the BYTEFIELD command twice (once for the integer array input and once for the string input).

```
SET(ivar[0] = 0)
SET(ivar[1] = 0)
SET(ivar[2] = 0)
SET(ivar[3] = 0)
SET(ivar[4] = 1)
SET(ivar[5] = 1)
SET(ivar[6] = 1)
SET(ivar[7] = 1)
COPY(sbits:"11110000")
BYTEFIELD(ivar[], sbyte1)
BYTEFIELD(sbits, sbyte2, 1)
```

Current output variables' contents:

```
sbyte1 = "\0F\"
sbyte2 = "\F0\"
```

CLEAR

The CLEAR command truncates string variables to zero bytes or sets integer variables and float variables to zero. Up to 100 variables can be specified in one CLEAR command.

Format

```
CLEAR(<varlist>)
```

Where:

```
varlist ::= var [, <varlist>]
Var ::= variable to clear (fvar, ivar, or svar)
```

Maximum number of variables: 100

Data Types

Argument	Type	Description
var1	variable (INPUT/ OUTPUT)	The variable to clear (fvar, ivar or svar).
var2	variable (INPUT/ OUTPUT) [OPTIONAL]	The variable to clear (fvar, ivar or svar).
var3	variable (INPUT/ OUTPUT) [OPTIONAL]	The variable to clear (fvar, ivar or svar).
...	variable (INPUT/ OUTPUT) [OPTIONAL]	Other variables to clear (fvar, ivar or svar).

For example:

```
CLEAR(var1)
CLEAR(var1,var2)
CLEAR(var1,var2,var3)
CLEAR(svar[45])
CLEAR(imatrix[5][5])
CLEAR(ivar, fvar, i_len, data_string[i_var])
CLEAR(temp)
CLEAR(sdata[index_x][index_y])
CLEAR(f_bits[3], i_var_array[2])
CLEAR(i_counter, temp)
```

In the following examples, values are assigned to string variables, the string variables are then used in an event message and the string variable’s values are cleared.

```
COPY(res_var: "Firewall")
COPY(msg_var: "Firewall 116 Minor Alarm")
ALERT(res_var, msg_var, 4)
CLEAR(res_var, msg_var)
RESULT:
res_var = “”
msg_var = “”
```

CLEARTAGS

The CLEARTAGS command performs a clear on all event reserved and date/time reserved variables that are not protected by the CONSTANTTAGS command.

This command should be called in the initialization state (state 4 in the e-Security standard template) of the agent before any input is parsed into the reserved variables.

The CLEARTAGS command operates on the event reserved variables and the date/time reserved variables. The CLEARTAGS command takes no parameters. The string variables are set to empty string “”; for example:

```
s_EVT and s_Sec.
```

The integer variable i_Severity is set to zero.

Format

```
CLEARTAGS ()
```

For example:

```
SET(i_Severity = 3)
COPY(s_BM:"Base Message")
COPY(s_Example:"Test")
CLEARTAGS()
```

Result:

```
i_Severity = 0
s_BM = ""
s_Example = "Test"
```

NOTE: s_Example is not an event or date/time reserved variable, so it was not cleared.

COMMENT

This takes one optional argument, which is a string. This is a method to enter comments into the agent template file. This allows you to enter comments from the visual editor without switching to the text editor.

Format

```
/*[string]*/
```

For example:

```
/* AGENT INFORMATION
; ---------------------------------------------------
Agent_Name:               Standard Template
Agent_Description:        Template to base new Wizard
   Agents on
Agent_Manufacturer:       N/A
Agent_Product/Version:    N/A
Agent_Version:            release 4.1
Agent_Date:               August 2003
; ---------------------------------------------------
   */
```

COMPARE

The COMPARE command examines two arguments and sets a variable depending on the result. The result of the comparison of type string or type numeric can be stored into a variable. If the variable is of type ivar, fvar or string, the variable will contain the value -1, 0 or 1.

- -1 is used if arg1 is less than arg2
- 0 is used if arg1 is equal to arg2
- 1 is used if arg1 is greater than arg2

Format

```
COMPARE(arg1, arg2, dest)
```

Data Types

Argument	Type	Description
arg1	all (INPUT)	Compare data 1. Must be a string or numeric.
arg2	all (INPUT)	Compare data 2. Must be the same type as Compare data 1.

dest	variable
	(OUTPUT)
		The variable in which the results of the compare will be placed:
		svar = “-1”, “0” or “1”
		ivar = -1, 0 or 1
		fvar = -1.0, 0.0 or 1.0

NOTE: The types of arg1 and arg2 must be either both a string or both numeric.

For example:

```
COMPARE(i_counter, 0, temp)
COMPARE(sdata, "ALM", i_sdata_cmp_val)
COMPARE(i_counter, i_counter2, temp)
COMPARE(i_counter, i_counter2, i_result[i_counter])
```

In the following example, text is compared to the contents of a string variable and the result of the comparison is stored in an integer variable. An event generates if the text is not the same as the value of the string variable.

```
COMPARE(s_data_var, "ALARM", i_compare_var)
IF(i_compare_var = 0)
ALERT(res_var, "Major ALARM", 5)
ENDIF()
```

NOTE: The IF( ),ELSE( ) and ENDIF( ) commands perform the same function as the COMPARE command, with the exception of comparing negative numbers.

CONSTANTTAGS

The CONSTANTTAGS command takes a variable number of parameters of reserved variable names (event and date/time). By declaring a reserved variable constant it protects the variable from being cleared by a call to the CLEARTAGS command.

An example of such a variable is s_PN, which holds the product name that the agent is processing. The s_PN variable should be declared constant and set once in the agent setup state.

This command should be called in the agent setup state (state 1 in the 4.1 standard template) for reserved variables that do not change as the agent processes events.

The CONSTANTTAGS command operates on the event reserved variables and the date/time reserved variables.

Format

```
CONSTANTTAGS (<reserved_variable> [, ...])
```

Data Types

Argument	Type	Description
reserved_variable		The list of reserved variables that will be set constant and not cleared by the CLEARTAGS Command.

For example:

```
COPY(s_PN:"PN")
COPY(s_ST:"ST")
COPY(s_BM:"BM")
CONSTANTTAGS(s_PN,s_ST)
CLEARTAGS()
```

Result:

```
s_PN = "PN"
s_ST = "ST"
s_BM = ""
```

Of the three event reserved variables, s_BM was not protected from CLEARTAGS by CONSTANTTAGS, so it was cleared.

CONVERT

The CONVERT command transforms an input string of type binary, octal, decimal, hex or raw into an output string variable into type binary, octal, decimal, hex or raw.

Format

```
CONVERT(string_in, type_in, svar_out, type_out)
```

Data Types

Argument	Type	Description
string_in	String (INPUT)	The input string to convert.

type_in	Pick List
	String
	String Var
	(INPUT)
		The type of the input string (string_in):
		Binary = “B” or “b”
		Octal = “O” or “o”
		Decimal = “D” or “d”
		Hex = “H” or “h”
		Raw = “R” or “r”

svar_out	svar (OUTPUT)	The string variable that contains the converted string data.

type_out	Pick List
	String
	String Var
	(INPUT)
		The type to convert the data to (converted string will be placed in svar_out):
		Binary = “B” or “b”
		Octal = “O” or “o”
		Decimal = “D” or “d”
		Hex = “H” or “h”
		Raw = “R” or “r”

For example:

```
CONVERT("10101010", "b", shex, "h")
CONVERT(sdata, "B", sraw, "r")
CONVERT("2356", "d", soctal, "o")
CONVERT("\3A\", "r", sbinary, "b")
CONVERT("2A3E", "h", sraw, "r")
CONVERT(data, "r", sdecimal, "d")
CONVERT(data, "o", shex, "H")
```

In the following example, the CONVERT command is called to perform various conversions.

```
CONVERT("\0afe\", "R", sdecimal, "D")
CONVERT("63", "d", sbinary, "b")
CONVERT("63", "d", shex, "h")
CONVERT("63", "d", soctal, "o")
CONVERT("1101010111110101", "b", sraw, "r")
```

Current Output Variables' Contents are:

```
sdecimal = "2814"
sbinary = "00111111"
shex = "3F"
soctal = "077"
sraw = "\d5 f5\"
```

COPY

The COPY command duplicates data from the receive buffer or source string, placing it into a string variable or a quoted string to a string variable. The Rx buffer pointer does not change when using this command.

The destination for the data (svar) must be specified with the copy parameters.

> **NOTE:** Within the Visual Editor of the Agent Builder, COPY, COPY-FROM-RX-BUFF-UNTIL-SEARCH, COPY-FROM-RX-BUFF, COPY-FROM-STRING-TO-STRING-UNTIL-SEARCH and COPY-STRING-TO-STRING are listed as separate commands. They are same command. They are provided as descriptions for different variations of the same command. If you were to use any variation of the COPY command in the text editor, you would enter COPY.

When using this command:

- Specify an offset into source to control where data is copied from the source data.
- The number of bytes to be copied to the destination variable can be specified with the length parameter (ilen), or the length can default to the length of the source data.
- In addition to specifying a numeric length parameter, a string can be used. By using a string, the Agent Engine copies bytes from the source data (starting at offset) into the destination variable up to, but not including, the first character of the string (if found). If the string is not found, no bytes are copied.
- If the offset (ioffset) or length (ilen) parameters are specified out of the range of the source variable, then as many bytes as possible, up to the end of the source data, are copied.

  If the offset is greater than or equal to the length of the source data, no bytes are copied into the destination variable.

If an offset is not specified, the offset defaults to zero.

Format

```
COPY(<DEST>: [SOURCE] [, [SEARCH] [, [ILEN] [,
   [IOFFSET] ]]])
COPY(<DEST>: [SOURCE] [, [ILEN] [, [IOFFSET] ]])
COPY(<DEST>: [ILEN] [, [OFFSET]])
```

Data Types

Argument	Type	Description
dest	svar (OUTPUT)	The data string variable to which bytes are copied.
source string	(INPUT) [OPTIONAL] Or svar	The string where bytes are copied from (default = Receive Buffer). If the search parameter is used.
search	string (INPUT) [OPTIONAL]	A string used to specify: copy up to the bytes to search for in the source string.
ilen	numeric (INPUT) [OPTIONAL]	The number of bytes to copy from the source to the destination.
ioffset	numeric (INPUT) [OPTIONAL]	The offset into the source at which to start copying data; copies all of the characters from the receive buffer to the transmit buffer.

The following examples copy bytes from the receive buffer to a destination svar (dest). The Rx buffer pointer position is added to the offset value to specify the first position of the data to be copied. The ^ symbol identifies the Rx buffer pointer position.

The following assumptions are made:

```
rxbuff="receive buffer"
^ (Rx buffer pointer position)
dest=""
source="A source string"
ilen=3
ioffset=3
```

Command	Result
`COPY(dest:)`	`dest = "receive buffer"`
`COPY(dest:5)`	`dest = "recei"`
`COPY(dest:,5)`	`dest = "ve buffer"`

The following examples copy bytes from a source string to a destination svar (dest).

Command	Result
`COPY(dest:source)`	`dest = "A source string"`

Command	Result
`COPY(dest:source,5)`	`dest = "A sou"`
`COPY(dest:source,5,6)`	`dest = "ce st"`

The following examples copy bytes from the receive buffer up to, but not including, the search string to the string variable. If the search string is not found in the receive buffer (after the Rx buffer pointer + offset position), no bytes are copied.

> **NOTE:** For hex substitution, \0000\ terminates a string. Therefore, “xxxx\0000\yyyy” becomes “xxxx”.

The following examples copy bytes from the receive buffer up to, but not including, the search string to a destination svar (dest). If the search string is not found in the receive buffer (after the Rx buffer pointer + offset position), no bytes are copied.

Command	Result
`COPY(dest:,"buffer")`	`dest = "receive "`
`COPY(dest:,"receive")`	`dest = ""`

The following examples copy bytes from a source string (must be a string variable) up to, but not including, the search string to a destination string variable (dest). If the search string is not found in the receive buffer (after the Rx buffer pointer + offset position), no bytes are copied.

Command	Result
`COPY(dest:source," string")`	`dest = "a source"`
`COPY(dest:source," .string")`	`dest = ""`

CRC

The CRC command computes a cyclical redundancy check on a string of bytes (hex or ASCII).

Format

```
CRC(source_data, dest_crc)
```

Data Type

Argument	Type	Description
source_data	string (INPUT)	The string data to perform the crc command on.
dest_crc	svar (OUTPUT)	The string variable in which the 2 byte crc result is stored.

For example:

In the following example, the computed CRC value is compared to a saved value. If the two CRC values are the same, an event message is generated.

```
CRC(svar, s_crc_var)
IF(s_crc_var = "\0A5F\")
EVENT(res, "Correct CRC generated", 0)
```

```
ENDIF()
```

NOTE: For hex substitution, \0000\ terminates a string; therefore, “xxxx\0000\yyyy” becomes “xxxx”.

DATE

The DATE command copies the current date (in the format MM-DD-YYYY) into a string variable. Optionally, it can copy the current day of the week into a string, integer, or float variable.

Format

```
DATE(date_string [, day_of_week] [, i_day_of_week]
   [, f_day_of_week])
```

Data Type

Argument	Type	Description
date_string	svar (OUTPUT)	The string variable in which the date will be stored (for example: svar = “11-18-2002”).

day_of_week	svar (OUTPUT) [OPTIONAL]
	ivar (OUTPUT) [OPTIONAL]
	Or
	fvar (OUTPUT) [OPTIONAL]
		(Optionally) The string variable in which the day of the week will be stored; written as the full Day name (for example: svar = Saturday)
		(Optionally) The integer or float variable in which the day of the week will be stored; written as full Day name = number:
		Monday = 1
		Tuesday = 2
		Wednesday = 3
		Thursday = 4
		Friday = 5
		Saturday = 6
		Sunday = 7
		(for example: Monday is ivar = 1)

For example:

In the following example, the date from the system is compared to a date string. If the two dates are the same, an event message is generated.

```
DATE(date_var, day_of_week)
IF(date_var = "11-18-2002")
ALERT(res, "Happy 23rd birthday!", 0)
ENDIF()
IF(day_of_week = "Saturday")
ALERT(res, "Time to go to the beach," 0)
ENDIF()
```

DATETIME

The DATETIME command converts an integer representation of the number of seconds since January 1, 1970, to date and time string variables. Optionally, it can copy the current day of the week into a string, integer, or float variable.

Format

```
DATETIME(itime_secs, svar_date, svar_time
   [, day_of_week] [, i_day_of_week]
   [, f_day_of_week])
```

Data Types

Argument	Type	Description
itime_secs	numeric (INPUT)	The integer number that contains the number of seconds since 1970.
svar_date	svar (OUTPUT)	The string variable in which the date will be stored (for example: 02-19-96).
svar_time	svar (OUTPUT)	The string variable in which the time will be stored (for example: 15:14:33).

day_of_week	svar
	(OUTPUT)
	[OPTIONAL]
	
	ivar
	(OUTPUT)
	[OPTIONAL]
	Or
	fvar
	(OUTPUT)
	[OPTIONAL]
		(Optionally) The string variable in which the day of the week will be stored; written as the full Day name (for example: svar = Saturday)
		
		(Optionally) The integer or float variable in which the day of the week will be stored; written as full Day name = number:
		Monday = 1
		Tuesday = 2
		Wednesday = 3
		Thursday = 4
		Friday = 5
		Saturday = 6
		Sunday = 7
		(for example: Monday is ivar = 1)

For example:

In the following example, the DATETIME command converts the number of seconds since 1970 into date and time strings:

```
DATETIME(0, sdatevar, stimevar)
```

In the following example, the DATETIME command gives you the day of the week, as well as the date and time:

```
DATETIME(946728000, sdate, stime, sday)
```

Current Output Variables' Contents:

```
sdatevar = "01-01-70"
stimevar = "00:00:00"
```

```
sdate = "01-01-2000"
stime = "12:00:00"
sday = "Saturday"
```

DBCLOSE

The DBCLOSE command closes the database connection. There are two required parameters.

- The first required parameter is the database handle that is returned by the DBOPEN command. This is either an integer or an integer variable.
- The second required parameter is the status of the close. This is either an integer variable or a float variable. A “1” will be returned upon success.

Format

```
DBCLOSE(i_dbhandle, i_closestatus)
```

DBDELETE

The DBDELETE command deletes rows from the selected table based upon selection criteria. There are four required parameters.

- The first required parameter is the database handle that is returned by the DBOPEN command. This is either an integer or an integer variable.
- The second required parameter is the status of the delete. This is either an integer variable or a float variable. The number of rows deleted will be returned upon success, inclusive of 0.
- The third required parameter is the table name from which to delete rows. It can be either a string or string variable.
- The fourth optional parameter is the where clause. It allows users to filter out unwanted data by a selection criterion. If left blank, the delete will delete all rows from the table.

The error codes for the DBDELETE command are as follows:

```
>0No error
0No rows deleted
-1DB handle is invalid
```

Format

```
DBDELETE(i_dbhandle, i_deletestatus, "tablename",
   "where clause")
```

For Example:

```
DBDELETE(i_dbhandle, i_deletestatus, "tablename")
DBDELETE(i_dbhandle, i_deletestatus, s_tablename,
   "where clause")
```

DBGETROW

The DBGETROW command works in conjunction with the DBSELECT Command. The user must obtain a selection first, using DBSELECT, before retrieving rows with the DBGETROW Command. This command will retrieve the next available row from a selection, keeping a cursor open so this command may be called in a loop, retrieving the next row upon each call. There are four required parameters.

- The first required parameter is the database handle that is returned by the DBOPEN command. This is either can be an integer or an integer variable.
- The second required parameter is the handle for the select. This can be either a string or string variable. This is the same handle as was assigned during the DBSELECT command.
- The third required parameter is the status of the get. This is either an integer variable or a float variable. A “1” will be returned upon success.
- The fourth required and subsequent optional parameters are the column data returned by the command. These columns may be string variables, float variables or integer variables. Column data of a different type than the parameter type is converted to the appropriate parameter type, if possible. Thus, if the table contains a float column, but the parameter is a string, the data will be converted from a float into a string. The user may include up to 48 of these parameters.

NOTE: The command will fill the lesser of the number of parameters defined and the number of actual columns in the database. If the database has 4 columns but you supply 7 of these parameters, only the first 4 will be filled.

The error codes for the DBGETROW command are as follows:

```
1No Error
-1Error retrieving row
```

Format

```
DBGETROW(i_dbhandle, "select1", i_selectstatus,
   s_col1, s_col2, s_col3, ..., s_col48)
```

For example:

```
DBGETROW(i_dbhandle, s_selecthandle, i_selectstatus,
   s_col1, s_col2)
```

DBINSERT

The DBINSERT command inserts a row of data into the database for a selected table. There are four required parameters.

- The first required parameter is the database handle that is returned by the DBOPEN command. This is either an integer or an integer variable.

- The second required parameter is the status of the insert. This is either an integer variable or a float variable. A “1” will be returned upon success.
- The third parameter is the table name to insert the data into.
- The fourth required and subsequent optional parameters are the column data to be inserted. These columns may be of any type. The user may include up to 48 of these parameters.

The command must include the exact number of parameters needed to insert one row of data. DBINSERT will not add a new record if a unique constraint is violated.

The error codes for the DBINSERT command are as follows:

```
1 No Error
-1 DB Handle is invalid / no row inserted
-2 Data request cannot be created
-7 SQL execution error
-16 SQL syntax error
```

Format

```
DBINSERT(i_dbhandle, i_insertstatus, "theTableName",
   "data1", "data2", ..., "data48")
```

For example:

```
DBINSERT(i_dbhandle, i_insertstatus, s_theTableName,
   "data1", I_data2, f_data3)
DBINSERT(i_dbhandle, i_insertstatus, "theTableName",
   s_data1, "data2")
```

DBOPEN

The DBOPEN command opens a connection to a supported database.

On the Microsoft Windows NT Agent only, DBOPEN will not work when the database name is configured to point to a "mapped drive". Since the agent runs as a service, it (typically) runs under the "system" account. This account does not have permissions to access remote shares, including mapped drives. This means any database connection (even through OBDC) on a Windows agent must be to a completely local database.

There are five required parameters.

- The first required parameter is the database type. This can be selected via a pick list, or using a string or string variable. The acceptable value for this parameter is Oracle9i.
- The second required parameter is the database name to connect to. It may be a string or a string variable.
- The third required parameter is the user name for database. It may be a string or string variable. This field can contain any text if users have not been specifically setup to access the database.

- The fourth required parameter is the password for the user. It may be a string or a string variable. This field can contain any text if users have not been specifically setup to access the database.
- The fifth required parameter is the database handle, which is returned by this command into the integer variable or float variable. The database handle will be greater than 0 upon success.

Format

```
DBOPEN("oracle9i", "Database name", "username",
   "password", i_dbhandle)
```

For example:

```
DBOPEN(s_dbtype, s_dbname, s_username, s_password,
   i_dbhandle)
DBOPEN(s_dbtype, "dbname", s_username, "password",
   i_dbhandle)
```

DBSELECT

The DBSELECT command works in conjunction with the DBGETROW command. The DBSELECT command opens a selection cursor into the database. This grabs a snapshot of the current records in the database that meet the selection criteria. Records entered after the DBSELECT command will not show up in record retrieval until another DBSELECT command is issued to update the selection.

There are seven required parameters.

- The first required parameter is the database handle that is returned by the DBOPEN command. This is either an integer, or an integer variable.
- The second required parameter is status of the select. This is either an integer variable or a float variable. A “1” will be returned upon success.
- The third required parameter is the select identifier. This can be either a string or string variable. This should be unique, if you have more than one DBSELECT command.
- The fourth required parameter is the number of rows to skip after the select has occurred. This allows the user to position the pointer in the DBGETROW command to new data, while allowing old data to be skipped over. This may be either an integer or an integer variable.
- The fifth required parameter is the table from which to obtain the data. It may be either a string or a string variable.
- The sixth optional parameter is the where clause. It allows users to filter out unwanted data by a selection criterion. If left blank, the select will contain all rows of the table. The format of the where clause is: where column-name=’data’.
- The seventh optional parameter is the columns returned by the DBSELECT command. If left blank, the select will contain all columns of the table.

The error codes for the DBSELECT command are as follows:

```
1 No Error
-1 DB_Handle is invalid
-2 Data request cannot be created
-3 Unsuccessful autocommit setting
-4 Memory allocation error
-5 SQL syntax error
-6 SQL execution error
```

Format

```
DBSELECT( i_dbhandle, i_selectstatus, "select1",
   i_rows_to_skip, "f_atom"<, "where clause"><,
   "col1<col2><...>">)
```

For example:

```
DBSELECT(i_dbhandle, i_selectstatus, "select1",
   i_rows_to_skip, "f_atom")
DBSELECT(i_dbhandle, i_selectstatus, s_select1, 23,
   S_TABLENAME, s_whereclause)
DBSELECT(i_dbhandle, i_selectstatus, s_select1, 23,
   S_TABLENAME, "where fname=’BOB’")
DBSELECT(i_dbhandle, i_selectstatus, s_select1, 23,
   S_TABLENAME, "where fname=’BOB’", "FIRST, LAST,
   ADDRESS")
```

DEC

⊟

The DEC command decrements a numeric variable by 1. When using DEC, you must specify either an ivar or an fvar.

Format

```
DEC(i_numvar)
```

Data Types

Argument	**Type**	**Description**
i_numvar	numvar (INPUT/ OUTPUT)	The variable to decrement (ivar or fvar)

For example:

```
SET(icounter = 2)
DEC(icounter)
DEC(icounter)
```

Result:

```
icounter = 0
```

DECODE

The DECODE command reverts a string that was encoded to preserve packet identification. This command identifies the match bytes (or characters) and the escape byte(s) (or characters) in order to remove the escape character. It removes each occurrence of the escape string preceding the matched bytes each time it is found in the data.

Format

```
DECODE(data_decode, match, escape)
```

Data Types

Argument	Type	Description
data_decode	svar (INPUT/ OUTPUT)	The string data variable to decode. The decoded result is placed back in this variable.
match	string (INPUT)	The string of bytes to match in the data_decode string variable.
escape	string (INPUT)	The escape string to remove from the data_decode variable.

For example:

The following example encodes a string, copies it to save the encoded version, then decodes it with the same parameters.

```
COPY(svar:"This is just a test of decode")
ENCODE(svar, " ", "\00\")
COPY(svar_encode:svar)
DECODE(svar, " ", "\00\")
```

Current Output Variables’ Contents:

```
svar = "This is just a test of decode"
svar_encode = "This\00\ is\00\ just\00\ a\00\ test\00\
    of\00\ decode"
```

DELETE

The DELETE command removes variables from the system to free memory allocated for their storage (this is especially useful for string variables).

It is recommended to delete svars when you are done to conserve memory. Up to 100 variables can be specified in one DELETE command.

Format

```
DELETE(<varlist>)
```

Where:

```
varlist ::= var [, <varlist>]
Var ::= variable to clear (fvar, ivar, or svar)
```

Maximum number of variables: 100

Data Types

Argument	Type	Description
var1	variable (INPUT/OUTPUT)	The variable to delete (fvar, ivar or svar).
var2	variable (INPUT/OUTPUT) [OPTIONAL]	The variable to delete (fvar, ivar or svar).
var3	variable (INPUT/OUTPUT) [OPTIONAL]	The variable to delete (fvar, ivar or svar).
...	variable (INPUT/OUTPUT) [OPTIONAL]	Other variables to delete (fvar, ivar or svar).

For example:

```
DELETE(ivar1)
DELETE(sdata, i_len, i_count, svar[22])
DELETE(imatrix3d[ix][iy][iz])
DELETE(f_array[i_count], svar[4], sdata)
DELETE(ichart[3][icount])
```

DISPLAY

The DISPLAY command displays the script variables and their current values in a popup window.

You can do the following:

- Use it when debugging scripts
- If you pass a string as the parameter, it displays the contents of that string
- Strings that contain hex data are displayed in hex format (that is, string="\0a 0d\")

The program first attempts to display the string in ASCII. If the string contains both printable and non-printable hex data, the printable hex characters are displayed in ASCII and the remaining string is displayed in hex format. For hex substitution, \0000\ terminates a string; therefore, “xxxx\0000\yyyy” becomes “xxxx”.

Format

```
DISPLAY(string_data)
```

Data Types

Argument	Type	Description
string_data	string (INPUT) [OPTIONAL]	Any particular string to display. If you leave this off, then all variables contents are displayed (strings, numbers and arrays) for every script.

For example:

```
DISPLAY( )
DISPLAY(sdata_var)
DISPLAY("Hello This is String Data")
DISPLAY(sdata_var)
```

ELSE

The ELSE command marks the ending of the true portion of the previous associated if() command. Parsing commands following the ELSE() are executed if the result of the IF() is FALSE. Commands are executed up to the next corresponding ENDIF()

Format

```
ELSE()
```

For example:

```
IF(i = 10)
ALERT("I is 10")
ELSE()
ALERT("I is not 10")
ENDIF()
```

You cannot directly compare against a negative number. To do this, use either of two methods:

- Use the parsing function compare
- Indirectly compare as follows:

```
SET(i_compare_val=-10)
IF(ivar > i_compare_val)
ALERT("ivar is greater than -10")
```

```
endif()
```

ENCODE

Use the ENCODE command to preserve packet identification. This command matches bytes (or characters) in data and escapes (or prefixes) those matched bytes with an escape string. The escape string is placed in front of the matched bytes everywhere those characters are found in the data.

Format

```
ENCODE(data_encode, match, escape)
```

Data Types

Argument	Type	Description
data_encode	svar (INPUT/ OUTPUT)	The string data variable to encode. The encoded result is placed back in this variable.
match	string (INPUT)	The string of bytes to match in the data_encode string variable.
escape	string (INPUT)	The escape string to place in front of each matched byte inside of the data_encode variable.

For example:

In the following example, two data strings are encoded to prefix all spaces with “#” and another to prefix all ‘t’s and ‘h’s with “!!”.

```
COPY(data:"Preface all spaces with ‘#’")
ENCODE(data, " ", "#")
COPY(svar:"Preface ‘t‘s and ‘h’s with ‘!!’")
ENCODE(svar, "th", "!!")
```

Result:

```
data = "Preface# all# spaces# with# ‘#’"
svar = "Preface ‘!!t’s and !!h’s wi!!t!!h ‘!!’"
```

ENDFOR

The ENDFOR command marks the end of the previous for() block.

Format

```
ENDFOR()
Example
FOR(i=0,i<3,i=i+1)
ALERT("Still in loop")
```

```
ENDFOR()
```

ENDIF

The ENDIF command marks the ending of the previous if() block.

Format

```
ENDIF()
```

For example:

```
IF(i = 10)
ALERT("I is 10")
ELSE()
ALERT("I is not 10")
ENDIF()
```

You cannot directly compare against a negative number. Use one of the following methods to do this:

- Use the parsing function compare
- Indirectly compare as follows:

```
SET(i_compare_val=-10)
IF(ivar >i_compare_val)
ALERT("ivar is greater than -10")
ENDIF()
```

ENDWHILE

The ENDWHILE command marks the end of the previous while() block.

Format

```
ENDWHILE()
Example
WHILE(i<3)
SET(i=i+1)
ENDWHILE()
```

ESNMPGET

The ESNMPGET command gets an SNMP enabled variable from an SNMP managed device, using the SNMP v3 protocols. There are eight required parameters.

- The first required parameter is the hostname of IP address of the target system. It may be either a string or string variable.

- The second required parameter is the type of access required. This may be either from the pick list, or an integer. The pick list to integer map is as follows:

Description	Pick List	Integer
No Authorization, No Privacy	noAuth noPriv	0
Authorization, No Privacy	Auth noPriv	2
Authorization and Privacy	AuthPriv	3

- The third required parameter is the username with which to connect
- The fourth required parameter is the authorization password. This is only utilized if Authorization was selected. This may be either a string or string variable.
- The fifth required parameter is the privacy password. This is only utilized if Privacy was selected. This may be either a string or string variable.
- The sixth required parameter is the OID, or object ID, of the object to get. This may be either a string or string variable.
- The seventh required parameter is the returned value from the GET. This may be a string variable only. Upon error, this parameter may include an error message instead of the expected result.
- The eighth required parameter is the status of the GET. This should be either an integer variable or a float variable. A status of “1” is returned upon success.

Format

```
ESNMPGET("hostname/ipaddress", 3, "username",
   "AuthPassword", "PrivPassword", "objectID",
   s_value_retrieved, i_status)
```

For example:

```
ESNMPGET("192.168.0.23", 2, s_username,
   s_AuthPassword, "PrivPassword",
   ".1.3.6.1.4.1.4286.1.1.0", s_value_retrieved,
   i_status)
ESNMPGET("127.0.0.1", 0, "Bob", "x", "x",
   "sysDescr.0", s_value_retrieved, i_status)
```

ESNMPSET

The ESNMPSET command gets an SNMP-enabled variable from an SNMP managed device, using the SNMP v3 protocols. There are nine required parameters.

- The first required parameter is the MIB data type of the variable the user wishes to set. The allowed values can be input via the pick list, a string or string variable. The map of allowed values is as follows:

Pick List	Value
Integer	i
Unsigned Integer :	u
Time Tick	t

Pick List	Value
IP Address	a
Object ID	o
Octet String	x
Decimal String	d
Display String	s

- The second required parameter is the hostname or IP address of the target system. It may be either a string or string variable.
- The third required parameter is the type of access required. This may be either from the pick list, or an integer. The pick list to integer map is as follows:

Description	Pick List	Integer
No Authorization, No Privacy	noAuth noPriv	0
Authorization, No Privacy	Auth noPriv	2
Authorization and Privacy	AuthPriv	3

- The fourth required parameter is the username with which to connect.
- The fifth required parameter is the authorization password. This is only utilized if Authorization was selected. This may be either a string or string variable.
- The sixth required parameter is the privacy password. This is only utilized if Privacy was selected. This may be either a string or string variable.
- The seventh required parameter is the OID or object ID of the object to set. This may be either a string or string variable.
- The eighth required parameter is the value to SET. This may be a string or string variable. Upon error, this parameter may include an error message instead of the expected result.
- The ninth required parameter is the status of the SET. This should be either an integer variable or a float variable. A status of “1” is returned upon success.

Format

```
ESNMPSET("MIB Data type", "ip address", 3, "username",
   "authorization_password", "privacy_password",
   "OID", "data", i_status)
```

For example:

```
ESNMPSET("o", "192.168.0.23", 2, s_username,
   s_AuthPassword, "PrivPassword",
   ".1.3.6.1.4.1.4286.1.1.0", s_value_set, i_status)
ESNMPSET( s_type, "127.0.0.1", 0, "Bob", "x", "x",
   "sysDescr.0", s_value_set, i_status)
```

EVENT

The EVENT command creates and sends an alert message. It takes no parameters. The EVENT command automatically constructs the alert message using the contents of the reserved variables.

Most of the reserved variables map directly to the meta-tags of the v3.2 Wizard template. Only those variables that are used in the script and are not set to “” are sent. Variables like i_Severity and s_Res are required for an alert message to be processed by the Agent Manager.

Event Reserved Variables

> **NOTE**: When a label is preceded with an ‘e.’, such as e.crt, this refers to current events. If a label is preceded with a ‘w.’, such as w.crt, this refers to historical events.

Variable	Short Description	Maps to meta-tag (label)
s_BM	Base Message	Message (msg)
i_Severity	Severity	Severity (sev)
s_Res	Resource	Resource (res)
s_SubRes	SubResource	SubResource (sres)
s_ET	Event Time	EventTime (et)
s_P	Protocol	Protocol (prot)
s_DP	Destination Port	DestinationPort (dp)
s_SP	Source Port	SourcePort (sp)
s_EVT	Event Name	EventName (evt)
s_SN	Sensor Name	SensorName (sn)
s_SIP	Source IP	Source IP (sip)
s_DIP	Destination IP	DestinationIP (dip)
s_SHN	Source Host Name	SourceHostName (shn)
s_DHN	Destination Host Name	DestinationHostName (dhn)
s_SUN	Source User Name	SourceUserName (sun)
s_DUN	Destination User Name	DestinationUserName (dun)
s_FN	File Name	FileName (fn)
s_EI	Extended Information	ExtendedInformation (ei)
s_RN	Reporter Name	ReporterName (rn)
s_ST	Sensor Type	Sensor Type (st)
s_PN	Product Name	ProductName (pn)
s_CRIT	Criticality	Criticality (crt)
s_VULN	Vulnerability	Vulnerability (vul)
s_CT1	Reserved Customer 1	Ct1 (ct1)
s_CT2	Reserved Customer 2	Ct2 (ct2)
s_CT3	Reserved Customer 3	Ct3 (ct3)
s_RT1	Reserved e-Security 1	Rt1 (rt1)
s_RT2	Reserved e-Security 2	Rt2 (rt2)
s_RT3	Reserved e-Security 3	Rt3 (rt3)
s_CV1 to s_CV100	Customer Variable 1 to 100 **NOTE**: 1 to 10 is type long (number) 11 to 20 is type date 21 to 100 is type string	Cv1 to Cv100 (cv1 to cv100)

Variable	Short Description	Maps to meta-tag (label)
s_RV1 to s_RV31	Reserved Value 1 to 31 **NOTE**: Reserved for e-Security's use.	Rv1 to Rv31 (rv1 to rv31)
s_RV32	DeviceCategory	Rv32 (rv32)
s_RV33	EventContext	Rv33 (rv33)
s_RV34	SourceThreatLevel	Rv34 (rv34)
s_RV35	SourceUserContext	Rv35 (rv35)
s_RV36	DataContext	Rv36 (rv36)
s_RV37	SourceFunction	Rv37 (rv37)
s_RV38	SourceOperationalContext	Rv38 (rv38)
s_RV39	MSSPCustomerName	Rv39 (rv39)
s_RV40 to s_RV43	Reserved Value 40 to 43 **NOTE:** Reserved for e-Security's use.	Rv40 to Rv43 (rv40 to rv43)
s_RV44	DestinationThreatLevel	Rv44 (rv44)
s_RV45	DestinationUserContext	Rv45 (rv45)
s_RV46	VirusStatus	Rv46 (rv46)
s_RV47	DestinationFunction	Rv47 (rv47)
s_RV48	DestinationOperationalContext	Rv48 (rv48)
s_RV49	ReservedVar49 **NOTE:** Reserved for e-Security's use.	Rv49 (rv49)
s_RV50	eSecTaxonomyLevel1	Rv50 (rv50)
s_RV51	eSecTaxonomyLevel2	Rv51 (rv51)
s_RV52	eSecTaxonomyLevel3	Rv52 (rv52)
s_RV53	eSecTaxonomyLevel4	Rv53 (rv53)
s_RV54 to s_RV100	Reserved Value 54 to 100 **NOTE:** Reserved for e-Security's use.	Rv54 to Rv100 (rv54 to rv100)

Auto-formatting

Reserved variables s_DP, s_SP and s_P are set to lowercase before the event message is sent. The reserved variables s_ST and s_PN are set to uppercase before the event message is sent. The event time variable's s_ET is set if left clear with the standard time format as follows:

```
s_Year-s_Month-s_Day~sHour:s_Min:s_Sec~s_AMPM24~s_TZ
```

You may override this feature by setting the s_ET variable with other information. At a minimum, both s_Hour and s_Month must be set for the ET to be created. All empty fields will appear in the ET field as NULL.

Date/Time Reserved Variables

The ET meta-tag s_ET variable is automatically populated if s_ET is left clear and s_Hour and s_Month are not empty. The date/time reserved variables should be set with values. Any empty field will show up as NULL. The s_Day field is formatted to two-digit values 01-09. The script writer may choose to convert the month value into a two-digit number using the TRANSLATE command and the months.csv file. The date/time reserved tags are as follows:

- s_Year
- s_Month
- s_Day
- s_Hour
- s_Min
- s_Sec
- s_TZ
- s_AMPM24

Event Control Reserved Variables

Two variables, s_SendEITag and s_SendETTag are used to determine whether the EVENT command will include the EI and ET fields, respectively, in an alert message. To disable the sending of either field, the variables must be set to OFF.

Format

```
EVENT ()
```

For example:

```
COPY(s_Res:"Resource")
SET(i_Severity = 3)
COPY(s_BM:"Alert")
EVENT()
```

EXPORTVAR

The EXPORTVAR command allows you to export a global or local script variable to an agent. From there, it may be read by SNMPGET commands and set by SNMPSET commands. Each variable that is exported will have its own SNMP Object Identifier (OID). The OID has the following structure:

```
.1.3.6.1.4.1.4286.1.5.u.v.w.x
```

Where,

- u will be the port number (set by the Agent Manager)
- v will be the port name
- w will be an index to the variable number
- x will be the value of the variable

Exporting variables in this way results in an OID that can be logically evaluated by the SNMPSET command and SNMPGET command.

Format

```
EXPORTVAR(var1)
```

Data Types

Argument	Type	Description
var1	variable (INPUT/ OUTPUT)	The variable to export (fvar, ivar or svar).

For example:

```
EXPORTVAR(ivar1)
SNMPSET("i","192.168.100.2", "public","svar1",
   "1","istatus")
```

FILEA

The FILEA command appends the contents of a string to the end of a flat file on disk. When using this command:

- Specify the filename using a string
- For Windows, the filename references the file as specified if the filename starts with a drive letter, colon and backslash (such as c:\)
- The full path of the file should be specified
- If the file does not exist, it is created
- If the file cannot be created, the FILEA command does nothing
- The file closes after the data has been appended to it

If you are writing this command as part of a script to be executed by an Agent, be sure to use the proper path syntax, including forward slashes (/). Remember to escape back slash and forward slash characters when specifying the path. The terminating zero on the end of the string is not written to the file.

Format

```
FILEA("filename", data)
```

Data Types

Argument	Type	Description
filename	string (INPUT)	The name of the file to which the data should be applied.
data	string (INPUT)	The data string to append to the file.

For example:

In the following example, the file \temp\mux_data is created and the contents of s_variable are added to the file:

```
FILEA("c:/\temp/\mux_data", s_variable)
FILEA("mux_data", "literal")
FILEA("mux_data", s_variable)
```

In the following example, a string is added to the end of an audit log file:

```
COPY(audit_str: "Sent 20 severity 5 alerts.")
FILEA("h:/\temp/\audit.log", audit_str)
```

FILEL

The FILEL command gets the length (in bytes) of a flat file and places the value into a numeric variable. When using this command:

- Specify the filename using a string
- For Windows, the filename references the file as specified if the filename starts with a drive letter, colon and backslash (such as c:\)
- If the file does not exist, the FILEL command does nothing and the contents of numvar are unchanged
- The file closes after the data has been read from it

If you are writing this command as part of a script to be executed by an Agent, be sure to use the proper path syntax, including forward slashes (/). Remember to escape back slash and forward slash characters when specifying the path.

Format

```
FILEL("filename", i_length)
```

Data Types

Argument	Type	Description
filename	string (INPUT)	The name of the file whose length is to be determined.
i_length	numvar (OUTPUT)	The length of the file, in bytes.

For example:

```
FILEL("h:/\tmp/\onfotron.log", i_length)
```

Returns the length of the infotron.log file, in bytes, for example:

```
i_length = 2390
```

FILER

The FILER command copies the contents of a flat file on disk into a string variable. When using this command:

- Specify the filename using a string.
- For Windows, the filename references the file as specified if the filename starts with a drive letter, colon and backslash (such as c:\)
- If the file does not exist, the FILER command does nothing and the contents of svar are unchanged
- The file closes after the data has been read from it

- Optionally, enter the maximum number of bytes to read. You cannot use the max_bytes parameter unless it is paired with the i_offset parameter.

If you are writing this command as part of a script to be executed by an Agent, be sure to use the proper path syntax, including forward slashes (/). Remember to escape back slash and forward slash characters when specifying the path.

Format

```
FILER("filename", dest, [i_offset [, i_max_bytes]])
```

> **NOTE**: You cannot use the max_bytes parameter unless it is paired with the i_offset parameter.

Data Types

Argument	Type	Description
filename	string (INPUT)	The name of the file to read the data string.
data	svar (OUTPUT)	The data read from the file is placed into this string variable.
i_offset	integer (INPUT) [OPTIONAL]	Specifies an offset number of characters at which to begin reading.
max_bytes	integer (INPUT) [OPTIONAL]	Optionally, specify the maximum number of bytes to read. **NOTE:** When using this argument, the i_offset argument must be specified.

For example:

```
CLEAR(data)
FILER("filename", data, 0, 20)
if(data = "")
ALERT(s_res_var, "Data file doesn’t exist or is
   empty.", 0)
ENDIF()
```

FILEW

💾

The FILEW command writes the contents of a string to a flat file on disk. When using this command:

- The previous contents of the file are overwritten
- Specify the filename using a string
- For Windows, the filename references the file as specified if the filename starts with a drive letter, colon and backslash (such as c:\)
- If the file does not exist, it is created

- If the file cannot be created, the FILEW command does nothing
- The file closes after the data is written to it

If you are writing this command as part of a script to be executed by an Agent, be sure to use the proper path syntax, including forward slashes (/). Remember to escape back slash and forward slash characters when specifying the path.

Format

```
FILEW("filename", data)
```

Data Types

Argument	Type	Description
filename	string (INPUT)	The name of the file to write the data string.
data	svar (OUTPUT)	The data to write to the file.

For example:

```
FILEW("filename", data)
FILEW("h:/\tmp/\infotron.stat", "SUCCESSFUL EXEC")
```

FOR

For

The FOR command provides capability for looping control flow. When using this command:

- The initialization statement is always executed
- If the result of the FOR() compare statement is true, the parsing commands after the FOR(), up to the next ENDFOR() are executed. The incrementation statement is then executed and control flow returns to the compare statement
- If the result of the FOR() compare is false, no parsing commands are executed between the FOR() and the ENDFOR(). The incrementation statement is not executed
- Although all data types are allowed on each side of the for() compare statement, only numeric values can be compared with numeric and string with string
- The operator for the FOR() compare can be <, =, >, <=, >=, <>, &, + or ^

You cannot directly compare against a negative number. Use one of the following methods to do this:

- Use the parsing function COMPARE
- Indirectly compare as follows:

```
SET(i_compare_val=-10)
FOR(ivar=0, ivar>i_compare_val, ivar=ivar-1)
ALERT("Still in loop")
ENDFOR()
```

Format

```
FOR(initialization, compare, increment)
```

Data Types

Argument	Type	Description
initialization	SET() parameter	Any valid parameter that can be passed to the SET() command. See SET() command definition.
conditional	IF() conditional	Any valid parameter that can be passed to the IF() command. See IF() command definition.
increment	SET() parameter	Any valid parameter that can be passed to the SET() command. See SET() command definition.

For example:

```
FOR(i=0, i<3, i=i+1)
```

GETCONFIG

Retrieves the current setting for a system property. This command is used to retrieve system properties set using the SETCONFIG command. These commands are used to set variables and retrieve current values for system properties that may change periodically, for example a log file that is renamed daily using the current date.

Available system properties are:

System Property	Example(s)
▪ System.OS.Family	Solaris and Windows
▪ System.OS.Name	Windows 2000
▪ System.OS.Version.Major	5
▪ System.OS.Version.Minor	0
▪ System.Net.Hostname	ESECServer
▪ System.Net.IP_List	list of IP's for this host separated by a semicolon, an example is "172.163.3.45;172.45.2.1"

See also SETCONFIG command.

There are two required parameters.

- The first required parameter defines the configuration option (FileConnector.InputFile) or (FileConnector.OutputFile).
- The second required parameter defines the configuration value to retrieve.

Format

```
GETCONFIG(Config Option, Value)
```

Data Types

Argument	Type	Description
Config Option	string (INPUT)	Name of the configuration variable to retrieve. Input file = “FileConnector.InputFile” Output file = “FileConnector.OutputFile”
Value	string (INPUT)	Configuration setting to retrieve.

For example:

```
GETCONFIG(“FileConnector.InputFile”, s_inputfilename)
GETCONFIG(“FileConnector.OutputFile”,
   s_outputfilename)
```

Current Output Variables’ Contents

```
“C:/\filename.txt”
```

GETENV

The GETENV command retrieves the value of an environment variable.

Format

```
GETENV(Environment Key, Variable to store value)
```

Data Type

Argument	Type	Description
Environment Key	string (INPUT)	Name of the environment variable.
Variable to store value	string Var (INPUT)	Destination of where the environment variable will be placed.

For example:

```
GETENV("ESEC_HOME", s_EsecHome)
```

HEXTONUM

The HEXTONUM command converts a hex string with up to 4 bytes of hex data into a decimal number and places the decimal number in an integer or a float variable. More than 4 bytes results in invalid data.

Format

```
HEXTONUM(bytes_data, i_val [,[-]i_4] [, ioffset])
```

Data Types

Argument	Type	Description
bytes_data	string (INPUT)	String of 1 to 4 bytes. (for example: “\FF\”, “\FF FF\”, “\3C 4A F2\”, “\43 76 F3 FF\”, or “TEST”). The hex number represented by these bytes will be converted into an integer value, i_val.
i_val	numvar (OUTPUT)	Decimal equivalent of hex number is placed in this variable, ivar or fvar.
i_len	numeric (INPUT) [OPTIONAL]	Number of hex bytes to convert to an integer (must have an absolute value range of 1 - 4). If you don’t set this parameter, the default value is the number of bytes in the input string, bytes_data, up to 4 bytes. If i_len is positive, then bytes are interpreted as Left-To-Right (Most-Significant-Byte to Least-Significant-Byte). If i_num_bytes is negative, then bytes are interpreted as Right-To-Left (Least-Significant-Byte to Most-Significant-Byte).
ioffset	numeric (INPUT) [OPTIONAL]	Offset number of bytes to skip in bytes_data.

For example:

In the following example, the data in the hex string “\5A32\” is converted to an integer value, interpreted MSB to LSB and then from LSB to MSB.

```
COPY(data:"\5A 32\")
HEXTONUM(data, ivar1)
HEXTONUM(data, ivar2, -2)
```

> **NOTE:** For hex substitution, \0000\ terminates a string; therefore, “xxxx\0000\yyyy” becomes “xxxx”.

Current Output Variables’ Contents:

```
ivar1 = 23090
ivar2 = 12890
```

IF

The IF command compares two values.

- If the result of the IF() statement is true, the parsing commands after the IF(), up to the next ELSE() or ENDIF(), are executed.
- If the result of the IF() is false, the parsing commands following the ELSE() up to ENDIF() are executed.
- If no ELSE() is used, no parsing commands are executed between the IF() and ENDIF() when the result of the IF() statement is false.
- Although all data types are allowed on each side of the IF() statement, only numeric values can be compared with numeric and string with string.
- The operator for the IF() compare can be <, =, >, <=, >=, <>, &, + or ^. Do not use the logical NOT operator (^) in conjunction with a string variable. Doing so will generate a syntax error.

You cannot directly compare against a negative number. Use one of the following methods to do this:

- Use the parsing function COMPARE.
- Indirectly compare as follows:

```
SET(i_compare_val=-10)
IF(ivar > i_compare_val)
ALERT("ivar is greater than -10")
ENDIF()
```

Format

```
IF(<expr>)
Where:
expr ::= var
        | (<expr>)
        | ^ <expr>
```

Where <expr> must evaluate to integer or float.

```
        | <expr> <|=|>|<=|>=|<>|&|+ <expr>
```

Where both <expr> must evaluate to same type.

Data Types

Argument	Type	Description
data1	variable (INPUT)	The data to compare to data2. If data2 is not used, then it becomes a logical (0 = false, anything else = true).
logical operator	< = > <= >= <> & + ^	Less Than Equal To Greater Than Less Than or Equal To Greater Than or Equal To Not Equal To Logical AND Logical OR Logical NOT
data2	all	The data to compare to data1. This must be the same type is data1.

Argument	Type	Description
	(INPUT) [OPTIONAL]	
…	same as above	Use up to 200 individual parameters to create complex logical expressions.

For example:

```
IF(s = "test" & i_count < 5)
script(test)
ELSE()
IF((i <= i_num) + (i_count <> 10) &
   (i_page))page("111")
ENDIF()
ENDIF()
```

INC

The INC command increments a numeric variable by 1. When using this command, you must specify either an integer variable or a floating variable.

Format

```
INC(i_counter)
```

Data Types

Argument	Type	Description
i_counter	numvar (INPUT/ OUTPUT)	The numeric variable to be incremented by 1.

For example;

```
SET(icounter = 0)
INC(icounter)
INC(icounter)
```

Result:

```
icounter = 2
```

INDICATOR

The INDICATOR command sends indicator messages to Sentinel. The messages contain text that is to be displayed on the specified indicator in Sentinel.

Format

```
INDICATOR(name, value)
```

> **NOTE:** Prior to v4.0, the INDICATOR command had additional arguments that are no longer used. For compatibility with older agents, these arguments are labeled "Not Used" in the Wizard Command Editor window.

Data Types

Argument	Type	Description
name	string (INPUT)	Indicator name.
value	string (INPUT)	Indicator text to be displayed in the Sentinel Console. For example: `PRINTER ON`

For example:

```
INDICATOR("memory", "5 MB")
INDICATOR(name, value)
```

> **NOTE:** The indicator name in the parsing command must match the indicator name in Sentinel; if it does not, the indicator will not be updated in the Sentinel Console.

INFO_CLEARTAGS

ℹ

This function will zero out (or clear, in the case of strings) all variables that are part of the info block set referred to by the handle. Use INFO_CONSTANTTAGS to prevent this from happening to a subset of those tags.

Format

```
INFO_CLEARTAGS(<IN handle>)
```

Data Types

Argument	Type	Description
IN handle	string (INPUT)	type of information block

INFO_CLOSE

ℹ

This command is used to close an infoblock session. When called, it will first send any unsent infoblocks just as the INFO_SEND command would. It will then send an infoblock session close message by setting the EOD (End Of Data) attribute of the infos element to "true". After sending the close message, the segment number ("segnum") is incremented by one.

Format

```
INFO_CLOSE(<IN handle>)
```

Data Types

Argument	Type	Description
IN handle	string (INPUT)	type of information block

INFO_CONSTANTTAGS

ℹ

Use this command to name tags that will not be cleared out when INFO_CLEARTAGS has been called. Pass in zero or more tag names to create the set of constant tags. Multiple calls to this function will reset the list of constant tags.

Format

```
INFO_CONSTANTTAGS(<IN handle>, [<IN tag name>, …])
```

Data Types

Argument	Type	Description
IN handle	string (INPUT)	type of information block
IN tag name	string (INPUT)	name to refer to IN handle

INFO_CREATE

ℹ

This will create a new information block set. You must pass a handle (which you will use in every other command to affect this informational block set). You must also pass a type. This is a string of your choosing, but it should be formalized (see INFO_SEND).

If you call INFO_CREATE on an already existing handle, it will clear the contents at that handle as though you had begun a new handle. You will need to call INFO_SETTAG and INFO_CONSTANTTAGS again.

Format

```
INFO_CREATE(<OUT handle>,<IN type>)
```

Data Types

Argument	Type	Description
OUT handle	string (OUTPUT)	name to refer to IN type
IN type	string (INPUT)	type of information block

INFO_DUMP

ℹ

This command will persist the current state of the info block set into a string variable. This was included to facilitate testing, but can also be used to play back information block sets, or save them to a text file or other type file of choice. It also lacks the side effect the INFO_SEND has in that it does not clear out the current state.

Format

```
INFO_DUMP(<IN handle>, <OUT string-variable>)
```

Data Types

Argument	Type	Description
IN handle	string (INPUT)	type of information block
OUT string-variable	string (OUTPUT)	string variable to refer to IN handle

INFO_PUSH

ℹ

This will tag the current values of all tag names (via their associated variables) and push them onto the end of a list of info blocks referred to by a handle. Blocks will continue to accumulate in the set until emptied by calling INFO_CREATE, INFO_SEND or INFO_CLOSE. For INFO_CREATE, no action is taken. For INFO_SEND, the info blocks are sent to agentmanager. For INFO_CLOSE, the info blocks are sent to agentmanager and an info block close (EndOfData or EOD) message is sent.

Format

```
INFO_PUSH(<IN handle>)
```

Data Types

Argument	Type	Description
IN handle	string (INPUT)	type of information block

INFO_SEND

ℹ

This takes the current set of info blocks and sends them out on a communication channel specified by the type that was used during INFO_CREATE, appended to the word “infoblock.”, including the period. So if the type were “vulnerability”, then the channel name that the message would be sent on would be named “infoblock.vulnerability”.

In addition, this command will clear out the current set of info blocks and increment the segment number (“segnum”) by one.

Format

```
INFO_SEND(<IN handle>)
```

Data Types

Argument	Type	Description
IN handle	string (INPUT)	type of information block

INFO_SETTAG

ℹ

This command will bind a script variable to a name of an attribute. When INFO_PUSH is called (see INFO_PUSH), all variables that were bound with this command will be set as attributes in a block entry.

Format

```
INFO_SETTAG(<IN handle, IN tag name, IN variable)
```

Data Types

Argument	Type	Description
IN handle	string (INPUT)	type of information block
IN tag name	string (INPUT)	type of tag name
IN variable	string (INPUT)	type of variable

Vulnerability Info Block Tags

The following are valid vulnerability Info Block tags for the INFO_SETTAG command. The tags marked as required must be set in order for the info block to be stored as a vulnerability. Even if the info block is not stored as a vulnerability, the tags marked as constant will still be extracted from the info block. If a tag is set that is not in the following list, the vulnerability back end will ignore the tag.

Tag Name	Explanation	Type	Constant	Required
ScannerInstance	The name the user gives to this scanner instance. Usually set in the agent parameters.	String	X	
ProductName	Name of the scanner.	String	X	
ProductVersion	Version of the scanner	String	X	
ScannerType	The type of scanner.	String	X	

Tag Name	Explanation	Type	Constant	Required
Vendor	The scanner vendor name.	String	X	
ScanType	PARTIAL or FULL	String	X	
ScanStartDate	The time the scan started	String		
ScanEndDate	The time the scan ended	String		
IP	The IP of the resource	String		X
HostName	The hostname of the resource	String		
Location	The location of the resource	String		
Department	The department of the resource	String		
BusinessSystem	The business system of the resource	String		
OperationalEnvironment	The operation environment of the resource	String		
Regulation	The regulation of the resource	String		
RegulationRating	The regulation rating of the resource	String		
Criticality	The criticality of the resource [1 – 25]	Number		
VulnModule	The module used to detect the vulnerability	String		
PortNumber	The port number of the vulnerability	Number		
PortName	The name of the port of the vulnerability	String		
NetworkProtocol	The network protocol of the vulnerability	Number		
ApplicationProtocol	The application protocol of the vulnerability	String		
AssignedVulnSeverity	The assigned vulnerability severity.	Number		
ComputedVulnSeverity	The computed vulnerability severity.	Number		

Tag Name	Explanation	Type	Constant	Required
VulnDescription	The vulnerability description.	String		
VulnSolution	The vulnerability solution.	String		
VulnSummary	The vulnerability solution.	String		
VulnCrossRefs	A list of codes for the vulnerability.	String		
DetectedOs	The operating system detected when discovering the vulnerability	String		
DetectedOsVersion	The operating system version detected when discovering the vulnerability.	String		
ScannedApp	The application detected when discovering the vulnerability	String		
ScannedAppVersion	The application version detected when discovering the vulnerability	String		
VulnUserName	The vulnerability username.	String		
VulnUserDomain	The domain of the vulnerability user.	String		
VulnTaxonomy	The taxonomy of the vulnerability.	String		
ScannerClassification	The vulnerability classification given by the scanner.	String		
ExtendedInformation	Extended information to store along with this vulnerability	String		
VulnName	The name of the vulnerability given by the scanner.	String		

INFO_* Command Example

e-Security batches vulnerability scans into smaller chunks (info block sessions) that can be more easily processed. An info block session contains multiple info block sets, each with an increasing segment number (“segnum”) followed by an info block session close message. An instance of an info block session is referred to by its globally unique “id”. Each time INFO_SEND is called, an info block set with the currently “pushed” values and the current segment number (“segnum”) will be sent. Immediately after the info block set is sent, the segnum will be incremented by one. The INFO_SEND is called for each batch of data, after which the INFO_CLOSE command is called to close the info block session. The info block close message consists of an info block set with the attribute EOD set to “true”.

For example:

```
INFO_CREATE(h_vuln,"vulnerability")
INFO_SETTAG(h_vuln,"ALPHA", s_alpha)
INFO_SETTAG(h_vuln,"BETA", i_beta)
INFO_SETTAG(h_vuln,"GAMMA", s_gamma)
INFO_SETTAG(h_vuln,"DELTA", i_delta)
INFO_SETTAG(h_vuln,"^1E*P$S I(L)O.N--", f_epsilon)
INFO_CONSTANTTAGS(h_vuln,"GAMMA","DELTA","^1E*P$S
   I(L)O.N--")
SET(i_beta=12345)
SET(i_delta=123456789)
SET(f_epsilon=1.234)
COPY(s_alpha:"a is for apple")
COPY(s_gamma:"c is for coffee")
INFO_PUSH(h_vuln)
INFO_CLEARTAGS(h_vuln)
INFO_PUSH(h_vuln)
INFO_DUMP(h_vuln, s_simulate)
INFO_SEND(h_vuln)
SET(i_beta=6789)
SET(i_delta=987654321)
SET(f_epsilon=3.1415926)
COPY(s_alpha:"a is for acorn")
COPY(s_gamma:"c is for carrot")
INFO_PUSH(h_vuln)
INFO_SEND(h_vuln)
INFO_CLOSE(h_vuln)
```

Results:

```
<?xml version="1.0" encoding="UTF-8"?>
<infos id="B008961E00CB1026B8F000065BBD13AB"
   type="vulnerability" segnum=”0” version=”4.2.0.0”
   EOD=”false”>
<info ALPHA="a is for apple" BETA="12345"
   DELTA="123456789" GAMMA="c is for coffee"
   _1EPSILON="1.234"/>
```

```
<info ALPHA="" BETA="0" DELTA="123456789" GAMMA="c is
   for coffee" _1EPSILON="1.234"/>
</infos>
<?xml version="1.0" encoding="UTF-8"?>
<infos id="B008961E00CB1026B8F000065BBD13AB"
   type="vulnerability" segnum=”1” version=”4.2.0.0”
   EOD=”false”>
<info ALPHA="a is for acorn" BETA="6789"
   DELTA="987654321" GAMMA="c is for carrot"
   _1EPSILON="3.1415926"/>
</infos>
<?xml version="1.0" encoding="UTF-8"?>
<infos id="B008961E00CB1026B8F000065BBD13AB"
   type="vulnerability" segnum=”2” version=”4.2.0.0”
   EOD=”true”>
</infos>
```

IPTONUM

The IPTONUM command converts a string representation of IPv4 address into an integer number and places the integer number in an integer variable. This function only supports IPv4 addresses. An IPv4 address that does not fall in the valid range results in invalid data.

Format

```
IPTONUM(ip_address, i_integer, i_valid)
```

Data Types

Argument	Type	Description
ip_address	svar(INPUT)	String IPv4 address.
i_integer	numeric(OUTPUT)	String IPv4 address is converted into an integer value. The integer value is placed in this variable.
i_invalid	ivar(OUTPUT) [OPTIONAL}	Value of 0 implies the IP is invalid. Valid of 1 implies the IP is valid.

For example:

In the following example, the IPv4 address “10.10.10.255” is converted to an integer number. i_valid is set to 1, which implies the result is valid.

```
IPTONUM("10.10.10.255", i_y, i_valid)
```

Current Output Variable’s Contents:

```
i_y = 168430335
i_valid = 1
```

In the following example, the invalid IPv4 address “10.10.10.258” is converted to an integer number 0. i_valid is set to 0, which implies the result is invalid.

```
IPTONUM("10.10.10.258", i_y, i_valid)
```

Current Output Variable’s Contents:

```
i_y = 0
i_valid = 0
```

The NUMTOIP command converts a number to an IP. See NUMTOIP for more information.

LENGTH or LENGTH-OPTION2

The LENGTH command sets a numeric variable from the length in bytes of a string variable (not counting the terminating zero).

> **NOTE:** Within the Visual Editor of the Agent Builder, LENGTH and LENGTH-OPTION2 are listed as separate commands. They are same command. They are provided as descriptions for different variations of the same command. If you were to use LENGTH-OPTION2 in the text editor, you would enter LENGTH.

Format

```
LENGTH(i_length, s_variable)
```

Data Types

Argument	Type	Description
s_variable	string (INPUT)	The string (usually string variable) in which the length is computed.
i_length	numvar (OUTPUT)	The length of the string variable, s_variable, is placed in this numeric variable.

For example:

```
LENGTH(i_length, source)
LENGTH(i_num_bytes, "It makes no sense to do this, as
   we know the string whose length we are checking")
```

Results:

```
i_num_bytes = 80
```

LOOKUP

The LOOKUP command matches data found in the receive buffer or in a string with key strings found in a specified lookup key file.

If a record is found that matches the data byte for byte, the parsing commands in the lookup key file record are processed.

If a string is specified as the first parameter in the LOOKUP command, the LOOKUP command uses that string when searching the lookup key file.

There are five arguments or parameters with this command.

- compare - If a numeric value is specified as this parameter, that number of bytes (the numeric value) of data from the receive buffer, starting at the Rx buffer pointer position, is used as the string when comparing to the lookup key file key strings.
- lookup name – This parameter specifies the lookup key file name relative to the WORKBENCH_HOME directory.
- imatch - An optional integer variable that may be specified that returns the status of the LOOKUP command. (0=no match found, 1=found match).
- parameter file - An optional parameter that is the name of a parameter file to use other than the default parameter file. The default parameter file name is <Agent>.par. This filename should not include the .par suffix.
- column name - An optional parameter is the column with the parameter file to use for lookup values. The default column name is the template name. If you specify this parameter, you must also use a parameter filename.

Format

```
LOOKUP(compare, lookup filename [, imatch] [,
   [parameter filename] [, column name]])
```

Data Types

Argument	Type	Description
compare	string (INPUT) or numeric (INPUT)	The data to be used to compare against the fields in the lookup key file. This is a byte-by-byte comparison. The number of bytes from the receive buffer, using the current Rx buffer pointer position, to use to compare against the fields in the lookup key file. This is a byte-by-byte comparison. **NOTE:** This will only work if rxbuff was used to set the receive buffer.
lookup filename	string (INPUT)	The lookup key file name
imatch	numvar (OUTPUT) [OPTIONAL]	A match was found. 0=No 1=Yes
parameter filename	string (INPUT)	The parameter filename. Default: agent.par
column name	string (INPUT)	The column within the parameter file to use. Default: `agent name`

For example:

```
LOOKUP(data, filename, imatch)
```

In the following example, the key_01 filename is determined from the name put in the parameter file, not the lookup key filename.

```
LOOKUP(s_variable, {key_01})
LOOKUP(s_variable, {key_01}, imatch, "Send One Alert",
  "GeoElements")
```

If any parameter definitions are in the lookup file, look for them in the GeoElements column of the Send One Alert parameter file.

NEGSEARCH

The NEGSEARCH command performs a backwards search for a string in the receive buffer. There are two parameters with this command.

- search - The search begins at the current Rx buffer pointer position and continues backwards until it finds the string or until it reaches the beginning of the receive buffer. If the search finds the string, the Rx buffer pointer updates to point to the first byte of the search string. If the search does not find the string, the Rx buffer pointer is unchanged.
- ifound - An optional parameter, it is an integer variable that is set to 1 if the search finds the string and is set to zero if the search does not find the string.

Format

```
NEGSEARCH(search[, ifound])
```

Data Types

Argument	Type	Description
search	string (INPUT)	The searched string in the receive buffer, starting with the current Rx buffer pointer position and searching backwards.
ifound	numvar (OUTPUT) (OPTIONAL)	Returns whether or not the search string was found. 0=not found 1=found

For example:

```
NEGSEARCH("MINOR ALARM")
NEGSEARCH(search_string)
```

The following examples search for a carriage-return and a line-feed:

```
NEGSEARCH("\0d0a\")
NEGSEARCH(data, ifound)
```

Another example:

The underscored letter represents the current Rx buffer pointer position in the example.

NOTE: For hex substitution, \0000\ terminates a string; therefore, “xxxx\0000\yyyy” becomes “xxxx”.

```
Rx Buffer = "Minor Alarm Radio A"
NEGSEARCH("Ala")
```

Result:

```
Rx Buffer = "Minor Alarm Radio A"
```

NUMTOHEX

The NUMTOHEX command converts a numeric number to hex data and places those hex bytes (up to 4 bytes) in a string.

Format

```
NUMTOHEX(i_decimal, hex_data)
```

Data Types

Argument	Type	Description
i_decimal	numeric (INPUT)	Integer value to translate into hex data.
hex_data	svar (OUTPUT)	String of 1 to 4 bytes that are the hex byte(s) given by the numeric value, i_decimal.

For example:

In the following example, the decimal number 16777215 is converted to hex data.

```
SET(i_decimal = 16777215)
NUMTOHEX(i_decimal, shex)
```

Current Output Variable’s Contents:

```
shex = "\ff ff ff\"
```

NUMTOIP

The NUMTOIP command converts a numeric number to an IPv4 address, and places the IP address in a string.

Format

```
NUMTOIP(i_integer, ip_address)
```

Data Types

Argument	Type	Description
i_integer	numeric(INPUT)	Integer value to translate into IPv4 address.
ip_address	svar(OUTPUT)	String IPv4 address

For example:

In the following example, the decimal number 16777215 is converted to IPv4 address.

```
SET(i_integer = 167772161)
NUMTOIP(i_integer, s)
```

Current Output Variable’s Contents:

```
s = "10.0.0.1"
```

The IPTONUM command converts an IP to a number. See IPTONUM for more information.

PARSER_ATTACHVARIABLE

The PARSER_ATTACHVARIABLE command allows the name of a name-value pair to be associated with a target_variable.

In most cases, suggest that you create a parser and attach a variable in the initialization state outside of the loop. Then you can reuse that parser by using it in the parsing loop.

For related parsing commands, see PARSER_CREATEBASIC command and PARSER_PARSESTRING command.

NVP (Name-value Pair) Parser

The following fragment of code demonstrates the NVP parser:

```
PARSER_CREATEBASIC (h_nvp, "nvp", "separator==",
   "entry_separator= ", "value_quotes=/"",
   value_quotes_optional=yes")
PARSER_ATTACHVARIABLE (h_nvp,"this",s_this)
PARSER_ATTACHVARIABLE (h_nvp,"me",s_me)
PARSER_ATTACHVARIABLE (h_nvp,"hello",s_hello)
PARSER_PARSESTRING (h_nvp, "this=/”that/” me=/"you =
   them/" hello=/”goodbye/”")
```

Parameters

The following parameters are recognized when they appear in the following format:

```
"<parameter>=<value>"
```

<parameter> is one of the items below and <value> is an appropriate value for that parameter.

- separator - the character you use to separate the name from the value
- entry_separator - the character you use to separate one name-value pair from the next
- name_quotes - the character you use to enclose the name (“ or ‘, for instance)
- value_quotes - the character you use to enclose the value

- name_quoted - set to yes to make the NVP paser observe the name_quotes option
- value_quoted - set to yes to make the NVP parser observe the value_quotes option
- name_quotes_optional - set to yes to allow option quotes on the name. If this is yes and quotes are omitted, then optional whitespace followed by the separator will terminate the name.
- value_quotes_optional - set to yes to allow option quotes on the name

If this is yes and quotes are omitted, optional whitespace followed by the entry_separator will terminate the value.

Format

```
PARSER_ATTACHVARIABLE(<parser_handle>, <name>,
   <target_variable>)
```

Data Types

Argument	Type	Description
parser_handle	string variable (INPUT)	The handle variable of a created parser.
name	string (INPUT)	The name of a name-value pair.
target_variable	any variable (OUTPUT)	The variable that will be set with the value associated with the name of a name-value pair.

The following is Checkpoint Parser example.

```
AGENT SETUP STATE:
PARSER_CREATEBASIC(h_nvp,"nvp", "separator==",
   "entry_separator= ", "value_quotes=/"",
   "value_quotes_optional=yes")
PARSER_ATTACHVARIABLE(h_nvp,"action", s_EVT)
PARSER_ATTACHVARIABLE(h_nvp,"d_port", s_DP)
PARSER_ATTACHVARIABLE(h_nvp,"proto", s_P)
PARSER_ATTACHVARIABLE(h_nvp,"src", s_SIP)
PARSER_ATTACHVARIABLE(h_nvp,"dst", s_DIP

PARSE STATE:
PARSER_PARSESTRING(h_nvp,s_RXBufferString)
```

PARSER_CREATEBASIC

The PARSER_CREATEBASIC command defines a parser and associates it with a parser_handle. For more information, see NVP (Name-value Pair) Parser under PARSER_ATTACHVARIABLE.

In most cases, suggest that you create a parser and attach a variable in the initialization state outside of the loop. Then you can reuse that parser by using it in the parsing loop.

For another related parsing command, see PARSER_PARSESTRING command.

Format

```
PARSER_CREATEBASIC(<parser_handle>, <parser_name>, [,
  <nvp> [, ...]])
```

Data Types

Argument	Type	Description
parser_handle	string variable (OUTPUT)	The variable with which you will refer to this parser from this point forward.
parser_name	string (INPUT)	The string name of the simple parser you are creating. **NOTE:** At this time, only nvp is recognized.
nvp	string (INPUT) (OPTIONAL)	The name-value pair. Zero or more strings that contain a property name, followed by an equal sign, followed by a value. The parameters that are recognized are determined by the parser_name that was chosen. **NOTE**: When the parser name is set to nvp, you must use the following arguments: "separator==" "entry_separator= " "value_quotes=/"" "value_quotes_optional=yes"
nvp1	string (INPUT) (OPTIONAL)	Name-value pair 1.
nvp2	string (INPUT) (OPTIONAL)	Name-value pair 2.
…	string (INPUT) (OPTIONAL)	Other name-value pairs.

For an example, see Checkpoint Parser example under PARSER_ATTACHVARIABLE, Data type.

PARSER_NEXT

The PARSER_NEXT command advances the parser to the next position in the parse string filling out the variables set by the command PARSER_ATTACHVARIABLE.

Format

```
PARSER_NEXT(<parser_handle>, <success_flag>)
```

Data Type

Argument	Type	Description
parser_handle	string variable (INPUT)	The handle variable of a created parser.
success_flag	numvar (INPUT)	0: unsuccessful parse 1: a successful parse

PARSER_PARSESTRING

The PARSER_PARSESTRING command will process the string_to_parse using the created parser referenced by the parser_handle. This allows you to construct any arbitrary string for parsing, rather than insist upon a stream source or the Rx Buffer.

For more information, see PARSER_ATTACHVARIABLE command and PARSER_CREATEBASIC command.

The reserved variable s_RXBufferString may be used as a string_to_parse after the Receive State to parse the script input. For more information, see NVP (Name-value Pair) Parser under PARSER_ATTACHVARIABLE.

Format

```
PARSER_PARSESTRING(<parser_handle>, <string_to_parse>)
```

Data Types

Argument	Type	Description
parser_handle	string variable (INPUT)	The handle variable of a created parser.
string_to_parse	string (INPUT)	The single string that will be run through this parser.

For an example, see Checkpoint Parser example under PARSER_ATTACHVARIABLE, Data type.

PAUSE

The PAUSE command causes the current script to immediately pause “n” number of seconds. The PAUSE command works between instructions in a parsing state and between states. The PAUSE command is useful in setting polling cycle times or to ensure you don’t poll too quickly (such as in polling a database log).

You may specify several PAUSE commands during parsing.

Format

```
PAUSE(iseconds)
```

Argument	**Type**	**Description**
iseconds	numeric (INPUT)	Number of seconds to pause before going to the next state.

For example:

```
PAUSE(10)
   PAUSE(iseconds)
```

Or

```
IF(slowing=true)
   pause(50)
ENDIF( )
```

POPUP

The POPUP command displays the contents of a string to a screen in a scrollable text window.

Format

```
POPUP(data [, title])
```

Data Types

Argument	**Type**	**Description**
data	string (INPUT)	The data string message to place in the popup window.
title	string (INPUT) [OPTIONAL]	The string to use as the title of the popup window (default = “Popup DATA”).

For example:

```
POPUP(data)
POPUP("Hello World", "Title String")
POPUP(data, title)
```

PRINTF

The PRINTF command copies formatted data into a string variable (svar). The PRINTF command is an advanced parsing command. If you are new to the parsing command language, consider using the COPY command and the APPEND command until you are comfortable with the language.

When using this command:

- Specify a svar as the destination string.
- Specify a format string.
- Specify any optional additional parameters to scan based on the format string.

Format String

To use HEX data in the format string, use the following convention:

```
\HX HX HX\
```

If you want to include a line feed at the end of the format string, the format string must look like the following string:

```
Format String\0a\
```

The format string for a carriage return is \0d0a\, for example:

```
PRINTF(message,"Voltage is %lf \0d0a\",f_volts)
```

The format string for a tab is \09\, for example:

```
PRINTF(message,"Voltage = \09\ %lf",f_volts)
```

Format

```
PRINTF(dest, format [, <paramList>])
```

where:

```
<paramList> ::= var [, <paramList>]
```

Data Types

Argument	Type	Description
dest	svar (OUTPUT)	The destination string variable in which to place the formatted string.
format	string (INPUT)	The format of the string to copy into the destination string variable. Similar to the format of the C printf command; for example, “Looping %d in %s” (see % Characters for Output Format).
parm1	all (INPUT) [OPTIONAL]	All data types except array. Must match the format string.
parm2	all (INPUT) [OPTIONAL]	All data types except array. Must match the format string.
…	all (INPUT) [OPTIONAL]	All data types except array. Must match the format string.

Format

```
% Characters for Output Format
```

Character	Type	Output Format
%d	integer	Signed decimal integer.
%le	float	Signed value having the form [-]d.dddd e [sign]ddd ...where d is a single decimal digit, dddd is one or more decimal digits, ddd is exactly three decimal digits and sign is+ or -.

Character	Type	Output Format
%lf	float	Signed value having the form [-]dddd.dddd ...where dddd is one or more decimal digits. The number of digits before the decimal point depends on the magnitude of the number and the number of digits after the decimal point depends on the requested precision.
%lg	float	Signed value printed in f or e format, whichever is more compact for the given value and precision. The e format is used only when the exponent of the value is less than -4 or greater than or equal to the precision argument. Trailing zeroes are truncated and the decimal point appears only if one or more digits follow it.
%s	string	Print a string variable.

Displaying Digits of Precision

By default, the PRINTF command displays a floating point number to six digits of precision. The six digits of precision default also applies to double precision numbers.

To display additional digits of precision, specify a value for the precision field in the PRINTF() format specification:

```
%[<width>][.<precision>] type>
```

For example:

```
PRINTF(dest, "%2.3lf", fvar)
```

Would produce the output: 22.012, representing 2 positions to the left of the decimal point and 3 positions to the right of the decimal point.

The following examples show how to pass string and integer variables.

```
PRINTF(dest,format_string) PRINTF(mystring,
   "val of matrix[%d][%d] = %s",
   index_x, index_y, matrix[index_x][index_y])
   PRINTF(dest,"Looping %d in state %s",iloop,state)
   PRINTF(dest,"Formatted %s Data into
   %s","string","dest")
```

The following example shows how to pass a float variable to a string.

```
PRINTF(message,"Voltage is %lf",f_volts)
```

To print floating point numbers, use %lf or %le.

REGEXPREPLACE

The REGEXPREPLACE command searches and replaces strings, using regular expressions. When the search finds the string, it substitutes the regexpreplace string. The REGEXPREPLACE command does a global replace, not just a replace of the first occurrence.

Format

```
REGEXPREPLACE(dest_string, search, replace)
```

Data Types

Argument	**Type**	**Description**
dest_string	svar (INPUT/ OUTPUT)	The string variable that will have bytes replaced.
search	string (INPUT) or svar (INPUT/ OUTPUT)	The search string to replace.
replace	string (INPUT) Or svar (INPUT/ OUTPUT)	The replacement string; can be of zero length to indicate null string.

For example:

```
COPY(string:"The 1st time")
REGEXREPLACE(string, "1st", "2nd")
```

Result:

```
string = "The 2nd time"
```

NOTE: In this example, you may substitute a regular expression for the `“1st” string.`

To replace with null string

```
COPY(string:"The 1st time")
REGEXPREPLACE(string, "1st", "")
```

Result:

```
string="The time"
```

For more information on regular expressions and the portable character set, see Regular Expressions.

Sentinel uses a POSIX (Portable Operating System Interface for UNIX)-compliant library for regular expressions. POSIX is a set of IEEE and ISO standards that help assure compatibility between POSIX-compliant operating systems, which includes most varieties of UNIX.

REGEXPSEARCH, REGEXPSEARCH_EXPLICIT or REGEXPSEARCH_STRING

The REGEXPSEARCH command performs a forward search in the receive buffer (Rx Buffer) or designated input string variable for a string, using regular expressions. It also supports expression groups.

> **NOTE:** Within the Visual Editor of the Agent Builder, REGEXPSEARCH, REGEXPSEARCH_EXPLICIT or REGEXPSEARCH_STRING are listed as separate commands. They are same command. They are provided as descriptions for different variations of the same command. If you were to use REGEXPSEARCH_EXPLICIT or REGEXPSEARCH_STRING in the text editor, you would enter REGEXPSEARCH.

Receive Buffer

The search within the receive buffer goes as follows:

- The search begins at the current Rx buffer pointer position and continues searching forward until the search finds the string or until the search reaches the end of the receive buffer.
- If the search finds the string, the Rx buffer pointer updates to point to the first byte of the string for which it searched. This Rx buffer pointer position is retained when transitioning across states unless explicitly changed using the RESET command.
- If the search does not find the string, the Rx buffer pointer does not move.

When using this command to search the receive buffer, the optional second parameter is an integer variable that is set to 1 if the search finds the string and sets to 0 if the search does not find the string.

String Variable

String variables do not support the parse pointer, so dynamics when searching in a string variable are different. The regular expression pattern will either match some or all of the input string. If the regular express pattern is configured with expression groups, then input string content that matches the expression groups can be stored in output variables. There are two expression grouping output options. One is to populate the list of variables in order of the expression groups, and the other is to designate a string array.

If the regular expression successfully matches the input – string variable, a designated list of variables or output array is set with the group values and the found variable is set to one more than the number of groups or zero upon match failure.

When the output of the group values is to be a string array, the first element indexed with “0” will contain the match string. The match string will contain the content that matched the entire regular expression independent of expression groups. So, the first expression group’s content will be stored in the array position indexed with “1”. When looping through the output array, keep in mind the i_Found_Tokens value compensates for the first element being the match string by always being one more than the total number of groups. In a for loop, the stop condition of being less than the value i_Found_Tokens will still work, but you will likely start your index at “1” instead of “0”.

When designating the group values to be stored in a list of output variables instead of an array, the command is capable of performing type conversion. Although the input string is of type string, components within the string may be numerals. If the intent is to treat these numerals as integers or floating point values, simply designating the output variables with the proper type will cause a conversion to be performed.

Simple REGEX Matching

Expression	Description
.	Any character
\d	Any digit
\w	Any alphanumeric character
\s	Any white space
+	1 or more of the previous
*	0 or more of the previous

Format

As a receive buffer:

```
REGEXPSEARCH(search[, ifound])
```

As a string variable:

```
REGEXPSEARCH(Input_String, s_Regular_Exp_Pattern,
   i_Found_Tokens[, s_Output_Results[]])
REGEXPSEARCH(s_Input_String, s_Regular_Exp_Pattern,
   i_Found_Tokens, s_Match[, var1, var2, ...)]
```

Data Types

Argument	Type	Description
s_Input_String	String or String Variable (INPUT) [OPTIONAL]	The string or string variable to search for regex matches specified in regex.
s_Regular_Exp_Pattern	String (INPUT)	The string to search for in the receive buffer (searching from the current Rx Buffer pointer position forward) or an input string literal or input string variable.

Argument	Type	Description
i_Found_Tokens	numvar (OUTPUT) [OPTIONAL]	Returns whether or not the search string was found. 0: Regular expression pattern doesn't match 1: Regular expression pattern matches, but not expression groups designated 2: Regular expression pattern matches with 1 expression group designated N+1: Regular expression pattern matches with N expression groups designated **NOTE:** The variable I_found_tokens can be used as a test for match, since the value will be non-zero when the regular expression matches.
s_Match	String (OUTPUT) [CONDITIONAL]	Is only populated on pattern match, and must be designated when a list expression group output variables are used. When the group values are stored in an output array, then s_Match is NOT a valid parameter.
Variable List OR s_Output_Results[]	All are possible (OUTPUT) [OPTIONAL] OR String Array (OUTPUT) [OPTIONAL]	The list of variables to place the group values into. Value is assignment is in order of group values designated when following precedence rules.

The following examples search for a carriage-return and a line-feed in the receive buffer:

```
REGEXPSEARCH("\0d0a\")
```

The following example searches for the word alarm in the receive buffer:

```
REGEXPSEARCH("alarm")
```

NOTE: For hex substitution, \0000\ terminates a string; therefore, “xxxx\0000\yyyy” becomes “xxxx”.

A detailed example of searching for a pattern within a literal string value:

```
REGEXPSEARCH("2003 Jan 15 13:34:20",
   "(/\d+)/\s+(/\w+)/\s+(/\d+)/\s+(/\d+):(/\d+):(/\d+)
   ", i_Success, s_Match, s_Year, s_Month, s_Day,
   s_Hour, s_Minute, s_Second)
```

Where,

```
i_Success = 7
s_Match = 2003 Jan 15 13:34:20
s_Year = 2003
s_Month = Jan
s_Day = 15
s_Hour = 13
s_Minute = 34
s_Second = 20
```

For more information on regular expressions and the portable character set, see section Regular Expressions in Chapter 2.

Sentinel uses a POSIX (Portable Operating System Interface for UNIX)-compliant library for regular expressions. POSIX is a set of IEEE and ISO standards that help assure compatibility between POSIX-compliant operating systems, which includes most varieties of UNIX.

REPLACE

The REPLACE command searches and replaces strings.

When the search finds the string, it substitutes the replace string. The REPLACE command does a global replace, not just a replace of the first occurrence.

Format

```
REPLACE(dest_string, search, replace)
```

Data Types

Argument	Type	Description
dest_string	svar (INPUT/ OUTPUT)	The string variable that will have bytes replaced.
search	string (INPUT)	The search string to replace.
replace	string (INPUT)	The replacement string.

For example:

```
COPY(string:"The 1st time")
REPLACE(string, "1st", "2nd")
```

Result:

```
string = "The 2nd time"
```

> **NOTE:** In this example, you may substitute a regular expression for the “1st” string.

RESET

The RESET command resets the Rx buffer pointer to zero.

Format

```
RESET()
```

For example, the Rx buffer pointer position is shown by the ^ symbol.

```
rxbuff = "abcdefg"
              ^
RESET()
```

Result:

```
"abcdefg"
 ^
```

RXBUFF

The RXBUFF command overwrites the receive buffer with the contents of a quoted string or string variable. The contents of the receive buffer will change immediately and the Rx buffer pointer and held value will reset to zero.

Format

```
RXBUFF(s_data)
```

Data Types

Argument	Type	Description
s_data	string (INPUT)	The data string to write to the receive buffer. This string will immediately be the new receive buffer string.

For example:

In the following example, the FILER command reads a file called alert.data and places the contents of that file into a string variable called s_data. This example uses the assumption that:

```
alert.data: "Minor Alarm Xterminal A"
```

Next, the RXBUFF Command places that data into the receive buffer, just as though the data was received from a port.

```
FILER("alert.data", s_data)
RXBUFF(s_data)
//copies data from Rx BUFFER into S_Alarm_Priority,
   stopping before the string "Alarm")
COPY(S_Alarm_Priority:,” Alarm”)
```

Result:

```
S_Alarm_Priority= "Minor"
```

SEARCH

The SEARCH command performs a forward search in the receive buffer (Rx Buffer) for a string.

The search goes as follows:

- The search begins at the current Rx buffer pointer position and continues searching forward until the search finds the string or until the search reaches the end of the receive buffer.
- If the search finds the string, the Rx buffer pointer updates to point to the first byte of the string for which it searched. This Rx Buffer pointer position is retained when transitioning across states unless explicitly changed using the RESET Command.
- If the search does not find the string, the Rx Buffer pointer does not move.

When using this command, the optional second parameter is an integer variable that is set to 1 if the search finds the string and set to 0 if the search does not find the string.

Format

```
SEARCH(search[, ifound])
```

Data Types

Argument	Type	Description
search	string (INPUT)	The string to search for in the receive buffer (searching from the current Rx buffer pointer position forward).
ifound	numvar (OUTPUT) [OPTIONAL]	Returns whether or not the search string was found. 0 = not found 1 = found

For example:

The following examples search for a carriage-return and a line-feed.

```
SEARCH("\0d0a\")
SEARCH(data, ifound)
```

The following example searches for the word alarm:

```
SEARCH("alarm")
```

NOTE: For hex substitution, \0000\ terminates a string; therefore, “xxxx\0000\yyyy” becomes “xxxx”.

SET

The SET command processes a mathematical expression and updates a numeric value (numvar) with the result of the evaluation.

When using this command:

- Specify a destination numvar, followed by an equal sign, followed by any combination of ( ) - + * /, numerals and numeric variables.
- You must specify at least one numeric to the right of the equal sign.
- There is no restriction on the number of embedded parenthesis.
- All arguments are converted to a float; the result is converted to the type (integer or float) of the destination numvar.
- Up to 98 entries can be entered after the equal sign; these entries include: (,), *, /, +, -, any numeric and numeric variables.
- When operations have the same order of operation level, they are handled from left to right; the order of operation is described in the following table.

Level 1	:	()	for example: parenthesis
Level 2	:	*/	for example: multiplication, division
Level 3	:	+ -	for example: addition, subtraction

Format

```
SET(idest = <expr>) or SET(fdest = <expr>)
```

Where:

```
set_command ::= SET(<idest>=<expr>) |
  SET(<fdest>=<expr>)
expr ::= (<expr>)
         | expr ( '+' | '-' | '*' | '/' ) expr
         | ivar | fvar | number
```

Data Type

Argument	Type	Description
idest	numvar (OUTPUT)	The numeric variable (fvar or ivar) in which the value will be saved.
inum1	numeric (INPUT)	An fvar, ivar or number.
inum2	numeric (INPUT) [OPTIONAL]	An fvar, ivar or number.
inum3	numeric (INPUT) [OPTIONAL]	An fvar, ivar or number.
…	numeric (INPUT) [OPTIONAL]	An fvar, ivar or number.

For example:

```
SET(idest=inum1)
SET(i_loop=10)
SET(idest=inum1+inum2)
SET(idest=(inum1+inum2) * inum3)
SET(i_counter=i_counter+1)
SET(i_val = (ivar)*(ivar/3) + 15/fvar - (5 +
   20/iloop))
```

SETBYTES

The SETBYTES command allows you to set bytes within a string variable to a particular value, either passed as an integer or a string. If passed as an integer, valid ranges are 0 to 255. If a string is used as the replace parameter, then the string is placed starting at the index position in the destination string variable.

Format

```
SETBYTES(dest_string, index, replace)
```

Data Types

Argument	Type	Description
dest_string	svar (INPUT/ OUTPUT)	The string variable that will have bytes replaced.
index	numeric (INPUT)	The index (counting bytes starting with 0 for the first byte) into dest_string in which the bytes will be used to replace.
replace	string (INPUT) Or integer (INPUT)	The string bytes that will be written into the dest_string. The value to set for the index #n byte in the destination string.

For example:

```
COPY(string:"Bandwidth Util. = 22%")
SETBYTES(string, 18, "44")
```

Current Output Variables' Contents:

```
string = "Bandwidth Util. = 44%"
```

SETCONFIG

This command sets a system property. The current setting for the system property may then be retrieved using the GETCONFIG command. These commands are used to set system properties and retrieve current values for system properties that may change periodically, for example, a log file that is renamed daily using the current date.

Available system properties are:

System Property	Example(s)
▪ System.OS.Family	Solaris and Windows
▪ System.OS.Name	Windows 2000
▪ System.OS.Version.Major	5
▪ System.OS.Version.Minor	0
▪ System.Net.Hostname	ESECServer
▪ System.Net.IP_List	list of IP's for this host separated by a semicolon, an example is "172.163.3.45;172.45.2.1"

See also the GETCONFIG command.

There are two parameters with this command.

- The first required parameter defines the configuration option (“FileConnector.InputFile” or “FileConnector.OutputFile”) to set.
- The second required parameter defines the configuration value to set.

Format

```
SETCONFIG(Config Option, Value)
```

Data Types

Argument	Type	Description
Config Option	string (INPUT)	Name of the configuration variable to set. Input file = “FileConnector.InputFile” Output file = “FileConnector.OutputFile”
Value	string svar (INPUT)	Configuration setting.

For example:

```
SETCONFIG(“FileConnector.InputFile”, s_inputfilename)
SETCONFIG(“FileConnector.OutputFile”,
   s_outputfilename)
```

Current Output Variables’ Contents:

```
“C:/\test.dat”
```

SHELL

!

The SHELL command runs a shell script or command.

Format

```
SHELL(command [, wait_parameter][,
   wait_return_status])
```

Data Types

Argument	Type	Description
command	string (INPUT)	The path and filename of the command to run. By default, the PATH environment variable is used.
wait/no_wait	numvar [OPTIONAL]	Allows the SHELL command to wait (or not wait) for the launched program to complete execution before continuing processing. 0 = no_wait 1 = wait for program to complete

return_status	numvar [OPTIONAL]	Numeric value when the wait/no_wait option is used.
		SUCCESS = 1
		FAIL = 0

The following example initiates a PC batch file or a UNIX shell script:

```
SHELL("device_poll")
```

The following example launches Notepad:

```
SHELL("c:/\winnt/\system32/\notepad.exe")
```

The following example waits for the clock command to complete execution:

```
SHELL("clock",1)
```

The following example waits for a PC batch file or a UNIX shell script to complete execution then gets its return status:

```
SHELL("device_poll",1,i_ret)
```

The following example executes the clock process and does not wait for its completion:

```
SHELL("clock",0)
```

SKIP

The SKIP command adds a number to the Rx buffer pointer value.

The number can be positive or negative. If the resultant Rx buffer pointer position is less than zero, the Rx buffer pointer is set to zero. If the resultant Rx buffer pointer position is past the end of the receive buffer, the Rx buffer pointer is set to point to the last byte in the receive buffer.

Format

```
SKIP([+ | -] iskip_amount)
```

Data Types

Argument	Type	Description
iskip_amount	numeric (INPUT)	The number of bytes to move the Rx

For example:

```
SKIP(iskip_amount)
SKIP(+iskip_amount)
SKIP(-iskip_amount)
SKIP(5)
SKIP(-1)
```

Following are examples demonstrating the Rx buffer pointer position after a skip command, for the data:

```
aaaaaa bbbbb c d ee
      ^

SKIP(-2)
aaaaaa bbbbb c d ee
    ^

SKIP(-1)
aaaaaa bbbbb c d ee
     ^

SKIP(0)
aaaaaa bbbbb c d ee
      ^

SKIP(1)
aaaaaa bbbbb c d ee
       ^
SKIP(2)
aaaaaa bbbbb c d ee
        ^

SKIP(3)
aaaaaa bbbbb c d ee
         ^

SKIP(4)
aaaaaa bbbbb c d ee
          ^

SKIP(8)
aaaaaa bbbbb c d ee
              ^
```

SKIPWORD

The SKIPWORD command modifies the Rx buffer pointer so that it points to the beginning of a word.

This command considers a word to be each sequence of continuous printable bytes separated by at least one non-printable byte. Printable bytes are defined as ASCII and extended ASCII-0-255 (per ISO 8859-1).

By using positive and negative skip values, the Rx buffer pointer skips forward or backward through the receive buffer to the first or next printable byte in the receive buffer.

The Rx buffer pointer will not move past the end of the receive buffer or before the beginning of the receive buffer, even if the SKIPWORD command would cause it to do so.

A value of zero does not cause the Rx buffer pointer to change. The SKIPWORD command treats all characters less than 33 and between 126 and 161 as white space.

Format

```
SKIPWORD([+ | -] iwords)
```

Data Types

Argument	Type	Description
iwords	numeric (INPUT)	The number of words to move the Rx buffer pointer in the receive buffer.

For example:

```
SKIPWORD(iwords)
SKIPWORD(3)
SKIPWORD(+iwords)
SKIPWORD(-iwords)
SKIPWORD(-4)
```

Following are examples demonstrating the Rx buffer pointer position after a SKIPWORD command, for the data:

```
aaaaaa bbbbb c d ee
      ^

SKIPWORD(-2)
aaaaaa bbbbb c d ee
^

SKIPWORD(-1)
aaaaaa bbbbb c d ee
^
```

```
SKIPWORD(0)
aaaaaa bbbbb c d ee
      ^

SKIPWORD(1)
aaaaaa bbbbb c d ee
       ^

SKIPWORD(2)
aaaaaa bbbbb c d ee
             ^

SKIPWORD(3)
aaaaaa bbbbb c d ee
               ^

SKIPWORD(4)
aaaaaa bbbbb c d ee
                 ^

SKIPWORD(5)
aaaaaa bbbbb c d ee
                  ^
```

SNMPGET

The SNMPGET command sends SNMP requests to query the value of an SNMP variable.

All variables requested convert to a string representation and are stored in a specified destination variable.

- If the request fails, the status integer variable is set to zero
- If the request succeeds, the status integer variable is set to 1

Format

```
SNMPGET(address, community, mib_var, mib_value [,
   istatus])
```

Data Type

Argument	Type	Description
address	string (INPUT)	The IP address of the SNMP agent (for example: “192.168.0.10”).

Argument	Type	Description
community	string (INPUT)	The community string of the SNMP agent (for example: “public”).
mib_var	string (INPUT)	The text or numeric mib variable of the SNMP agent (for example: “mgmt.mib2.system.sysDescr.0” or ".1.3.6.1.4.1.4286.1.1.2.0").
mib_value	svar (OUTPUT)	The data variable that holds the MIB variable’s value.

istatus	numvar (OUTPUT) [OPTIONAL]	The status of the snmpget.
		FAIL = 0
		SUCCESS = 1

The following example uses text representations of the MIB variable.

```
SNMPGET("192.168.0.10", "public",
   "mgmt.mib2.system.sysDescr.0", s, i_status)
```

The following example uses numeric representations of the MIB variable.

```
SNMPGET("192.168.0.10", "public",
   ".1.3.6.1.4.1.4286.1.1.2.0", s, i_status)
```

SNMPSET

The SNMPSET command sends SNMP set commands to set the value of an SNMP variable. The mib file is held in the mibs directory. Wizard automatically loads and searches all mib files in the mibs directory.

All variables sent will be converted from a string representation to the variable type of the SNMP variable. If the request fails, the status integer variable (ivar) is set to 0. If the request succeeds, the status integer variable (ivar) is set to 1.

> **NOTE:** If the MIB value you are trying to set is greater than 2,560 bytes, it will be truncated to exactly that length. For example, 2,561 will show as 2,560; 4,000 will show as 2,560; etc.

Format

```
SNMPSET(mib_var_data_type, address, community,
   mib_var, mib_value [, i_status])
```

Data Types

Argument	Type	Description

mib_var_ data_type	string (INPUT)	The data type of the mib variable you are setting. The following are valid:
		i = Integer
		u = Unsigned integer
		t = Timeticks
		a = IP address
		o = Objid
		s = Octet String
		x = Octet string
		d = Decimal String

address	string (INPUT)	The IP address of the SNMP agent (for example: “192.168.0.10”).
community	string (INPUT)	The community string of the SNMP agent (for example: “public”).
mib_var	string (INPUT)	The mib variable (OID) of the SNMP agent. (for example: "mgmt.mib2.system.sysDescr.0" or “.1.3.6.1.4.1.4286.1.100”).
mib_value	string (INPUT)	The MIB variable is set to this value.

i_status	numvar (OUTPUT) [OPTIONAL]	The status of the snmpset.
		FAIL = 0
		SUCCESS = 1

For example:

The following example sets an integer mib variable identified by the OID ".1.3.6.1.4.1.4286.100" to the value of "1". If the operation is successful, i_status is 1. If the operation fails, i_status is 0.

```
SNMPSET("i", "192.168.100.2", "public",
   ".1.3.6.1.4.1.4286.1.100", "1", i_status)
```

SOCKETW

The SOCKETW command performs a NON-BLOCKING (network byte STREAM socket) open, connect, write of data to a socket (IP and TCP Port) and closes the socket. Optionally, it returns the status of the socket write attempt.

Format

```
SOCKETW(address, i_port, data [, istat])
```

Data Types

Argument	Type	Description
address	string (INPUT)	IP address of the socket.
i_port	numeric (INPUT)	TCP port number of the socket.

Argument	Type	Description
data	string (INPUT)	Data string to write to the socket.
istat	numvar (OUTPUT)	Optional returned status. istat = Number of bytes written; > 0 (SUCCESS) istat = 0 (FAILURE)

Examples:

```
SOCKETW("192.168.15.25", 5051, "Data Write Socket")
SOCKETW("192.168.15.25", i_port, "Data to Socket\0d\")
SOCKETW(s_ip_address, i_port, "\54AF0D0B91\",
   i_status)
SOCKETW(s_ip_address, i_port, "\54AF0D0B91\",
   f_status)
SOCKETW(s_ip_address, 6004, "\54AF0D0B91\", f_status)
SOCKETW(s_ip_address, 6004,sdata, f_status)
```

STONUM

The STONUM (string to number) command converts a string variable (svar) into a numeric variable (numvar).

CAUTION: String variables consisting of something other than the string representation of an integer or a float value may produce unpredictable results. All integer values are limited to 2147483647; values greater than this are truncated to 2147483647.

Format

```
STONUM(string, ivar)
```

Data Types

Argument	Type	Description
inum	numvar (OUTPUT)	The numeric variable in which the number is stored (ivar or fvar).
string	string (INPUT)	The string representation of a number (for example: “306”).

For example:

```
STONUM(source, idest)
STONUM(string_number, ivar)
STONUM("6512", ivar)
```

STRIP or STRIP-ASCII-RANGE

The STRIP command removes all occurrences of the strip string or ASCII range from the svar. The STRIP command always performs multiple-pass strips until the strip string or ASCII range is no longer found in the destination string variable.

When using this command, specify the string variable from which characters can be stripped. The remaining parameters can be either a string or numeric range start and end value.

> **NOTE:** Within the Visual Editor of the Agent Builder, STRIP and STRIP-ASCII-RANGE are listed as separate commands. They are same command. They are provided as descriptions for different variations of the same command. If you were to use STRIP-ASCII-RANGE in the text editor, you would enter STRIP.

Format

```
STRIP(dest, strip)
STRIP(dest, start ASCII range, stop ASCII range)
```

Data Types

Argument	Type	Description
dest	svar (INPUT/ OUTPUT)	The string variable that contains the string data that will be stripped of bytes depending on the second argument.
strip or start ASCII range	string or numeric (INPUT)	The string or start ASCII value to strip from the dest string.
stop ASCII range	numeric (INPUT [optional])	stop ASCII value **NOTE**: If start ASCII range is specified, this parameter is required.

The following examples are multiple-pass strips.

```
COPY(test:"THHELLOE")
STRIP(test, "HELLO")
```

After the STRIP() command, the variable test has the value of THE.

```
COPY(test2:"ABCDDEDDDFGDDH")
STRIP(test2, "D")
```

After the STRIP() command, the variable test2 has value of ABCEFGH.

```
COPY(test3:"ABCDDEDDDFGDDH")
STRIP(test3, 68, 69)
```

After the STRIP() command, the variable test3 has value of ABCFGH.

TBOSSETCOMMAND

The TBOSSETCOMMAND command builds a 3-byte TBOS command packet that may be transmitted to a device using the TBOS protocol.

The TBOS display number, command number, and command type are all used to put the correct TBOS command packet (3-bytes) into the output string variable. The format of the TBOS packet created using this parsing command is described in the following Remote Command Request tables.

Character 1		
Bit Numbers(s)	**Value**	**Meaning**
8 7	0 1	Operation Code: 01 = Remote Command Request (character 1)
6 5 4	MSB LSB	Display Number: 000 = No. 1 001 = No. 2 ... 111 = No. 7
3	0	No Meaning
2 1	MSB LSB	Type: 00 = momentary 01 = latch 10 = unlatch

Character 2		
Bit Numbers(s)	**Value**	**Meaning**
8 7	1 0	Operation Code: 10 = Remote Command Request (character 2)
6 5 4 3 2 1	MSB LSB	Remote Command Number: 000000 = No. 1 000001 = No. 2 ... 111111 = No. 63

Character 3		
Bit Numbers(s)	**Value**	**Meaning**
8	1	Echo of Character: The remote command response is the echo of this byte back to the port.
7	1	
6	0	
5	0	
4	1	
3	1	
2	0	
1	0	

Format

```
TBOSSETCOMMAND(cmd_bytes, idisp_num, icmd_num, type)
```

Data Types

Argument	**Type**	**Description**
cmd_bytes	svar (OUTPUT)	The hex data bytes (3 bytes total) that will be placed into this string variable and that may be used to transmit to a TBOS device in the Next State Transmit box.
idisp_num	numeric (INPUT)	The TBOS display number (or address) of the device (1 - 8). **NOTE:** Valid ranges for idisp_num are only 1 through 8; using any other value, the output (cmd_bytes), is set to all zeros, “\00 00 00\”.
i_cmd_num	numeric (INPUT)	The TBOS command number (1 - 64). **NOTE:** Valid ranges for i_cmd_num are only 1 through 64; using any other value, the output (cmd_bytes) is set to all zeros, “\00 00 00\”.
type	numeric (INPUT) Or string (INPUT)	The TBOS command type (0 - 2): 0 = momentary 1 = latch 2 = unlatch **NOTE:** Valid ranges for type are only 0 through 2; using any other value, type is set to 0 = “momentary” by default. The TBOS command type in string format. “momentary” or “m” = momentary “latch” or “l” = latch “unlatch” or “u” = unlatch This string is case-insensitive.

For example:

```
TBOSSETCOMMAND(string_cmd_bytes, 1, 1, 0)
TBOSSETCOMMAND(s_bytes, 1, 1, "latch")
TBOSSETCOMMAND(s_bytes, i_display, i_cmd_num, "U")
TBOSSETCOMMAND(s_bytes, i_display, i_cmd_num, 2)
TBOSSETCOMMAND(s_bytes, 1, 1, "momentary")
TBOSSETCOMMAND(s_bytes, 1, 1, "latch")
```

Remember to verify that the output cmd_bytes is set to "\00 00 00\" in order to check for any errors on inputs out of range. For example:

```
TBOSSETCOMMAND(cmd_bytes, i_display, i_cmd_num, "M")
IF(cmd_bytes = "\00 00 00\") /* INPUTS OUT OF RANGE */
...
ENDIF()
```

The following example builds a tbos command for display number 5, command number 33, and unlatched type.

```
TBOSSETCOMMAND(sbytes, 5, 33, 2)
```

Current Output Variables’ Contents:

```
sbytes = "\ba0 cc\"
```

TBOSSETREQUEST

The TBOSSETREQUEST command builds a 1-byte TBOS request packet that may be transmitted to a device using the TBOS protocol. The TBOS display number and request number is used to place the correct TBOS scan request byte into the output string variable. The format of the TBOS packet created using this parsing command is described in the following Character Scan Request and Response tables.

Character 1 – Character Scan Request

Bit Numbers(s)	Value	Meaning
8	0	Operation Code:
7	0	00 = Character Scan Request
6	MSB	Display No.:
5		000 = No. 1
4	LSB	001 = No. 2
		...
		111 = No. 3
3	MSB	Type:
2		000 = No. 1
1	LSB	001 = No. 2
		...
		111 = No. 8

Character 1 – Character Scan Response		
Bit Numbers(s)	**Value**	**Meaning**
8	MSB	Each bit in this response byte has a special meaning based on the character number sent (1-8) and the protocol of the device of the display number sent (1-8).
7		
6		
5		
4		
3		
2		
1	LSB	

Format

```
TBOSSETREQUEST(cmd_bytes, idisp_num, irequest_num)
```

Data Types

Argument	Type	Description
cmd_bytes	svar (OUTPUT)	The hex data byte is placed into this string variable and may be used to transmit to a TBOS device in the Next State Transmit box.
idisp_num	numeric (INPUT)	The TBOS display number (or address) of the device (1 - 8). **NOTE:** Valid ranges for idisp_num are only 1 through 8; with any other value, the output, cmd_bytes, will be set to all zero, "\00\."
irequest_num	numeric (INPUT)	The TBOS scan character number (1 - 8). **NOTE:** Valid ranges for irequest_num are only 1 through 8; with any other value, the output, cmd_bytes, will be set to zero, "\00\."

For example:

```
TBOSSETREQUEST(string_request_byte, 1, 1)
TBOSSETREQUEST(s_byte, idisp_num, i_scan_number)
```

The following example builds a TBOS scan request character for display number 2 and request number 1.

```
TBOSSETREQUEST(sbytes, 2, 1)
```

Current Output Variables' Contents:

```
sbytes = "\08\"
```

TIME

The TIME command copies the current time (in the format HH-MM-SS) into a string variable, ivar or fvar.

Format

```
TIME(dest)
```

Data Types

Argument	Type	Description
dest	svar (OUTPUT)	The string representation of the time is placed in this string variable (for example: “23-11-55”).
	numvar (OUTPUT)	The number of seconds since 00:00:00 UTC, January 1, 1970, is placed into this numeric variable (can be an fvar).

For example:

```
TIME(time_of_day)
TIME(i_num_seconds)
TIME(f_num_seconds)
```

NOTE: If you use an fvar, the time returned will be accurate to the microsecond.

TOKENIZE

The TOKENIZE command copies each component of a string between the delimiters into a string array. This can be useful when you are reading delimited data from a file and passing data to a script to be run on demand.

Every character in the string is treated as a potential token separator. For example, using the token separator "THE END" would not use the entire string as the separator. Rather, individual characters would be used as potential separators:

```
"T"
"H"
"E"
"E"
"N"
"D"
```

Format

```
TOKENIZE(data, delimiter, tokens[], itokens)
```

Data Types

Argument	Type	Description
data	svar (INPUT)	The data to be tokenized (for example: “xterm|subres|33|50”).
delimiter	string (INPUT)	The delimiter(s) to separate the tokens.
token	array (OUTPUT)	The array of tokens as found from the delimiterized string input data.
itokens	numvar (OUTPUT)	The number of tokens placed in the token string array.

For example:

```
COPY(data:"This|Data|Is|Tokenized")
TOKENIZE(data, "|",tokens[], inumtokens)
```

Current Output Variables’ Contents:

```
inumtokens = 4
tokens[0]= "This"
tokens[1]= "Data"
tokens[2]= "Is"
tokens[3]= "Tokenized"
```

In the following example, the data passed to the script is:

```
"There#are|several*fields|in*this#string".
```

There are three different token separators we want to use: #, | and *.

Current Output Variables' Contents:

```
i_tokens = 7
messages[0] = "There"
messages[1] = "are"
messages[2] = "several"
messages[3] = "fields"
messages[4] = "in"
messages[5] = "this"
messages[6] = "string"
```

In the following example, the data in the receive buffer is:

```
"Firewall Alarm - Major;Denial of Service Alarm -
   Major;"
COPY(rxbuff:)
TOKENIZE(rxbuff,";",msgs[],i_msgs)
```

Current Output Variables' Contents:

```
i_msgs = 2
msgs[0] = "Firewall Alarm - Major"
```

```
msgs[1] = "Denial of Service Alarm - Major"
```

TOLOWER

⬇

The TOLOWER command converts the contents of a string variable to all lowercase characters. The contents of the string variable that is passed through this command becomes all lowercase.

Format

```
TOLOWER(stringvar)
```

Data Types

Argument	Type	Description
stringvar	string (INPUT/ OUTPUT)	The string variable that contains the string to be converted to all lowercase.

For example:

```
s_var = "This Is Lower Case"
TOLOWER(s_var)
```

Result:

```
s_var = "this is lower case"
```

TOUPPER

⬆

The TOUPPER command converts the contents of a string variable to all uppercase characters. The contents of the string variable that is passed through this command becomes all uppercase.

Format

```
TOUPPER(stringvar)
```

Data Types

Argument	Type	Description
stringvar	string (INPUT/ OUTPUT)	The string variable that contains the string to be converted to all uppercase.

For example:

```
s_var = "This Is Upper Case"
toupper(s_var)
```

Result:

```
s_var = "THIS IS UPPER CASE"
```

TRANSLATE

The TRANSLATE command loads a comma-separated value (csv) file in memory, allowing for a fast lookup of whether or not the key entry is contained in the file and allowing retrieval of other data associated with the key.

The following are related to the TRANSLATE command.

- Comma-separated Value (CSV)
- Case-insensitive Key Searches
- Found Status
- Data Variables

Comma-separated Value (CSV) File

The csv file is a relative path from an agent's script directory. Agent Builder does not support edting of these files; therefore, e-Security suggests generating them through Microsoft Excel. The filename can be a string or a variable.

The csv file format is shown in the following example of a file named friends.csv:

```
key1,data1,data2,data3
Bob,blue,25,210
Alice,green,19,110
Pat,purple,36,145
```

To find if a particular friend is in your friend.csv file, the TRANSLATE command would look as follows:

```
TRANSLATE("Bob","friends.csv",i_found)
```

Or

```
COPY(s_Name:"Bob")
TRANSLATE(s_Name,"friends.csv",i_found)
```

Case-insensitive Key Searches

The key parameter can be either a string or a string variable. Additionally, an integer number or variable is supported. As the csv file is loaded into memory, the key of each entry is set to lowercase. The key in the TRANSLATE command is also set internally to lowercase to enable case-insensitive key searches.

Continuing the example of a csv file:

```
TRANSLATE("boB", "friends.csv",i_found)
```

This would have also found Bob in the csv file.

Found Status

The found status is set to 1 if the key is contained in the csv file and zero if the key is not contained in the csv file. A csv file with just key entries can be used with the TRANSLATE command just to determine if the key is a member of that file. For security purposes, a csv file may contain a list of known hostile IP addresses or valid usernames with other policy information like permissions and allowable access times.

> **NOTE:** Keys expressing ranges are not supported: IP addresses and numeric ranges.

Data Variables

Along with determining whether or not a key entry is in the csv file, associated data can be retrieved for that key. A variable number of script variables can be used to indicate into which variables to store the data. String, integer and float variables are supported. All data entries are stored as strings and will be converted to the type of variable supplied in the TRANSLATE command.

Continuing the example of friends.csv:

```
Bob,blue,25,210
Alice,green,19,110
Pat,purple,36,145
```

You can get the associated data with:

```
TRANSLATE(s_friend, "friends.csv", i_found, s_color,
   i_age, i_weight)
```

Where:

- If s_friend contains Alice, then i_found would equal 1, s_color would equal green, i_age would equal 19 and i_weight would equal 110.
- If the key entry is not found, then the variables are not modified (s_color, i_age, i_weight).
- If the entry for Alice was:
  `Alice,green,19,`

  Using the same TRANSLATE, the variable i_weight would be cleared (0 for integers, 0.0 for floats and "" strings). s_color would be green and i_age will be 19.
- If the entry for Alice was:
  `Alice,green,,thin,Ford`

  Using the same TRANSLATE, the variable i_age would be cleared, and thin would be converted into an integer(0) and put into i_weight. s_color would be green and Ford would be ignored.
- If the entry for Alice was:
  `Alice,25,19,110`

  Using the same TRANSLATE, the variable s_color would contain 25. i_age would be 19 and i_weight would be 110.

Format

```
TRANSLATE(<key>, <csv_file>, <found_status> [,
   <variable>, ...])
```

Data Types

Argument	Type	Description
key		The key to search for in the csv file.
csv_file		The filename of the csv file.
found_status		the integer variable set to 1 if the key is round in the csv file or zero if the key is not found in the csv file.
variable		the list of variables to place the data associated with the key into.

TRIM

Removes all white space (blanks) from both ends of a string, and replaces multiple white spaces within a string with single spaces. Blanks include the following characters:

- <tab>
- <carriage-return>
- <newline>
- <vertical-tab>
- <form-feed>
- <space>

Format

```
TRIM(svar)
```

Data Types

Argument	Type	Description
string	svar (INPUT)	String to trim white space from. The resulting string is stored in the input variable.

For example:

```
COPY(s_var:" Hello    World  "
TRIM(s_var)
```

Current Output Variables' Contents:

```
s_var = " Hello World "
```

WHILE

The WHILE command provides capability for looping control flow.

The While command goes as follows:

- If the result of the WHILE() statement is true, the parsing commands after the WHILE(), up to the next ENDWHILE() are executed.
- If the result of the WHILE() is false, no parsing commands are executed between the WHILE() and the ENDWHILE().

Although all data types are allowed on each side of the operator for the WHILE() statement, only numeric values can be compared with numeric and string with string.

The operator for the WHILE() compare can be <, =, >, <=, >=, <>, &, +, or ^.

> **CAUTION:** Do not use the logical NOT operator (^) in conjunction with a string variable. Doing so will generate a syntax error.

You cannot directly compare against a negative number. Use one of the following methods:

- Use the parsing function COMPARE
- Indirectly compare as follows:

```
SET(i_compare_val=-10)
WHILE(ivar >i_compare_val)
SET(ivar=ivar-1)
ENDWHILE()
```

Format

```
WHILE(<expr>)
```

Where:

```
expr ::= var
       | (<expr>)
       | ^ <expr>
```

Where <expr> must evaluate to integer or float.

```
       | <expr> <|=|>|<=|>=|<>|&|+ <expr>
```

Where both <expr> must evaluate to same type.

Data Types

Argument	Type	Description
data1	all (INPUT)	The data to compare to data2. If data2 is not used, then it becomes a logical (0 = false, anything else = true).
logical operator	< = > <= >= <> & + ^	Less Than Equal To Greater Than Less Than or Equal To Greater Than or Equal To Not Equal To Logical AND Logical OR Logical NOT
data2	all (INPUT) [OPTIONAL]	The data to compare to data1. This must be the same type as data1.
…	same as above	Use up to 200 individual parameters to create complex logical expressions.

For example:

```
WHILE(i<3)
SET(i=i+1)
ALERT("Still in loop")
ENDWHILE()
ALERT("Exited loop")
```

Chapter 4 – Wizard Administrator Functions

This chapter is intended for the Wizard system administrator. It describes various administrative functions the system administrator performs and provides information regarding Wizard's background processes.

NOTE: The first time the Wizard Agent Builder is run, you may see the following message: "Directory 'Agents' does not exist." It will be automatically created for you. Some information may have been lost." Select OK; the directory will be created and the Wizard Agent Builder will launch. If this message displays beyond the first time the Agent Builder is run, the Agent directory might have been inadvertently deleted and you will need to verify if information was lost.

Wizard Utilities and Applications

Wizard is comprised of a user interface (Agent Builder) and several additional utilities that work with the Agent Builder to perform network monitoring.

Agent Builder

The Wizard user interface is Agent Builder. Agent Builder allows you to configure the agents on your network as well as the ports and scripts that are used to communicate to the hosts. Agent Builder runs only on Windows.

NOTE: If you experience a problem with the way Wizard windows display after dragging the window to a different position, check your Display settings from the Microsoft Windows Control Panel. Under the Effects tab, uncheck Show window contents while dragging.

Port

In Wizard, ports enable an Agent to locate the security event data on the network by providing the IP address and other information about the source (security device [router, IDS, switch, etc…]). Each row in the Port Configuration table runs one agent script to one event source.

Agent Manager

The Agent Manager starts and stops port processing.

Agent Engine

The agent engine processes the template logic for each port. One agent engine is run for each active port.

popup.exe

The popup.exe utility is used by the Agent Engine to assist in processing popup or display parsing commands.

popup.cfg

The popup.cfg file is an optional file used to control timeouts of the popup and display parsing commands. If you do not have a popup.cfg, then the display and popup parsing commands do not time out.

To set a timeout for the display command, enter the statement:

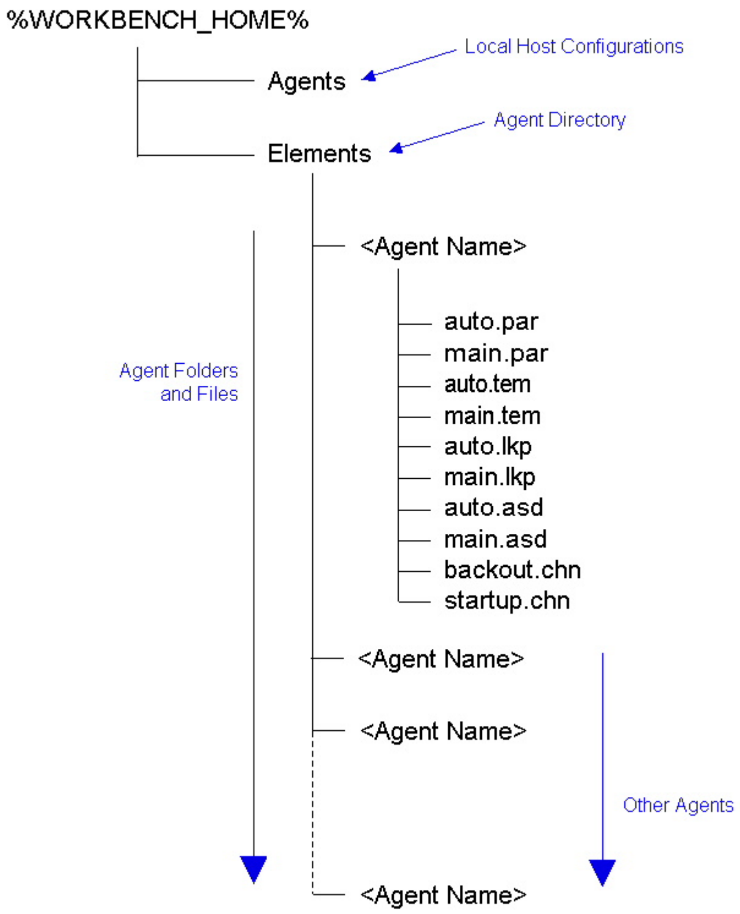
```
displaytimeout <true/false>.
```

The display timeout is set to 20 seconds.

To set a timeout for the popup command, enter the statement:

```
timeout <timeout in seconds>.
```

Wizard Directory Structure



Key

Agents	Port configurations files (Wizard Hosts)
Elements	Agent files
.par	Parameter files
.tem	Template files
.lkp	Lookup files
.asd	Active state description files

backout.chn	Backout script files
startup.chn	Startup script files

Chapter 5 – Wizard and Sentinel Meta-tags

Meta-tags store meta data. Meta-data is information about data, pre-defined variable names for meta data. For example, the source IP of an attack is stored in the SourceIP meta-tag. Product names are stored in the ProductName meta-tag. Data used to populate meta-tags is either extracted from device log data or is set as part of the agent processing.

To access the Event Configuration and mapping feature in the Sentinel Data Manager, click on the Events tab.

> **NOTE**: In the Free Form RuleLg Correlation Rule Language, when a label is preceded with an 'e.', such as e.crt, this refers to current events. If a label is preceded with a 'w.', such as w.crt, this refers to historical events.

The value in the Agent Variable column is the name of the agent variable to set in order to populate the corresponding Meta-tag. For more information about parsing commands see Chapter 3 and the documentation for specific agents located in

```
%ESEC_HOME%\wizard\elements\<agent name>\docs.
```

> **NOTE**: In the table below, Labels and Meta-tags are used in the Sentinel Control Center. Agent Variables are used in the agent parsing. Not all meta-tags have a corresponding Agent Variable.

The types specified in the Type column have the following properties:

- string – limited to 255 characters (unless otherwise specified)
- integer – 32 bit signed integer
- UUID – 36 character (with hyphens) or 32 character (without hyphens) hexadecimal string in the format XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX (e.g. - 6A5349DA-7CBF-1028-9795-000BCDFFF482)
- date – Agent Variable must be set with date as number of milliseconds since January 1, 1970 00:00:00 GMT. When displayed in Sentinel Control Center, meta-tags of type date will be displayed in a regular date format.
- IPv4 – IP address in dotted decimal notation (i.e. – xxx.xxx.xxx.xxx)

Label	**Meta-tag**	**Type**	**Description**	**Agent Variable**
CorrelatedEventUuids	ceu	string	List of event UUIDs associated with this correlated event. Only relevant for correlated events.	
Criticality	crt	integer	The criticality of the asset identified in this event.	s_CRIT

Label	**Meta-tag**	**Type**	**Description**	**Agent Variable**
Ct1 thru Ct2 (Reserved Customer)	ct1 thru ct2	string	Reserved for use by customers for customer-specific data (String).	s_CT1 and s_CT2
Ct3 (Reserved Customer)	ct3	integer	Reserved for use by customers for customer-specific data (Number).	s_CT3
CustomerVar1 thru CustomerVar10	cv1 thru cv10	integer	Reserved for use by customers for customer-specific data (Number).	s_CV1 thru s_CV10
CustomerVar11 thru CustomerVar20	cv11 thru cv20	date	Reserved for use by customers for customer-specific data (Date).	s_CV11 thru s_CV20
CustomerVar21 thru CustomerVar89	cv21 thru cv89	string	Reserved for use by customers for customer-specific data (String).	s_CV21 thru s_CV89
SARBOX	cv90	string	Sarbanes Oxley specific data.	s_CV90
HIPAA	cv91	string	Health Insurance Portability and Accountability Act (HIPAA) specific data.	s_CV91
GLBA	cv92	string	Gramm-Leach-Bliley Act (GLBA) specific data.	s_CV92
FISMA	cv93	string	Federal Information Security Management Act (FISMA) specific data.	s_CV93
NISPOM	cv94	string	National Industrial Security Program Operating Manual (NISPOM) specific data.	s_CV94
SIPCountry	cv95	string	Country of source IP.	s_CV95
DIPCountry	cv96	string	Country of destination IP.	s_CV96
CustomerVar97 thru CustomerVar100	cv97 thru cv100	string	Reserved for use by customers for customer-specific data (String).	s_CV97 thru s_CV100
DateTime	dt	date	The normalized date and time of the event, as given by the agent.	

Label	**Meta-tag**	**Type**	**Description**	**Agent Variable**
DestinationHostName	dhn	string	The destination host name to which the event was targeted.	s_DHN
DestinationIP	dip	IPv4	The destination IP address to which the event was targeted.	s_DIP
DestinationPort	dp	string (32)	The destination port to which the event was targeted.	s_DP
DestinationUserName	dun	string	The destination user name on which an action was attempted. Example: Attempts to reset the password of root.	s_DUN
EventID	id	UUID	Unique identifier for this event.	
EventTime	et	string	The normalized time of the event as reported by the sensor; parsed into the format: Y-M-D-H:M:S~AMPM24~TZ.	s_ET
EventName	evt	string	The descriptive name of the event as reported (or given) by the sensor. Example "Port Scan".	s_EVT
ExtendedInformation	ei	string (1000)	Stores additional agent-collected information. Values within this variable are separated by semi-colons (;). Example: A domain for an ID or file names.	s_EI
FileName	fn	string (1000)	The name of the program executed or the file accessed, modified or affected. Example: The name of a virus-infected file or a program detected by an IDS.	s_FN
Message	msg	string (4000)	Free-form message text for the event.	s_BM
Protocol	prot	string	The network protocol of the event.	s_P

Label	**Meta-tag**	**Type**	**Description**	**Agent Variable**
ProductName	pn	string	Indicates the type, vendor and product code name of the sensor from which the event was generated. Example: Check Point FireWall=CPFW.	s_PN
ReporterName	rn	string	The host name or IP address of the device to which an event was logged or from which notification of the event is sent.	s_RN
ReservedVar1 thru ReservedVar10	rv1 thru rv10	integer	Reserved by e-Security for expansion (Number).	s_RV1 thru s_RV10
ReservedVar11 thru ReservedVar20	rv11 thru rv20	date	Reserved by e-Security for expansion (Date).	s_RV11 thru s_RV20
ReservedVar21 thru ReservedVar25	rv21 thru rv25	UUID	Reserved by e-Security for expansion (UUID).	s_RV21 thru s_RV25
(Windows and Solaris) ReservedVar26 thru ReservedVar29	rv26 thru rv29	string	Reserved by e-Security for expansion (String).	s_RV26 thru s_RV29
ControlPack (Linux only)	rv26	string	e-Security control categorization level 1	s_RV26
ControlMonitor (Linux only)	rv27	string	e-Security control categorization level 2	s_RV27
ReservedVar28	rv28	string	Reserved by e-Security for expansion (String).	s_RV28
SourceIPCountry (Linux only)	rv29	string	Country of source IP address.	s_RV29
AttackID	rv30	string	Normalized Attack ID (Advisor attack ID)	s_RV30
DeviceName	rv31	string	Name of security device	s_RV31
DeviceCategory	rv32	string	Device category (AV, DB, ESEC, FW, IDS, OS). AV: Anti-virus DB: database ESEC: system event FW: firewall IDS: intrusion detection OS: operating system	s_RV32

Label	Meta-tag	Type	Description	Agent Variable
EventContext	rv33	string	Event context (threat level).	s_RV33
SourceThreatLevel	rv34	string	Source threat level.	s_RV34
SourceUserContext	rv35	string	Source user context.	s_RV35
DataContext	rv36	string	Data context.	s_RV36
SourceFunction	rv37	string	Source function.	s_RV37
SourceOperationalContext	rv38	string	Source operational context.	s_RV38
MSSPCustomerName	rv39	string	MSSP customer name.	s_RV39
ReservedVar40 thru ReservedVar43	rv40 thru rv43	string	Reserved by e-Security for expansion (String).	s_RV40 thru s_RV43
DestinationThreatLevel	rv44	string	Destination threat level.	s_RV44
DestinationUserContext	rv45	string	Destination user context.	s_RV45
VirusStatus	rv46	string	Virus status.	s_RV46
DestinationFunction	rv47	string	Destination function.	s_RV47
DestinationOperationalContext	rv48	string	Destination operational context.	s_RV48
ReservedVar49	rv49	string	Reserved by e-Security for expansion (String).	s_RV49
eSecTaxonomyLevel1	rv50	string	e-Security event code categorization - level 1.	s_RV50
eSecTaxonomyLevel2	rv51	string	e-Security event code categorization - level 2.	s_RV51
eSecTaxonomyLevel3	rv52	string	e-Security event code categorization - level 3.	s_RV52
eSecTaxonomyLevel4	rv53	string	e-Security event code categorization - level 4.	s_RV53
ReservedVar54 thru ReservedVar55	rv54 thru rv55	string	Reserved by e-Security for expansion (String).	s_RV54 thru s_RV55
SourceAssetName	rv56	string	Source (Asset Mgmt) – Asset Name	s_RV56
SourceMacAddress	rv57	string	Source (Asset Mgmt) – Mac Address	s_RV57
SourceNetworkIdentity	rv58	string	Source (Asset Mgmt) – Network Identity	s_RV58
SourceAssetCategory	rv59	string	Source (Asset Mgmt) – Asset Category	s_RV59
SourceEnvironmentIdentity	rv60	string	Source (Asset Mgmt) – Environment Identity	s_RV60
SourceAssetValue	rv61	string	Source (Asset Mgmt) - AssetValue	s_RV61
SourceCriticality	rv62	string	Source (Asset Mgmt) - Criticality	s_RV62

Label	Meta-tag	Type	Description	Agent Variable
SourceSensitivity	rv63	string	Source (Asset Mgmt) - Sensitivity	s_RV63
SourceBuilding	rv64	string	Source (Asset Mgmt) - Building	s_RV64
SourceRoom	rv65	string	Source (Asset Mgmt) - Room	s_RV65
SourceRackNumber	rv66	string	Source (Asset Mgmt) – Rack Number	s_RV66
SourceCity	rv67	string	Source (Asset Mgmt) - City	s_RV67
SourceState	rv68	string	Source (Asset Mgmt) - State	s_RV68
SourceCountry	rv69	string	Source (Asset Mgmt) - Country	s_RV69
SourceZipCode	rv70	string	Source (Asset Mgmt) – Zip Code	s_RV70
SourceAssetOwner	rv71	string	Source (Asset Mgmt) – Asset Owner	s_RV71
SourceAssetMaintainer	rv72	string	Source (Asset Mgmt) – Asset Maintainer	s_RV72
SourceBusinessUnit	rv73	string	Source (Asset Mgmt) – Business Unit	s_RV73
SourceLineOfBusiness	rv74	string	Source (Asset Mgmt) – Line of Business	s_RV74
SourceDivision	rv75	string	Source (Asset Mgmt) - Division	s_RV75
SourceDepartment	rv76	string	Source (Asset Mgmt) - Department	s_RV76
SourceAssetId	rv77	string	Source (Asset Mgmt) – Source Asset Id	s_RV77
DestinationAssetName	rv78	string	Destination (Asset Mgmt) – Asset Name	s_RV78
DestinationMacAddress	rv79	string	Destination (Asset Mgmt) – Mac Address	s_RV79
DestinationNetworkIdentity	rv80	string	Destination (Asset Mgmt) – Network Identity	s_RV80
DestinationAssetCategory	rv81	string	Destination (Asset Mgmt) – Asset Category	s_RV81
DestinationEnvironmentIdentity	rv82	string	Destination (Asset Mgmt) – Environment Identity	s_RV82
DestinationAssetValue	rv83	string	Destination (Asset Mgmt) – Asset Value	s_RV83
DestinationCriticality	rv84	string	Destination (Asset Mgmt) - Criticality	s_RV84

Label	Meta-tag	Type	Description	Agent Variable
DestinationSensitivity	rv85	string	Destination (Asset Mgmt) - Sensitivity	s_RV85
DestinationBuilding	rv86	string	Destination (Asset Mgmt) - Building	s_RV86
DestinationRoom	rv87	string	Destination (Asset Mgmt) - Room	s_RV87
DestinationRackNumber	rv88	string	Destination (Asset Mgmt) – Rack Number	s_RV88
DestinationCity	rv89	string	Destination (Asset Mgmt) - City	s_RV89
DestinationState	rv90	string	Destination (Asset Mgmt) - State	s_RV90
DestinationCountry	rv91	string	Destination (Asset Mgmt) - Country	s_RV91
DestinationZipCode	rv92	string	Destination (Asset Mgmt) – Zip Code	s_RV92
DestinationAssetOwner	rv93	string	Destination (Asset Mgmt) – Asset Owner	s_RV93
DestinationAssetMaintainer	rv94	string	Destination (Asset Mgmt) – Asset Maintainer	s_RV94
DestinationBusinessUnit	rv95	string	Destination (Asset Mgmt) – Business Unit	s_RV95
DestinationLineOfBusiness	rv96	string	Destination (Asset Mgmt) – Line of Business	s_RV96
DestinationDivision	rv97	string	Destination (Asset Mgmt) - Division	s_RV97
DestinationDepartment	rv98	string	Destination (Asset Mgmt) - Department	s_RV98
DestinationAssetId	rv99	string	Destination (Asset Mgmt) – Destination Asset Id	s_RV99
ReservedVar100	rv100	string	Reserved by e-Security for expansion (String).	s_RV100
Resource	res	string	The resource name.	s_Res
DeviceAttackName	rt1	string	For use with Advisor. Attack name from Security Device.	s_RT1
Rt2	rt2	string	Populated with the correlation rule name when a correlation rule is generates an event.	s_RT2
Rt3	rt3	integer	Reserved by e-Security for expansion (Number).	s_RT3

Label	Meta-tag	Type	Description	Agent Variable
SourceHostName	shn	string	The source host name from which the event originated.	s_SHN
SourceID	src	UUID	Unique identifier for the e-Security process that generated this event.	
SourceIP	sip	IPv4	The source IP address from which the event originated	s_SIP
SensorName	sn	string	The name of the "ultimate detector" of the event when received in raw data. Example "FW1" for a firewall.	s_SN
Severity	sev	integer	The normalized severity of the event (0-5).	i_Severity
SourcePort	sp	string (32)	The source port from which the event originated.	s_SP
SensorType	st	string (5)	The single character designator for the sensor type (N, H, I, O, P, V, C, W). C: Correlation H: host-based I: internal (system event) N: network-based O: Other P: performance (system event) V: Anti-virus W: Watchlist	s_ST
SourceUserName	sun	string	The source user name used to initiate an event. Example "jdoe" during an attempt to "su".	s_SUN
SubResource	sres	string	The sub-resource name.	s_SubRes
Vulnerability	vul	integer	The vulnerability of the asset identified in this event.	s_VULN

Label	**Meta-tag**	**Type**	**Description**	**Agent Variable**
WizardAgent	agent	string (64)	e-Security agent that generated the event. For system events, agent will be either Performance or Internal.	
WizardPort	port	string (64)	e-Security agent port description.	

Chapter 6 – Sentinel Control Center User Permissions

User permissions are broken down as follows:

- General
  - Public Filters
  - Private Filters
  - Integration Actions
- Active View
  - Menu Items
  - Summary Displays
- iTRAC
  - Template Management
  - Process Management
- Incidents
- Agent Management
- Analysis
- Advisor
- Administration
  - Correlation
  - DAS Statistics
  - Event File Information
  - Server Views
  - Global Filters
  - iTRAC Role Management
  - Menu Configuration
  - User Management
  - User Session Management

Default Users

The installer will create the following default users on the Sentinel Server:

Oracle and MS SQL Authentication:

- esecdba - Schema owner (configurable at install time).
- esecadm - Sentinel administrator user (configurable at install time).

> **NOTE**: For UNIX, the Installer also creates the operating system user with the same user name and password.

- esecrpt - Reporter user, password as the admin user.
- ESEC_CORR - Correlation Engine users, used to create incidents.
- esecapp - e-Security application username for connecting to the database.

Windows Authentication:

- e-Security DB Administrator - Schema owner (configurable at install time).
- e-Security Administrator - Sentinel administrator user (configurable at install time).
- e-Security Report User - Reporter user, password as the admin user.
- e-Security Application DB User - e-Security application username for connecting to the database.

General

Permission Name	Description
Save Workspace	Allows the user to save preferences. If this permission is unavailable, the user will never be prompted to save changes to preferences when logging out or exiting the Sentinel Control Center.
Column Management	Allows the user to manage the columns in the Active View tables.
Snapshot	Allows the user to take a snapshot of Active View tables.

General – Public Filters

Permission Name	Description
Create Public Filters	Allows the user to create a filter with an owner ID of PUBLIC. If the user does not have this permission, then the value PUBLIC will not be listed as one of the owner IDs that the user can create a filter for.
Modify Public Filters	Allows the user to modify a public filter.
Delete Public Filters	Allows the user to delete a public filter.

General – Private Filters

Permission Name	Description
Create Private Filters	Allows the user to create private filters for themselves or for other users.
Modify Private Filters	Allows the user to modify their own private filters and private filters created by other users.
Delete Private Filters	Allows the user to delete their own private filters and private filters created by other users.
View/Use Private Filters	Allows the user to view their own private filters and private filters crated by other users.

General – Integration Actions

Permission Name	Description
Send to HP Open View	Allows the user to send events, incident and associated objects to HP-OVO.
Send Event to HP Service Desk	Allows the user to send events, incident and associated objects to HP Service Desk.
Send to Remedy Help Desk	Allows the user to send events, incident and associated objects to Remedy.

Active Views

Permission Name	Description
View Active View Tab	Allows the user to see and use the Active View tab, menu and other related functions associated with the Active View tab.

Active Views – Menu Items

Permission Name	Description
Use Assigned Menu Items	Allows the user to use assigned menu items in the Active View Events table (the right-click menu).
Add to Existing Incident	Allows the user to add events to existing incidents using the Active View Events table (the right-click menu).
Remove from Incident	Allows the user to remove events from an existing incident using the Active View Events table (the right-click menu).
Email Events	Allows the user to e-mail events using the Active View Events table (the right-click menu).
View Advisor Attack Data	Allows the user to view the Advisor Attack Data stream from e-Security.
View Vulnerability	Allows the user to view the output of a Nessus scan.

Active Views – Summary Displays

Permission Name	Description
Use/View Summary Displays	Allows the user to access the Active View charts.

iTRAC

Permission Name	Description
View iTRAC Tab	Allows the user to see and use the iTRAC tab, menu and other related functions associated with the iTRAC tab.
Activity Management	Allows the user to access the Activity Manager.

Template Management

Permission Name	Description
View/Use Template Manager	Allows the user to access the Template Manager.
Create/Modify Templates	Allows the user to create and modify templates.

Process Management

Permission Name	Description
View/Use Process Manager	Allows the user to access the Process View Manager.
Control Processes	Allows the user to use the Process View Manager.

Incidents

Permission Name	Description
View Incidents Tab	Allows the user to see and use the View Incidents tab, menu and other related functions associated with the View Incidents tab.
Incident Administration	Allows the user to modify an incident.
View Incident(s)	Allows the user to see the details of an incident. If the user does not have this permission, then the Incident Details window will not display when the user double-clicks an incident in the Navigator window or on an incident in the case’s Incident tab.
Create Incident(s)	Allows the user to create Incidents in the Event menu accessible by right-clicking an event.
Modify Incident(s)	Allows the user to modify an incident in the Incident Details window.
Delete Incident(s)	Allows the user to delete incidents.
Assign Incident(s)	Allows the user to assign an incident in the Modify and Create Incident window.
Email Incidents	Allows the user to e-mail Incidents of interest.
Incident Actions	Allows the user to enable/disable the Incident Action Configuration/Execution.

Agent Management

Permission Name	Description
View Agents	▪ View the 'Agents' tab in Sentinel Control Center ▪ View the 'Wizard Hosts' tab in Agent Builder
Control Agents	▪ Includes all capabilities as the 'View Agents' permission ▪ Allows for the Command and Control of Agents from the Sentinel Control Center ▪ Allows for the Command and Control of Agents from the Wizard Agent Builder
Agent Administration	▪ Includes all capabilities as the 'Command Agents' permission ▪ In Agent Builder, Agent editing and deployment ▪ In Agent Builder, creating, editing, compiling and debugging Agents. ▪ In Agent Builder, uploading and downloading Agents. ▪ In Agent Builder, exporting Wizard Hosts ▪ In Agent Builder, adding, editing, deleting Ports ▪ In Agent Builder, setting Port options

Command and Control consists of:

- start/stop individual ports
- start/stop all ports
- restart hosts
- rename hosts

Analysis

Permission Name	Description
View Analysis Tab	Allows the user to see and use the View Analysis tab, menu and other related functions associated with the View System Overview tab.

Advisor

Permission Name	Description
View Advisor Tab	Allows the user to see and use the View Advisor tab, menu and other related functions associated with the View Advisor tab.

Administration

Permission Name	Description
View Administration Tab	Allows the user to see and use the View Administration tab, menu and other related functions associated with the View Administration tab.

Administration – Correlation

Permission Name	Description
Use/View Correlation Engine Manager	Allows the user to see and use the Correlation Engine.

Permission Name	Description
Use/View Correlation Rules	Allows the user to start or stop the Correlation Rules.

Administration – Global Filters

Permission Name	Description
View/Use Global Filters	Allows the user to access the Global Filter Configuration window.
Modify Global Filters	Allows the user to modify the global filters configuration. **NOTE:** To access this function, View Global Filters permission must also be assigned.

Administration – Menu Configuration

Permission Name	Description
Menu Configuration	Allows the user to access the Menu Configuration window and add new options that display on the Event menu when you right-click an event.

Administration - DAS Statistics

Permission Name	Description
DAS Statistics	Allows the user to view DAS activity (DAS binary and query).

Administration – Event File Information

Permission Name	Description
Event File Information	Allows the user to view event file status.

Administration – Server Views (Linux only)

Permission Name	Description
View Servers	Allows the user to monitor the status of all processes.
Control Servers	Allows the user to start, restart and stop processes.

Administration – User Management

Permission Name	Description
Use/View User Account	Allows the user to use and view user accounts.
Create User Account	Allows the user to create a user account. **NOTE:** To access this function, View/User Account permission must also be assigned.
Modify Existing User Account	Allows the user to modify an existing user’s account. **NOTE:** To access this function, View/User Account permission must also be assigned.

Permission Name	**Description**
Delete User Account	Allows the user to delete an existing user's account. **NOTE:** To access this function, View/User Account permission must also be assigned.

Administration – User Session Management

Permission Name	**Description**
User Session Management	Allows the user to view, lock and terminate active users (logins to Sentinel Control Center).

Administration – iTRAC Role Management

Permission Name	**Description**
iTRAC Role Management	Allows the use and view the role manager in the Admin tab.

Chapter 7 – Sentinel Correlation Engine

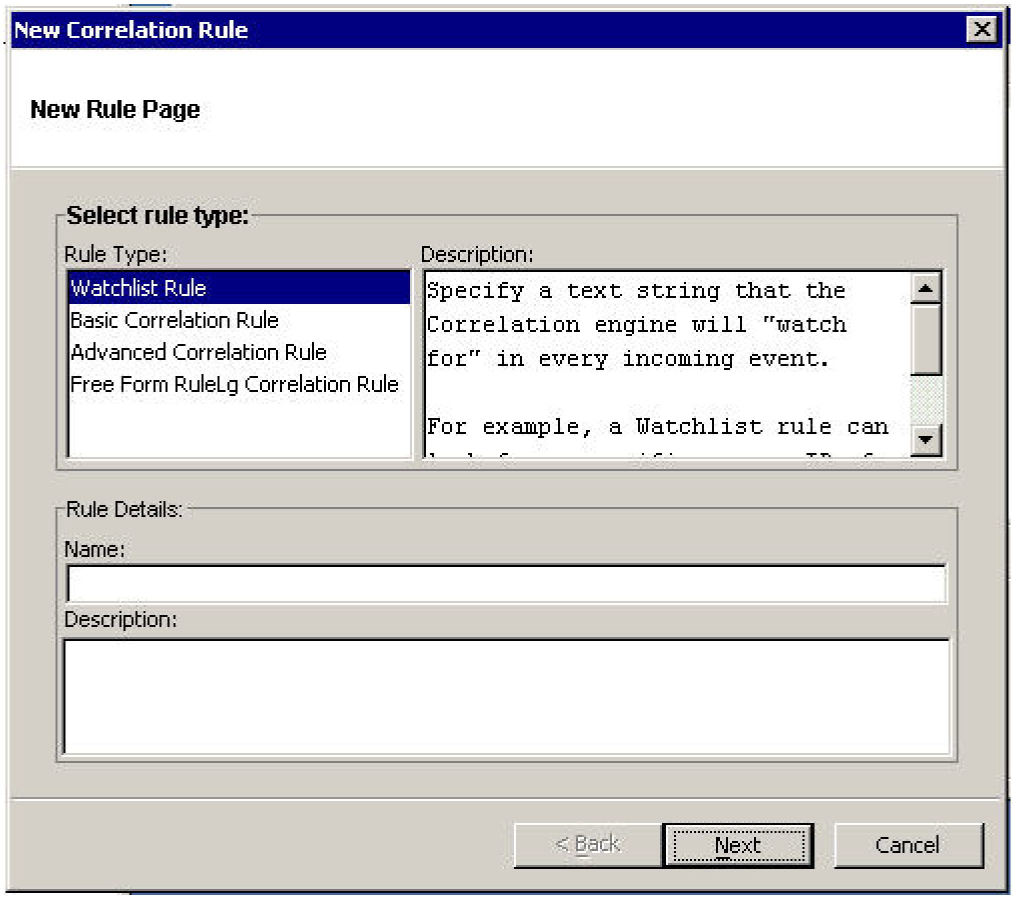
The Sentinel Correlation Engine is a memory-resident, multi-threaded application. Multi-threading allows the correlation engine to take advantage of multi-processor hardware, such as SMP (symmetric multiprocessing) machines.

The correlation engine is designed to receive data from security devices, network devices and other application sources and search for significant patterns, usually within certain timeframes. These patterns could indicate attacks, intrusions, misuse or failure. When a correlated event is generated, the rt2 field will populate with the correlation rule name.

Sentinel Correlation Engine provides a scalable deployment. This architecture allows for the deployment of a distributed network of correlation engines working in concert to correlate in real-time across security relevant data, including real-time monitored security events, vulnerability scan results for potentially targeted systems and asset information indicating those systems' relative importance to critical business processes and their association with other systems in the organization.

Sentinel Correlation Engine is rule-based. You direct the correlation engine's processing through rules created in the Sentinel Control Center's rule editor. The rule editor is based on a collection of Rule Wizards that give multiple options for rule creation. The Rule Wizards are:

- Watchlist
- Basic Correlation
- Advanced Correlation
- Free Form RuleLg



Rule Type: Watchlist Rule

Description:

```
Specify a text string that the
Correlation engine will "watch
for" in every incoming event.

For example, a Watchlist rule can
```

Correlation Filter Types

For Watchlist, Basic Correlation and Advanced Correlation, there are four different Filter Types to choose from. They are:

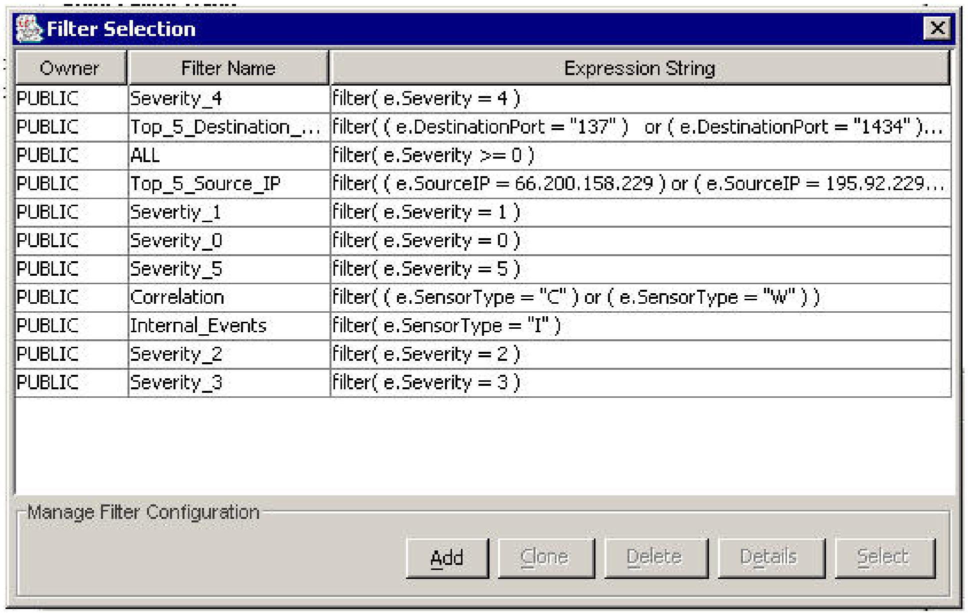
- Allow All - Equivalent to running filter severity greater than and equal to zero.
- Pattern - Any regular expression with a grep-like syntax. A rule can look for a specific source IP address of a hacker and notify you anytime that IP address is seen in any event message.
- Filter Manager - A drop-down list that displays the Filter Manager to select or create a new filter.
- Builder - Create criteria for inclusion and exclusion of events based on boolean algebra.

Pattern Type Correlation Filter

A Pattern Type Correlation Filter uses any regular expression with a grep-like syntax. The regular expression match is done with a concatenation of all of the present meta-tags for each incoming event. For example, virusXYZ will watch for the string virusXYZ in any present meta-tags for each incoming event.

Filter Manager Type Correlation Filter

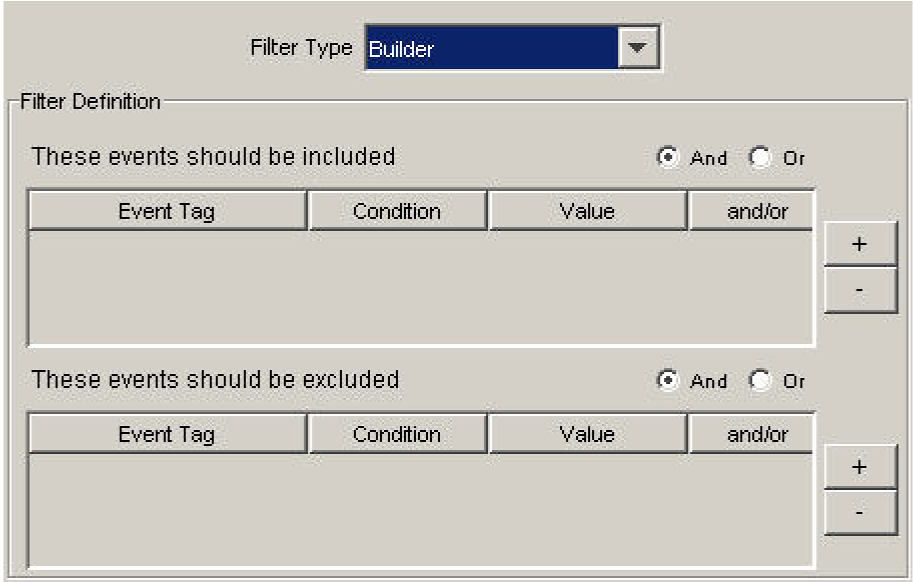
This option allows to select an existing filter or to create a filter to use in your correlation though the Filter Manager window.



Owner	Filter Name	Expression String
PUBLIC	Severity_4	filter(e.Severity = 4)
PUBLIC	Top_5_Destination_...	filter((e.DestinationPort = "137") or (e.DestinationPort = "1434")...
PUBLIC	ALL	filter(e.Severity >= 0)
PUBLIC	Top_5_Source_IP	filter((e.SourceIP = 66.200.158.229) or (e.SourceIP = 195.92.229...
PUBLIC	Severtiy_1	filter(e.Severity = 1)
PUBLIC	Severity_0	filter(e.Severity = 0)
PUBLIC	Severity_5	filter(e.Severity = 5)
PUBLIC	Correlation	filter((e.SensorType = "C") or (e.SensorType = "W"))
PUBLIC	Internal_Events	filter(e.SensorType = "I")
PUBLIC	Severity_2	filter(e.Severity = 2)
PUBLIC	Severity_3	filter(e.Severity = 3)

Builder Type Correlation Filter

There are two parts to the Builder Type Correlation Filter. One part is criteria for inclusion (which events should be included in the pattern match) and the other part is exclusion (which events should be excluded from the pattern match).


Filter Type Builder

Filter Definition

These events should be included ● And ○ Or

Event Tag	Condition	Value	and/or

These events should be excluded ● And ○ Or

Event Tag	Condition	Value	and/or

- Which events should be included in the pattern match - Use this table to specify conditions to limit which events will trigger the correlation.
    - Event Tag - The Event Tag column is a drop-down list of available event tags (also known as meta-tags) that can be correlated on.
    - Condition - The Condition column is a drop-down list of operators used in constructing a correlation condition.
    - Value - The Value column is a free-form field that you can use to enter values if the conditions =, !=, <, >, <=, or >= are chosen. If either =Meta-Tag or !=Meta-Tag are selected in the Condition column, the Value

column will contain a drop-down list of available meta-tags from which to choose. You may enter anything with the following constraints:
- Single quotes may never be entered.
- Wild cards characters are the asterisk (*) and the period (.) and may appear anywhere in the string if using regex.
- There are no escape characters; that is, wild cards cannot be escaped out.

  - and/or - Switch from and to or by clicking one of these boxes. When multiple conditions are specified in this table, the 'and' and 'or' buttons allow you to specify whether all of the conditions need to be met or just one of the conditions needs to be met. Choose and to indicate that all conditions must be met. Choose or to indicate that only one of the conditions must be met.

> **NOTE:** Your selection becomes valid only if the table contains a second or more rows. All rows in the table will default to this logical operator except the last row. Combinations of 'and' and 'or' are not possible between rows within the table.

  - +/- Buttons The + button will add an additional row to the end of the table. The - button will remove the selected row from the table regardless of its position within the table.
- Which events should be excluded from the pattern match - Use this table to specify conditions to limit which events will not trigger the correlation rule.
  - Event Tag - A list of available event tags that can be correlated on.
  - Condition - The Condition column is a drop-down list of operators used in constructing a correlation condition.
  - Value - The Value column is a free-form field that you can use to enter values if the conditions =, !=, <, >, <=, or >= are chosen. If either =Meta-Tag or !=Meta-Tag are selected in the Condition column, the Value column will contain a drop-down list of available meta-tags from which to choose. You may enter anything with the following constraints:
    - Single quotes may never be entered.
    - Wild cards characters are the asterisk (*) and the period (.) and may appear anywhere in the string if using regex.
    - There are no escape characters; that is, wild cards cannot be escaped out.
  - and/or - Switch from and to or by clicking one of these boxes. When multiple conditions are specified in this table, the 'and' and 'or' buttons allow you to specify whether all of the conditions need to be met or just one of the conditions needs to be met. Choose and to indicate that all conditions must be met. Choose or to indicate that only one of the conditions must be met.

> **NOTE:** Your selection becomes valid only if the table contains a second or more rows. All rows in the table will default to this logical operator except the last row. Combinations of 'and' and 'or' are not possible between rows within the table.

  - +/- Buttons - The + button will add an additional row to the end of the table. The - button will remove the selected row from the table regardless of its position within the table.

Correlation Rule Definition

Correlation Rule Wizards: Watchlist, Basic Correlation and Advanced Correlation allow you to quickly add a pre-defined rule type, depending on what you want to accomplish. The wizard for each rule type handles generation of the correlation rule in the native Correlation Engine rule language. Each of these rules is created using the Correlation Rules window under the Admin tab.

The Rule Wizard includes a free-form editor that allows you to use the RuleLg Correlation Definition Language to add the rule directly into the native Correlation Engine rule language.

Watchlist

There are four different Filter Types to choose from. They are:

- Allow All - Equivalent to running filter severity greater than and equal to zero.
- Pattern - Any regular expression with a grep-like syntax.
- Filter Manager - A drop-down list that displays the Filter Manager to select or create a new filter.
- Builder - Create criteria for inclusion and exclusion of events based on boolean algebra.

For more information, see Creating a Watchlist Rule.

Basic Correlation

There are four different Filter Types to choose from. They are:

- Allow All - Equivalent to running filter severity greater than and equal to zero.
- Pattern - Any regular expression with a grep-like syntax.
- Filter Manager - A drop-down list that displays the Filter Manager to select or create a new filter.
- Builder - Create criteria for inclusion and exclusion of events based on boolean algebra.

This rule allows you to count the number of times certain conditions are met within a specific timeframe.

For example, a Basic Correlation rule can look for the same source IP address reported five times in five minutes, even if the events are reported from different products, such as an intrusion detection system (IDS) and a firewall.

For more information, see Creating a Basic Correlation Rule.

Advanced Correlation

There are four different Filter Types to choose from. They are:

- Allow All - Equivalent to running filter severity greater than and equal to zero.
- Pattern - Any regular expression with a grep-like syntax.
- Filter Manager - A drop-down list that displays the Filter Manager to select or create a new filter.
- Builder - Create criteria for inclusion and exclusion of events based on boolean algebra.

This rule allows you to:

- Count the number of times certain conditions are met within a specific timeframe.
- Incorporate all of the features of the simple correlation rule, as well as evaluate events against past events.

For example, an Advanced Correlation rule can look for events from the same source IP address to the same destination IP address with the same event name that occur both inside and outside a firewall (meaning the attack may have made it through the firewall).

For more information, see Creating an Advanced Correlation Rule.

Free Form RuleLg Correlation

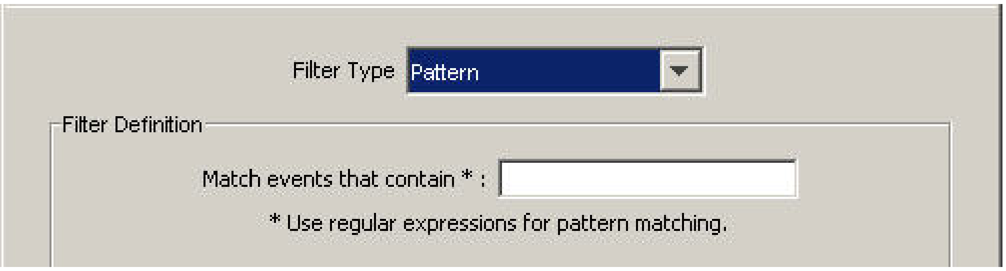
The RuleLg correlation rule definition language allows you to have complete control in defining correlation rules. Before you use this correlation rule type, you should be familiar with the RuleLg correlation rule definition language.

For more information, see Creating a Free Form RuleLg Correlation Rule.

Creating a Watchlist Rule

Create a Watchlist Rule when you want to specify a string that the Correlation Engine will watch for in every incoming event. To create a Watchlist Rule:

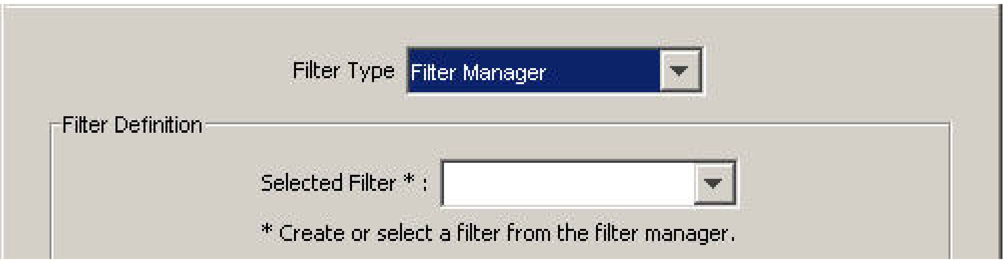
- Select Watchlist Rule in the first Correlation Rule Wizard window. Complete information for:
  - Rule Name - name that will appear in the list of rules. Maximum number of characters is 255 with no periods allowed. Extended ASCII characters are not allowed. The Rule name is case-sensitive.
  - Description – Brief description. Maximum length of the descriptive text is 1024 characters.
- Filter Type
  - Allow All -
  - Pattern – Watch for event that contains *



Filter Type: Pattern

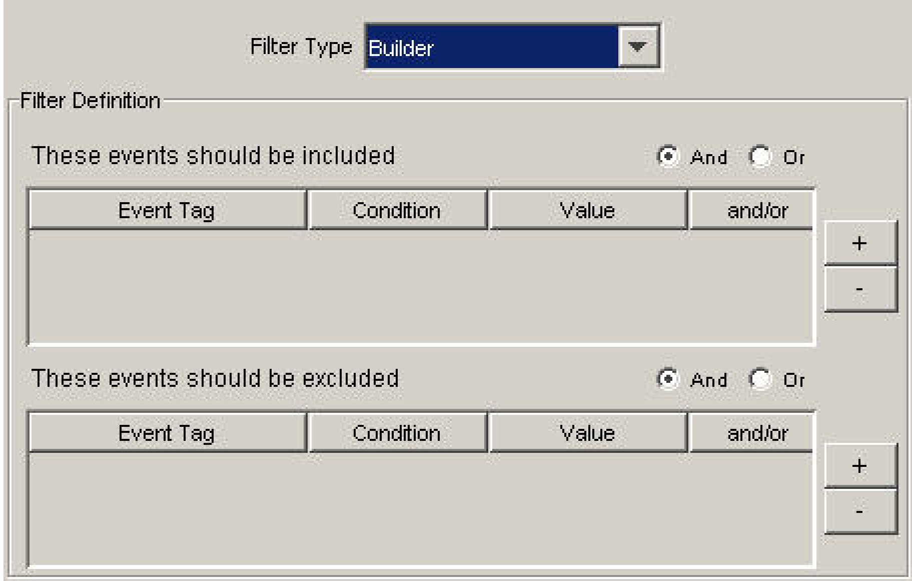
Filter Definition
Match events that contain * :
* Use regular expressions for pattern matching.

  - Filter Manager - ({ownerid}:{Filter name}:<Field name>

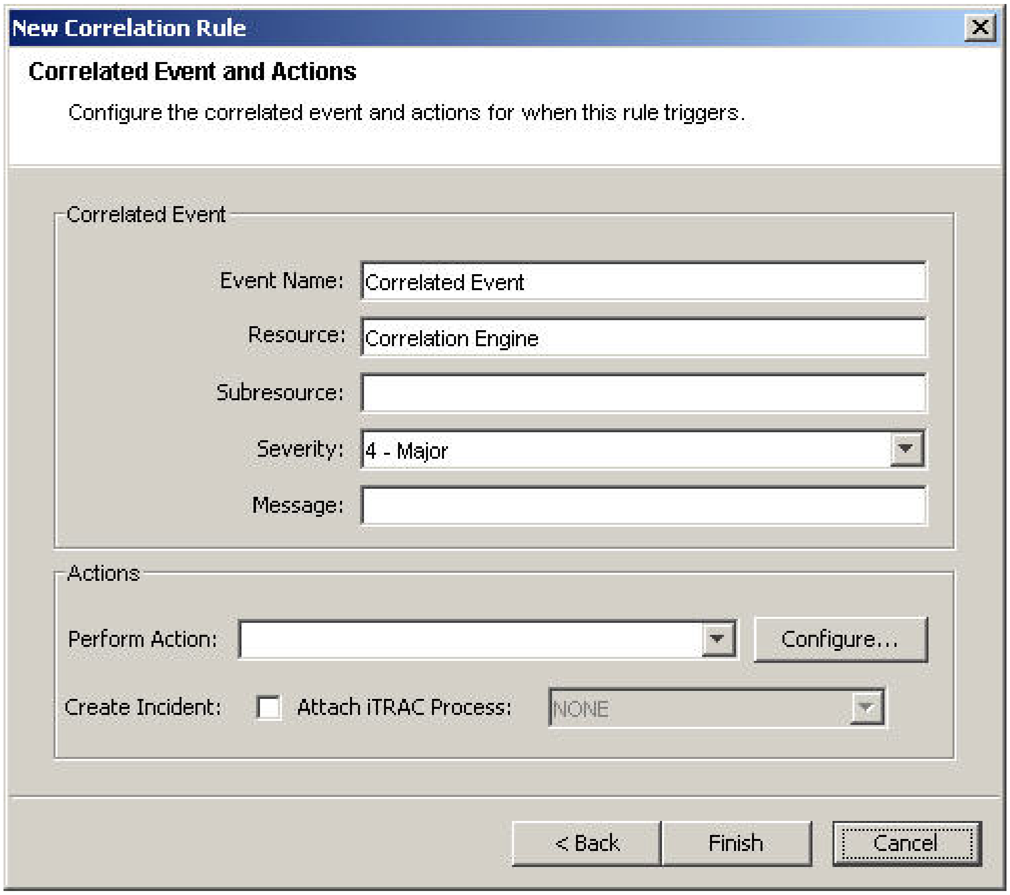


Filter Type: Filter Manager

  - Builder



- Correlated Event and Actions Page - This panel defines what action will automatically be taken when events match this correlation rule. The only required entry is the severity level, which by default is level 4.
  - Event Name – Default: Correlated Event. This is the text name of the correlated event.
  - Resource - Default: Correlation Engine. This is the text name of a resource in the system.
  - Subresource - Default: <none>. This is the subresource name for resources with multiple sub resources.
  - Set Severity level to - Default: 4, this is the severity level the event will be assigned. Valid values are 0, 1, 2, 3, 4 (default) and 5. A dropdown list is provided with valid severity levels.
  - Custom Message text - Default: <none>. This is the text that will appear along with the event. It is useful in identifying the condition that triggered the Watchlist Rule. Maximum number of characters is 4,000. The text you enter in this box is pre-pended to the text of the correlation event with a pipe separator. For example, input of "New message" would result in a correlated message of "New message|Three instances of....".
  - Perform Action (Oracle only) - Default: <none>. This is the name of an executable file that is run upon triggering the Watchlist rule. The file must be in the $ESEC_HOME/sentinel/exec directory and be executable by the esecadm user. There is no input validation in this free form text box. You can specify the meta-tags you want to send to the executable.
  - Perform Action (MSSQL Only) - Default: <none>. This is the name of an executable file that is run upon triggering the rule. The file must be in the %ESEC_HOME%\sentinel\bin directory and be executable by the esecadm user. There is no input validation. You can specify the meta-tags you want to send to the executable. Below are two examples of a correlation rule sending an email and a correlation rule sending the correlation event to HP OVO.



Correlated Event and Actions

Configure the correlated event and actions for when this rule triggers.

Correlated Event

Event Name: Correlated Event

Resource: Correlation Engine

Subresource:

Severity: 4 - Major

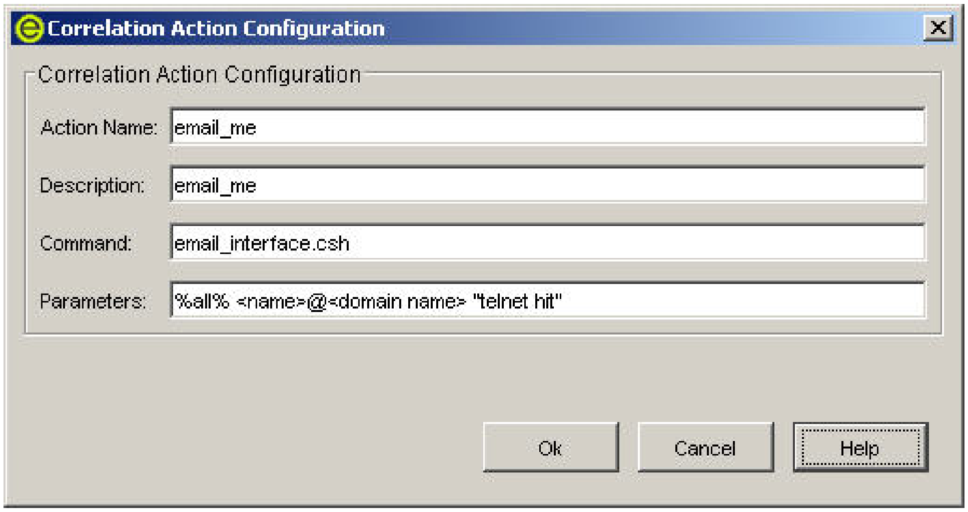
Message:

Actions

Perform Action:

Create Incident: ☐ Attach iTRAC Process: NONE

The command line and parameter line is fed in as a string. In its parsing it applies the same rules that \ (back slash) is an escape character. It can be used to escape the \, % and the " characters. For example, \%\"\\ is equivalent to %"\. If you need a command that contains a back slash, i.e. to execute a Windows command in a sub-directory of sentinel\bin, enter two back slashes (\\) for each directory slash. For example to execute a batch file named run.bat under %esec_home%\sentinel\bin\batchfiles\ you would enter batchfiles\\run.bat. Remember all executables must be below %esec_home%\sentinel\bin\.



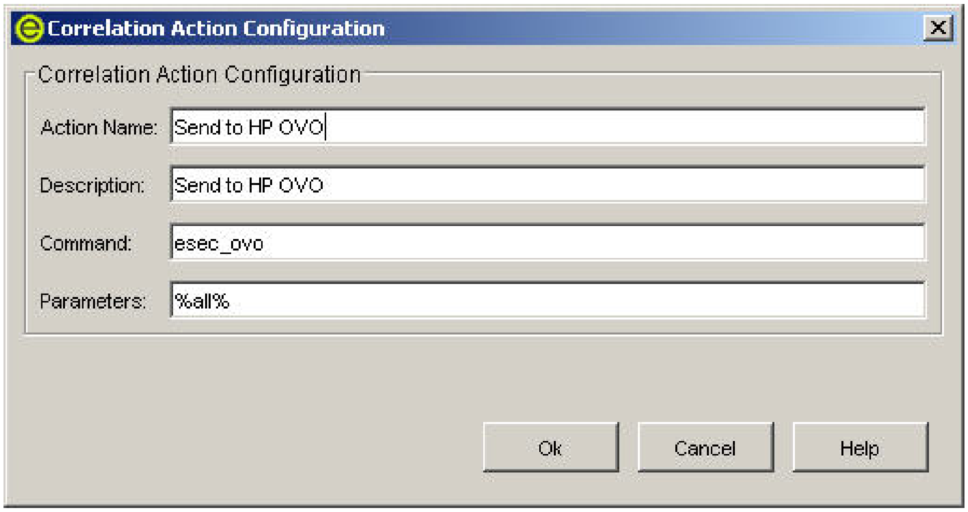
Correlation Action Configuration

Action Name: email_me

Description: email_me

Command: email_interface.csh

Parameters: %all% <name>@<domain name> "telnet hit"


Correlation Action Configuration
Correlation Action Configuration

Action Name: Send to HP OVO

Description: Send to HP OVO

Command: esec_ovo

Parameters: %all%

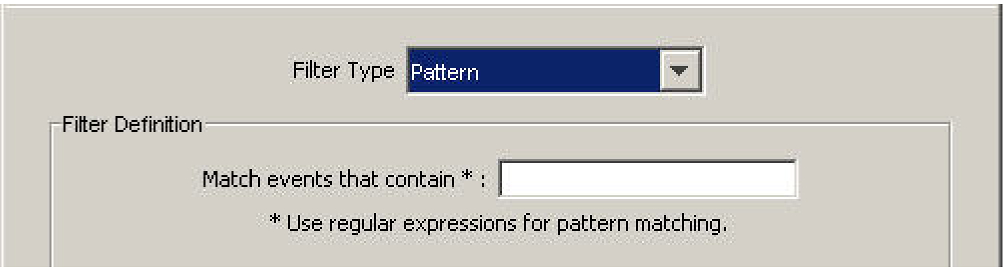
NOTE: For more information about Commands and Parameters, see Chapter 5 – Wizard and Sentinel Meta Tags in the User Reference Guide and Correlation Output Section.

- Create Incident – An action of the correlated event can also be the creation of an incident.
- Attach iTrac Process – The created incident can have attached to it an iTrac process.

Creating a Basic Correlation Rule

Create a basic correlation rule when you want to count the number of times certain conditions are met within a time frame. The steps involved are:

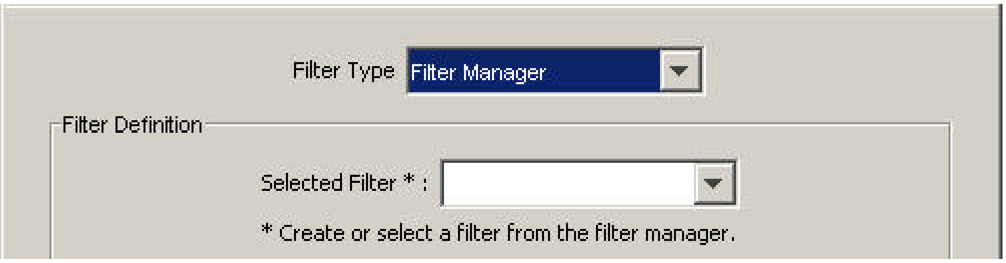
- Select Basic Correlation Rule in the first Correlation Rule Wizard window. Complete information for:
    - Rule Name - name that will appear in the list of rules. Maximum number of characters is 255 with no periods allowed. Extended ASCII characters are not allowed. The Rule name is case-sensitive.
    - Description – Brief description. Maximum length of the descriptive text is 1024 characters.
- Filter Type
    - Allow All
    - Pattern



Filter Type: Pattern

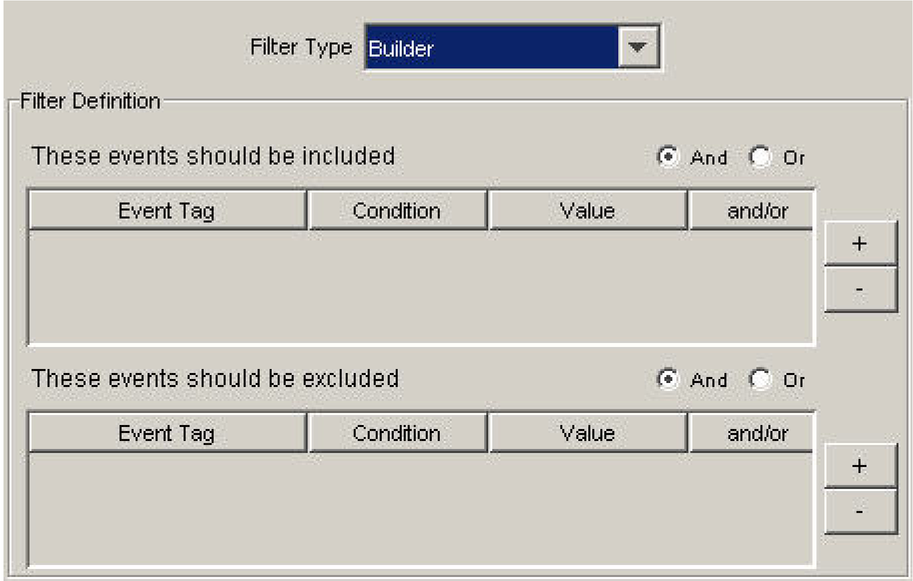
Filter Definition
Match events that contain * :
* Use regular expressions for pattern matching.

    - Filter Manager - ({ownerid}:{Filter name}:<Field name>



Filter Type: Filter Manager

  - Builder

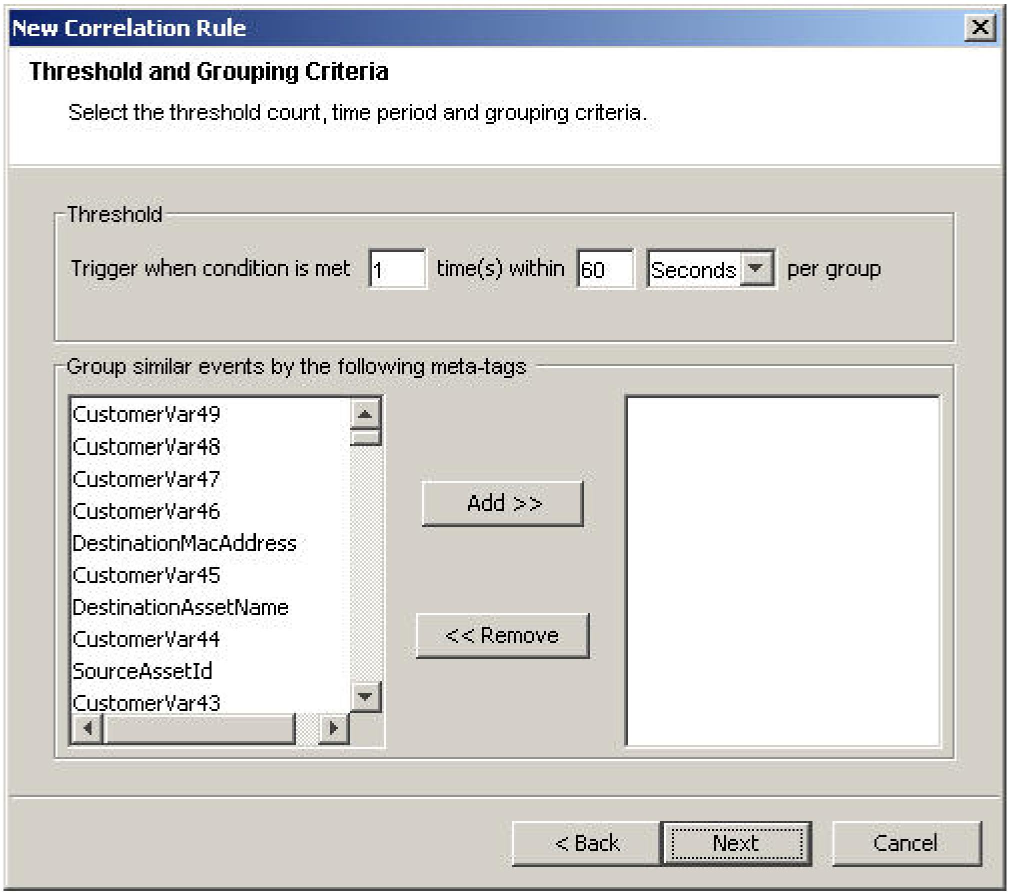


Filter Type: Builder

- Threshold and Grouping Criteria (top half of window) - Activate Rule: - This option allows you to input "match" criteria for multiple events that enter the system over a given time period.
  - When condition is met _times - Default: 1. A rule is triggered only after it has been detected the number of times specified. Valid range of inputs for this threshold value is 1 or greater.
  - within (timeframe) - Default: 60 seconds. This will bound the condition to the timeframe. This is a combined variable input and a drop-down list. The options of the drop-down list are: second, minutes, hours and days.

NOTE: When the time frame is 0, the trigger is considered to be instantaneous. For Basic Correlation, the event will happen at most 1 time for an instantaneous time frame.

- Threshold and Grouping Criteria Page (bottom half of window) - Correlates on distinct combinations of the following meta-tags - Select the meta-tags to use in combination to correlate on. Events are placed into groups based on the selected meta-tags.



New Correlation Rule

Threshold and Grouping Criteria

Select the threshold count, time period and grouping criteria.

Threshold

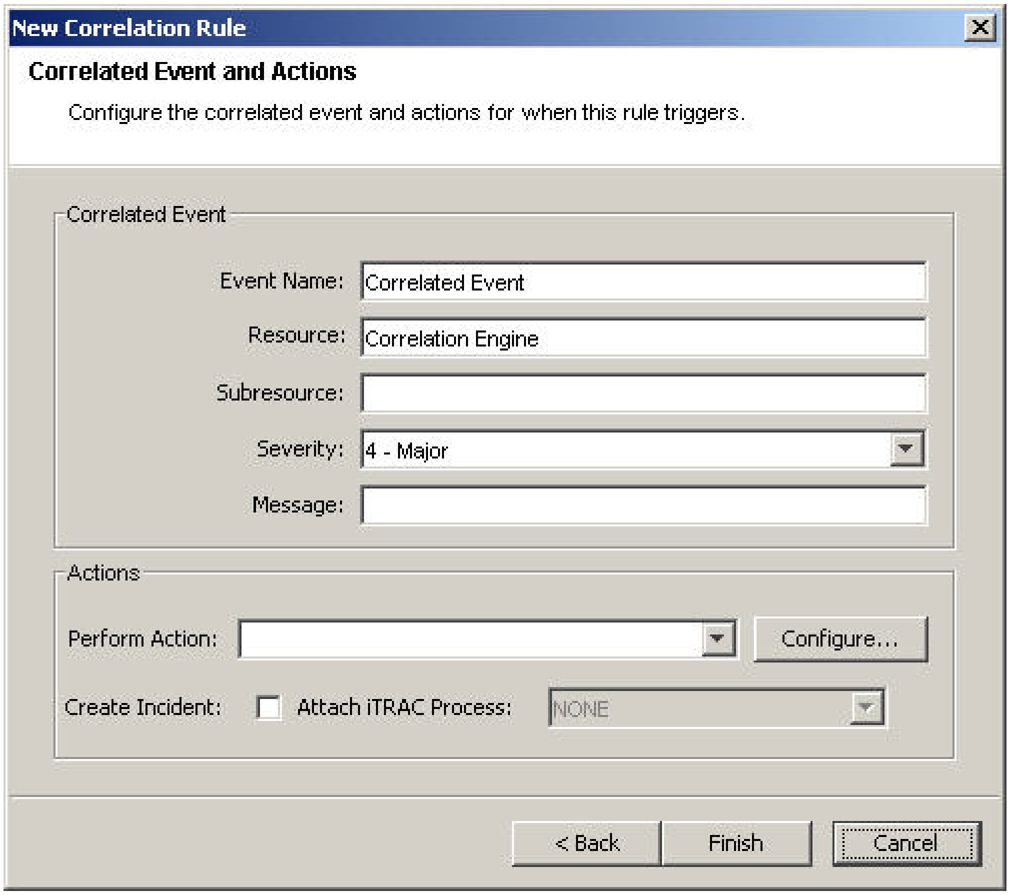
Trigger when condition is met 1 time(s) within 60 Seconds per group

Group similar events by the following meta-tags

CustomerVar49
CustomerVar48
CustomerVar47
CustomerVar46
DestinationMacAddress
CustomerVar45
DestinationAssetName
CustomerVar44
SourceAssetId
CustomerVar43

- Correlated Event and Actions Page - This panel defines what action will automatically be taken when events match this correlation rule. The only required entry is the severity level, which by default is level 4.
  - Event Name – Default: Correlated Event. This is the text name of the correlated event.
  - Resource - Default: Correlation Engine. This is the text name of a resource in the system.
  - Subresource - Default: <none>. This is the subresource name for resources with multiple sub resources
  - Set Severity level to - Default: 4. This is the severity level the event will be assigned. Valid values are 0, 1, 2, 3, 4 (default) and 5. A dropdown list is provided with valid severity levels.
  - Custom Message text - Default: <none>. This is the text that will appear along with the event. It is useful in identifying the condition that triggered the Watchlist Rule. Maximum number of characters is 4,000. The text you enter in this box is pre-pended to the text of the correlation event with a pipe separator. For example, input of "New message" would result in a correlated message of "New message|Three instances of....".
  - Run this command (Oracle only) - Default: <none>. This is the name of an executable file that is run upon triggering the Watchlist rule. The file must be in the $ESEC_HOME/sentinel/exec directory and be executable by the esecadm user. There is no input validation in this free form text box. You can specify the meta-tags you want to send to the executable.

- Perform Action (MSSQL Only) - Default: <none>. This is the name of an executable file that is run upon triggering the rule. The file must be in the %ESEC_HOME%\sentinel\bin directory and be executable by the esecadm user. There is no input validation. You can specify the meta-tags you want to send to the executable. Below are two examples of a correlation rule sending an email and a correlation rule sending the correlation event to HP OVO.



Correlated Event and Actions

Configure the correlated event and actions for when this rule triggers.

Event Name: Correlated Event

Resource: Correlation Engine

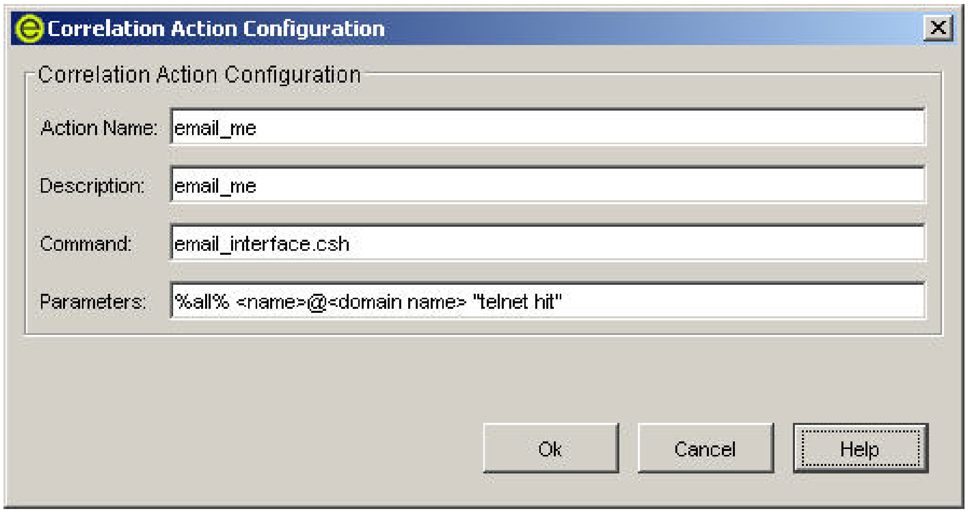
Subresource:

Severity: 4 - Major

Message:

Perform Action:

Create Incident: ☐ Attach iTRAC Process: NONE

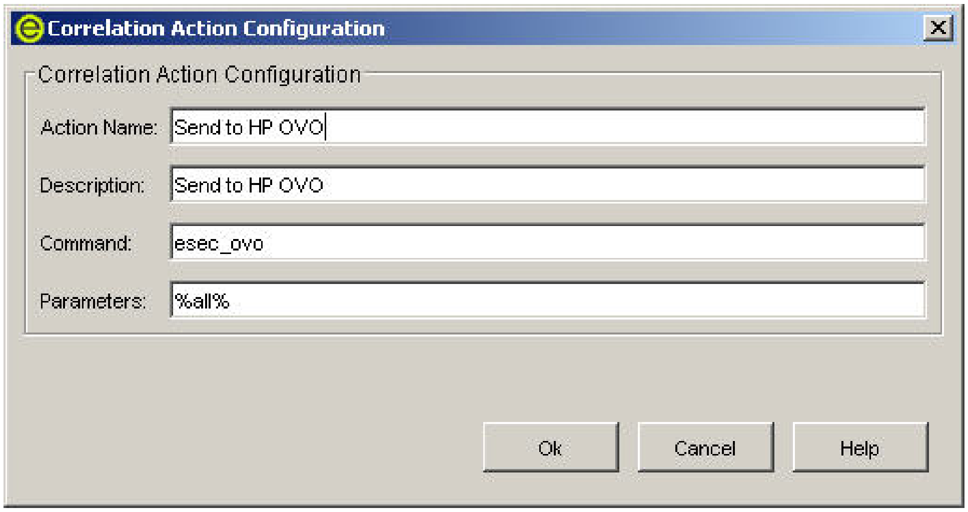

Correlation Action Configuration
Correlation Action Configuration
Ok
Cancel
Help

Action Name: email_me

Description: email_me

Command: email_interface.csh

Parameters: %all% <name>@<domain name> "telnet hit"


Correlation Action Configuration
Correlation Action Configuration
Ok
Cancel
Help

Action Name: Send to HP OVO
Description: Send to HP OVO
Command: esec_ovo
Parameters: %all%

NOTE: For more information about Commands and Parameters, see Chapter 5 – Wizard and Sentinel Meta Tags in the User Reference Guide and Correlation Output Section.

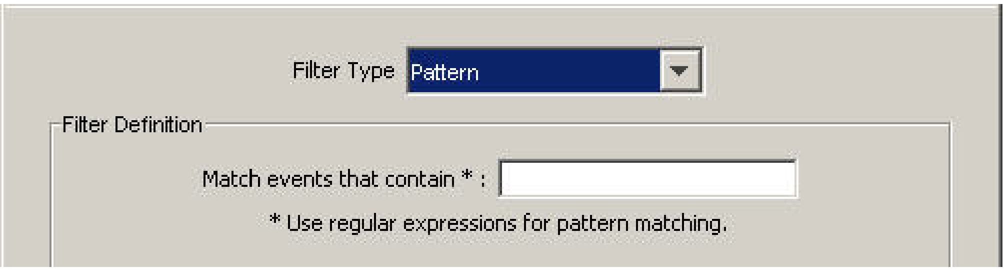
- Create Incident – An action of the correlated event can also be the creation of an incident.
- Attach iTrac Process – The created incident can have attached to it an iTrac process

Creating an Advanced Correlation Rule

An Advanced Correlation Rule allows you to add more complexity to the rule by adding an additional condition in the Additional Criteria window, in essence adding a level of logical ANDing to the rule definition.

Create an advanced correlation rule when you want to not only count the number of times that certain conditions are met but also want to receive an alert when events also satisfy criteria involving past events. The steps involved are:

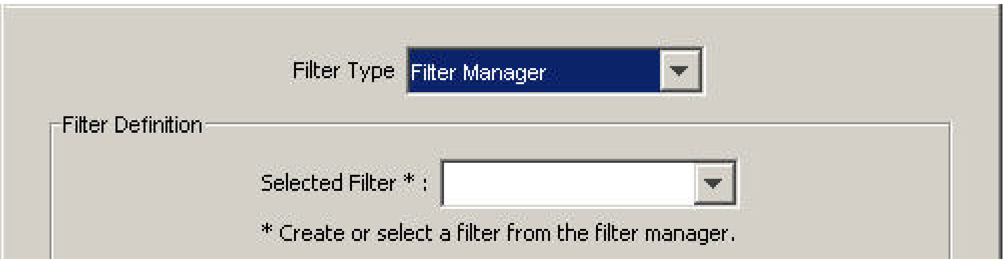
- Select Advanced Correlation Rule in the first Correlation Rule Wizard window. Complete information for:
    - Rule Name - name that will appear in the list of rules. Maximum number of characters is 255 with no periods allowed. Extended ASCII characters are not allowed. The Rule name is case-sensitive.
    - Description – Brief description. Maximum length of the descriptive text is 1024 characters.
- Filter Type
    - Allow All
    - Pattern



Filter Type: Pattern

* Use regular expressions for pattern matching.

  - Filter Manager - ({ownerid}:{Filter name}:<Field name>



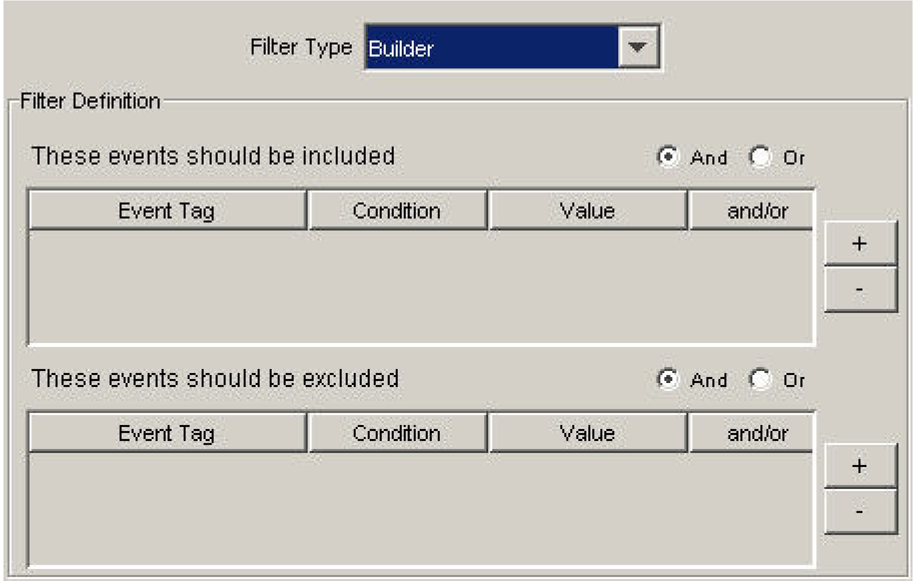
Filter Type: Filter Manager

Filter Definition

Selected Filter * :

* Create or select a filter from the filter manager.

  - Builder



Filter Type: Builder

Filter Definition

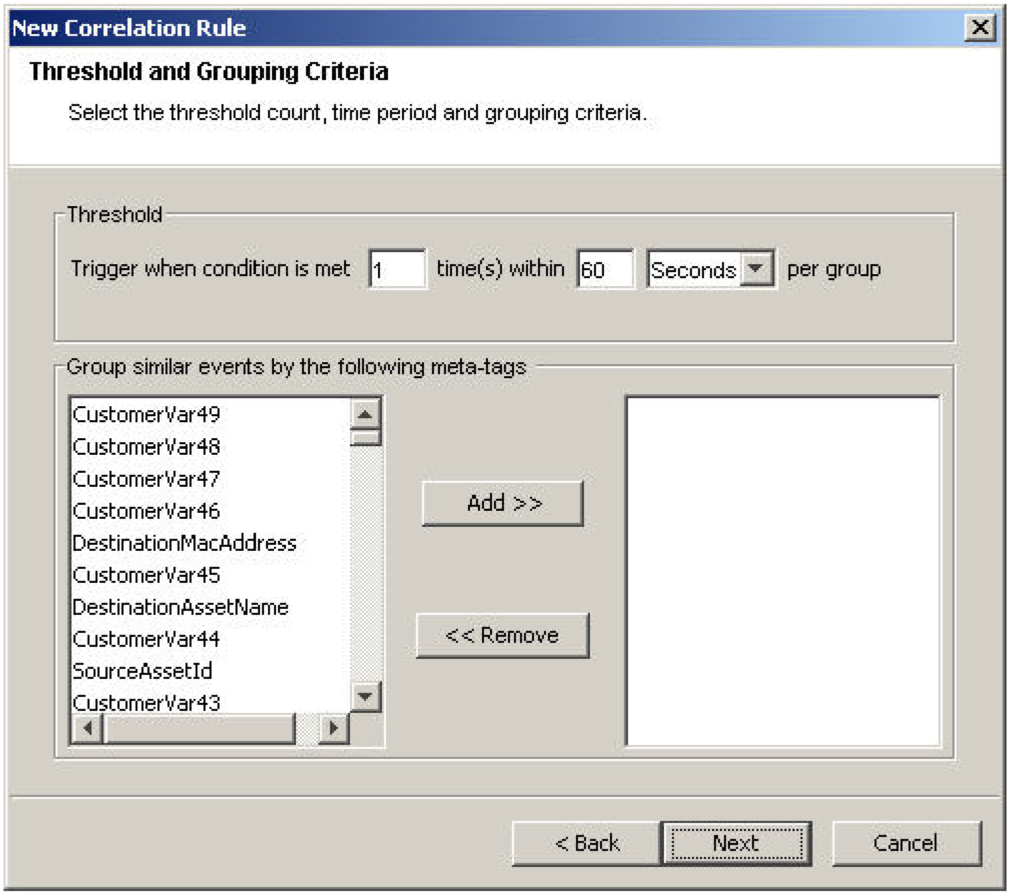
These events should be included ● And ○ Or

These events should be excluded ● And ○ Or

- Additional Criteria - This option allows you to input "match" criteria for multiple events that enter the system over a given time period. The default time is 60 seconds. This is a combined variable input and a drop-down list. The options of the drop-down list are: second, minutes, hours and days.
- Threshold and Grouping Criteria (top half of window) - Activate Rule: - This option allows you to input "match" criteria for multiple events that enter the system over a given time period.
  - When condition is met _times - Default: 1. A rule is triggered only after it has been detected the number of times specified. Valid range of inputs for this threshold value is 1 or greater.
  - within (timeframe) - Default: 60 seconds. This will bound the condition to the timeframe. This is a combined variable input and a drop-down list. The options of the drop-down list are: second, minutes, hours and days.

NOTE: When the time frame is 0, the trigger is considered to be instantaneous. For Basic Correlation, the event will happen at most 1 time for an instantaneous time frame.

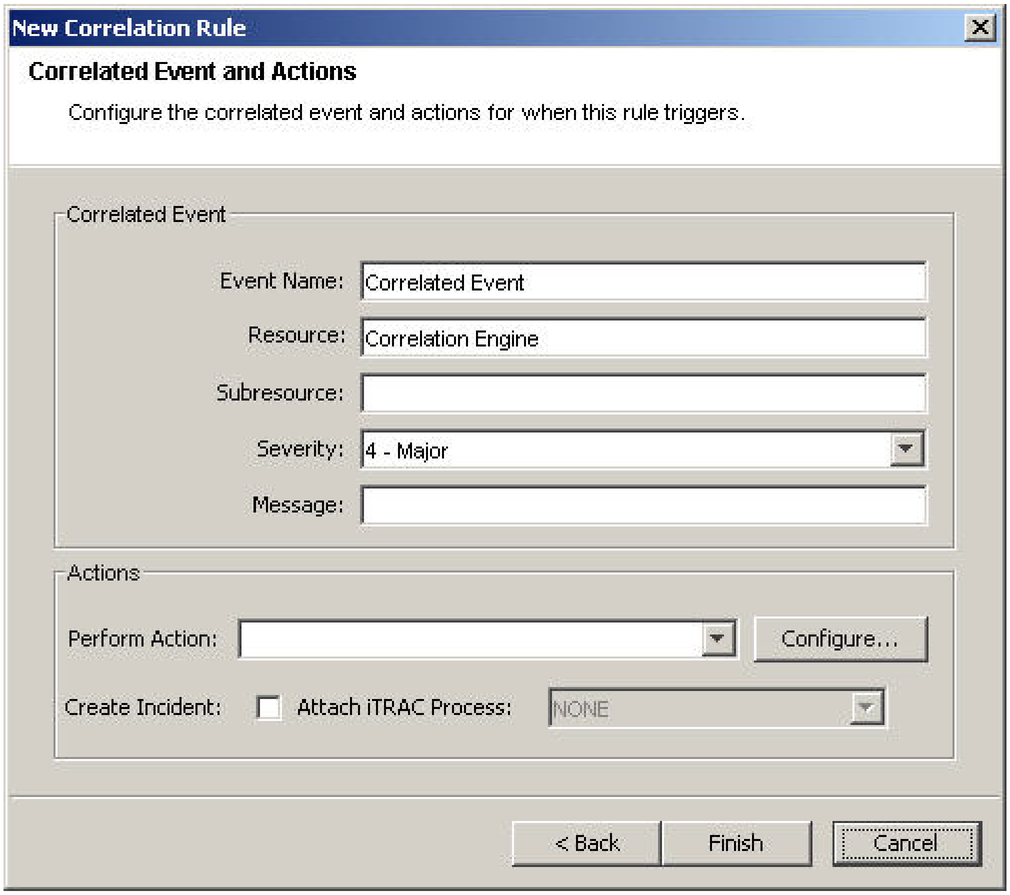
- Threshold and Grouping Criteria Page (bottom half of window) - Correlate on distinct combinations of the following meta-tags - Select the meta-tags to use in combination to correlate on. Events are placed into groups based on the selected meta-tags.



Trigger when condition is met: 1 time(s) within: 60 Seconds per group

- Correlated Event and Actions Page - This panel defines what action will automatically be taken when events match this correlation rule. The only required entry is the severity level, which by default is level 4.
  - Event Name – Default: Correlated Event. This is the text name of the correlated event.
  - Resource - Default: Correlation Engine. This is the text name of a resource in the system.
  - Subresource - Default: <none>. This is the subresource name for resources with multiple sub resources
  - Set Severity level to - Default: 4. This is the severity level the event will be assigned. Valid values are 0, 1, 2, 3, 4 (default) and 5. A dropdown list is provided with valid severity levels.
  - Custom Message text - Default: <none>. This is the text that will appear along with the event. It is useful in identifying the condition that triggered the Watchlist Rule. Maximum number of characters is 4,000. The text you enter in this box is pre-pended to the text of the correlation event with a pipe separator. For example, input of "New message" would result in a correlated message of "New message|Three instances of....".
  - Run this command (Oracle only) - Default: <none>. This is the name of an executable file that is run upon triggering the Watchlist rule. The file must be in the $ESEC_HOME/sentinel/exec directory and be executable by the esecadm user. There is no input validation in this free form text box. You can specify the meta-tags you want to send to the executable.

- Perform Action (MSSQL Only) - Default: <none>. This is the name of an executable file that is run upon triggering the rule. The file must be in the %ESEC_HOME%\sentinel\bin directory and be executable by the esecadm user. There is no input validation. You can specify the meta-tags you want to send to the executable. Below are two examples of a correlation rule sending an email and a correlation rule sending the correlation event to HP OVO.



New Correlation Rule

Correlated Event and Actions

Configure the correlated event and actions for when this rule triggers.

Correlated Event

Event Name: Correlated Event

Resource: Correlation Engine

Subresource:

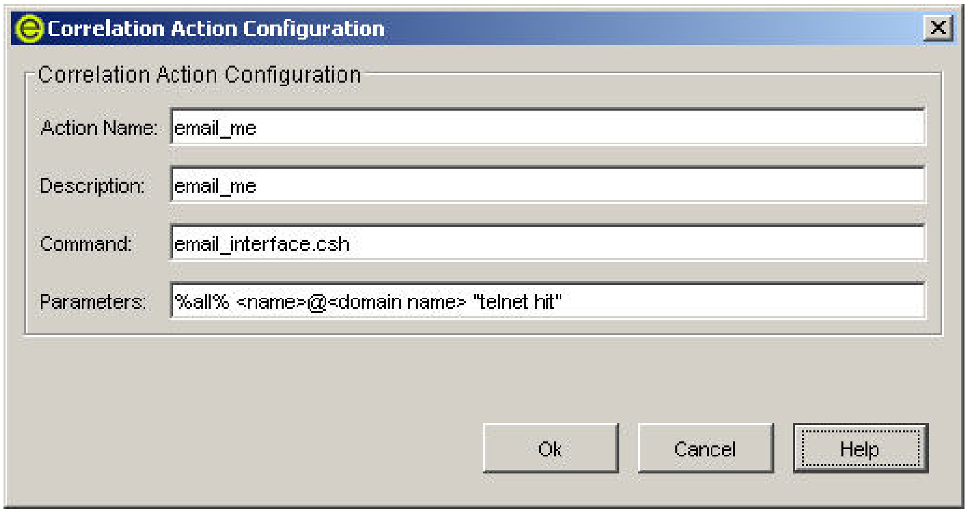
Severity: 4 - Major

Message:

Actions

Perform Action:

Create Incident: ☐ Attach iTRAC Process: NONE



Correlation Action Configuration

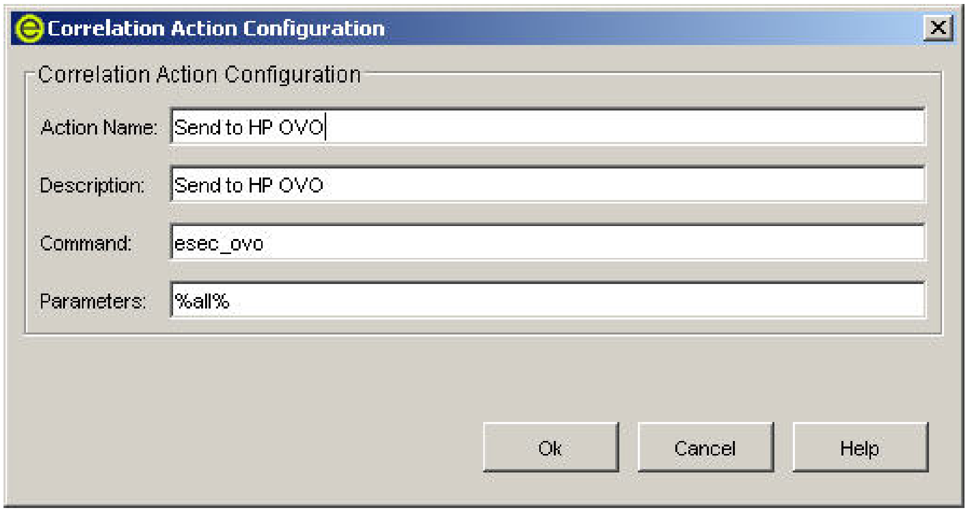
Correlation Action Configuration

Action Name: email_me

Description: email_me

Command: email_interface.csh

Parameters: %all% <name>@<domain name> "telnet hit"



Correlation Action Configuration

Action Name: Send to HP OVO

Description: Send to HP OVO

Command: esec_ovo

Parameters: %all%

NOTE: For more information about Commands and Parameters, see Chapter 5 – Wizard and Sentinel Meta Tags in the User Reference Guide and Correlation Output Section.

- Create Incident – An action of the correlated event can also be the creation of an incident.
- Attach iTrac Process – The created incident can have attached to it an iTrac process.

Creating a Free Form RuleLg Correlation Rule

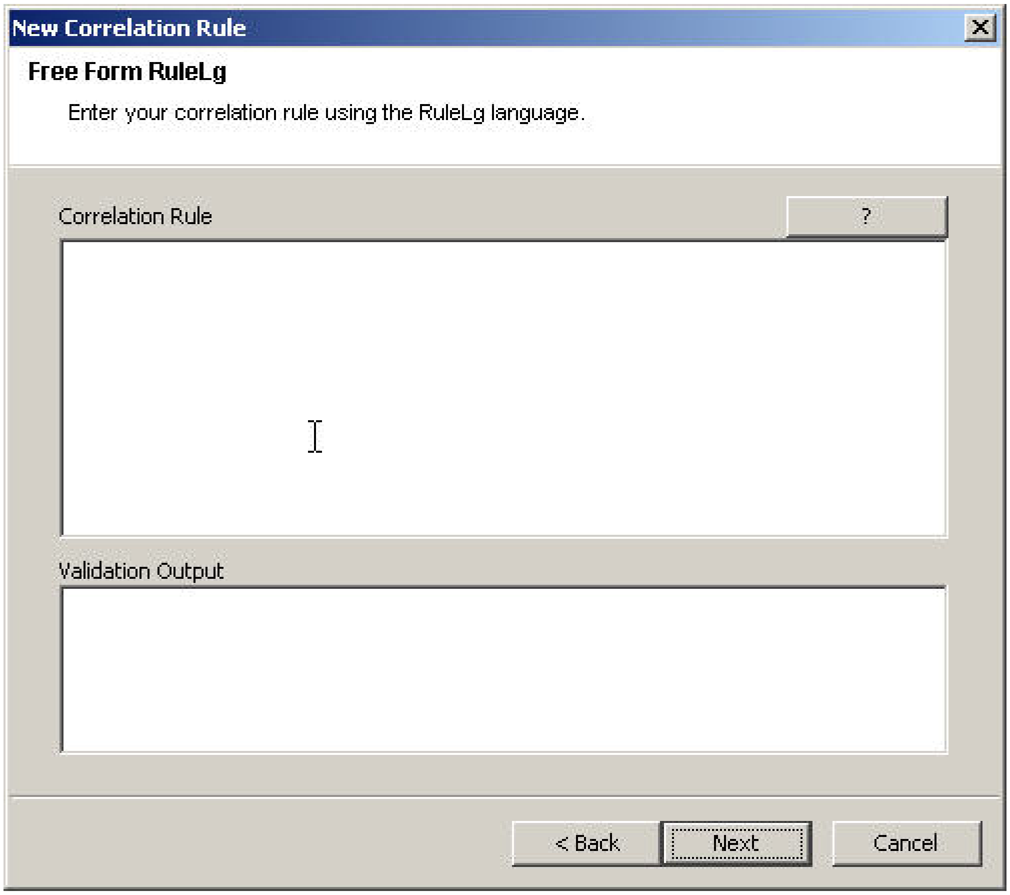
The Correlation Engine is built around three fundamental operations. These operations are combined to form a rule with flow, union and intersection operators. The three fundamental operations are:

- Filter operation
- Window operation
- Trigger operation

CAUTION: If you renamed a tag, do not use the original name when creating a correlation rule.

The rule language directly reflects these operations and how they can be combined in an intuitive fashion to define correlation rules. Each operation has been specifically designed and implemented for high performance and works on a set of events: receiving a set of events as input and returning a set of events. The current event processed by a rule often has a special meaning for the semantic of the language. The current event is always part of the set of events in and out of an operation unless the set is empty. If an input set of an operation is empty, then the operation is not evaluated.

From a simplified point of view, a correlation rule processes the events coming into the Correlation Engine in a serialized fashion, one by one. In reality, the Correlation Engine is able to process multiple events and evaluate multiple rules against one event concurrently.



Filter Operation

A filter (boolean expression) operation allows filtering according to the content of the current event; that is, its meta-tag values and the filter-specified boolean expression. The output of a filter is either the empty set (if the current event did not match the filter) or a set containing the current event and all of the other events from the incoming set.

- Filters operate on the current event, evaluating the boolean expression for the current event:
  - The filter operation returns the input set if the boolean expression evaluates to true.
  - The filter operation returns the empty set if the boolean expression evaluates to false.
- The boolean expression is a composite of comparison instructions and match instructions with the boolean operators 'and', 'or' and 'not'.

Filter Operation - RuleLg Operator Precedence and Associations

Filter boolean operator precedence (from highest [top] to lowest [bottom] precedence):

Operator	Meaning	Operator Type	Associativity
not	logical not	unary	none
and	logical and	binary	left to right
or	logical or	binary	left to right

The following apply:

- Comparison instructions allow evaluation of event meta-tag values with other event meta-tag values or constraints.
- Available comparison operators are =, !=, >, <, >=, <=.
- The available match instructions are match regular expressions, match regex(), or match subnets, match subnet().
- Comparison and match instructions may be nested using parentheses to any depth.
- The meta-tag names in comparison and match instructions must always be prepended with "e." to specify the current event.
- If a filter is the last or only operation of a correlation rule, then the output set of the filter is used to construct a correlated event (the correlated events being the filter operation output set of events with the current event first).
- If a filter is not the last operation of a correlation rule (that is, a flow operator is on its right), then the output set of a filter is used as the input set to other operations (through the flow operator).

For example: if the current event has a severity of 4 and the resource meta-tag contains either "FW" or "Comm", then a correlated event is sent with the current event (single event) listed as the correlated event.

```
filter(e.sev = 4 and (e.res match regex ("FW") or
  e.res match regex ("Comm")))
```

Another example is if any of the current event meta-tags contain "ABC", a correlated event is sent with the current event (single event) listed as the correlated event.

```
filter(e.all match regex("ABC"))
```

Window Operation

A window (simple boolean expression[, filter expression], int duration) operation works on the current event in relation to a window of past events. Past events are maintained by the window operation itself. The output of a window is either the empty set (if the current event did not match the simple boolean expression) or a set containing the current event and all of the past events for which the simple boolean expression is true.

The simple boolean expression is either a single comparison instruction or a single match instruction of a past event meta-tag value with either a current event meta-tag value or a constant. For boolean expressions:

- You must prepend a meta-tag name with "e." to specify the current event or with "w." to specify the past events
- Available comparison operators are =, !=, >, <, >=, <=, in and not in
- The available match instructions are match regular expressions, match regex(), or match subnets, match subnet()
- A "w.[meta-tag]" must be present in a window simple boolean expression
- If any past event evaluates to true with the current event for the simple boolean expression, the output set is the incoming event plus all matches in the window

- If no events in the window match the current event for the simple boolean expression, an empty set is output

Past events are maintained for the specified duration of the window operation.

The optional filter expression parameter of a window allows you to control what events the window maintains. This expression is any valid filter.

- Every event coming in to the Correlation Engine that passes this filter is put into the window of past events
- If no filter expression exists, then all events coming into the Correlation Engine are maintained by the window
- The current event is not placed into the window until after the current event window evaluation is complete
- Only the relevant parts of the past events are actually maintained by the window (to lessen memory usage)

If a window is the last or only operation, of a correlation rule, then the output set of the window is used to construct a correlated event (the correlated events being the window operation output set of events with the current event first).

Example 1

```
window(e.sip = w.sip, filter(e.sip match subnet
   (<xxx.xxx.x.x/yy>)), 60)
```

In the above example, if the current event has a source IP address in the specified by address xxx.xxx.x.x/yy with CIDR subnetmask and matches an event(s) that happened within the past 60 seconds, a correlated event is sent with the current event and any matched past events as the correlated events (current event first).

Example 2

```
window(e.sip = w.dip, 3600) intersection
window(e.dp = w.dp, 3600) intersection
window(e.evt = w.evt, 3600)
```

The above is a domino type of rule. An attacker exploits a vulnerable system and uses it as an attack platform.

Example 3

```
filter(e.sev > 3) flow (window(e.sip = w.sip, filter
   (e.sev >3), 5) intersection window(e.evt = w.evt,
   filter(e.sev >3), 5) intersection window(e.dip =
   w.dip, filter(e.sev >3), 5) intersection
   window(e.sn! = w.sn, filter(e.sev > 3),5)
```

The above example is an inside/outside type of rule. An attack signature is seen on two intrusion detection systems, one on the inside of a firewall and the other on the outside and the attack has a severity greater than 3.

Trigger Operation

A trigger operation's main purpose is to count a number of events for a specified duration. If the specified count is reached within the specified duration, then a set of events containing all of the events maintained by the trigger is output; if not the empty set is output.

- The trigger operation receives as input a set of events to be returned as part of the output set of events if the specified count, duration and discriminator(s) of the previous input sets and the current input set meet the criteria defined by the trigger operation.
- The count is an integer value specifying the number of events that need to occur within the duration window to output a non-empty set.
- The duration is an integer value in seconds specifying the duration events are maintained by the trigger operation.
- If duration is equal to zero, a trigger operation just compares the number of events in the input set with count and outputs the current event if this number is greater than or equal to count.
- When receiving a new input set of events, a trigger first discards the outdated events (events that have been maintained for more than the duration) and then inserts the current event. If the number of resulting events is greater than or equal to the specified count, then the trigger outputs a set containing all of the events.
- If a trigger is the last operation (or the only operation) of a correlation rule, then the output set of the trigger is used to construct a correlated event (the correlated events being the trigger operation output set of events with the current event first).
- If a trigger is not the last operation of a correlation rule (that is, a flow operator is to its right), then the output set of a trigger is used as the input set to other operations (through the flow operator).
- After a first time the trigger operation criteria are met (and, therefore, the trigger operation outputs a set of events), if the criteria are met again containing at least one of the previously outputted events and the trigger is the last operation (or the only operation), then the Correlation Engine does not construct a new correlated event, but instead constructs an update to the previous correlated event.
- The discriminator (meta-tag list) is a comma-delimited list of meta-tags. A trigger operation keeps different counts for each distinct combination of the discriminator meta-tags.

For example, if 5 events with the same source IP address happen within 10 seconds, send a correlated event with the five events as the correlated events (current event first).

```
trigger(5,10,discriminator(e.sip))
```

Though using the free-form rule option allows you to create expressions of unbounded complexity, these rules may not make sense. The supported normal form of a RuleLg expression is broken into three parts: the filter section, the window section and the trigger section. The three sections are connected with a flow operator.

The filter section may contain multiple connected filters.

Example:

```
(filter(e.sev = 5) union filter(e.sev =4))
(filter(e.sev = 5 or e.sev =4))
```

NOTE: This section is optional. When omitted it is equivalent to filter(1=1).

The window section may contain multiple intersecting windows.

Example:

(window(w.sev = e.sev,10) intersection window(w.sip = e.sip,10))

NOTE: This section is optional.

The trigger section may contain one trigger operation.

Example

```
(trigger(5,10))
```

NOTE: This section is optional. When omitted the rule behaves as if it ends with trigger(1,0).

Operators That Combine With Operations to Form Rules

The operators that combine with operations to form rules are:

- Flow operator
- Union operator
- Intersection operator

Filter, window and trigger operation operator precedence (from highest (top) to lowest (bottom)) are:

Operator	Meaning	Operator Type	Associativity
flow	Output set becomes input set	binary	left to right
intersection	set intersection (remove duplicates)	binary	left to right
union	set union (remove duplicates)	binary	left to right

Flow Operator

The output set of events of the left-hand side operation is the input set of events for the right-hand side operation.

For example:

```
filter(e.sev = 5) flow trigger(3, 60)
```

The output of the filter operation is the input of the trigger operation. The trigger only counts events with severity equal to 5.

Union Operator

The union of the left side operation output set and the right side operation output set. The resulting output set contains events from either the left-hand side operation output set or the right-hand side operation output set without duplicates.

For example:

```
filter(e.sev = 5) union filter(e.sip = 192.168.0.1)
```

is equivalent to

```
filter(e.sev = 5 or e.sip = 192.168.0.1)
```

Intersection Operator

The intersection of the left side operation output set and the right side operation output set. The resulting output set contains events that are common in both the left-hand side operation output set and the right-hand side operation output set without duplicates.

For example:

```
filter(e.sev = 5) intersection filter(e.sip =
  192.17.16.32)
```

is equivalent to

```
filter(e.sev = 5 and e.sip = 192.17.16.32)
```

Example Correlation Rules

This document provides a set of example rule-based correlation rules, along with the pre-requisites (requirements) necessary for the rules to be effective. Your rules may be different depending upon your system configuration.

The e.rv50 through e.rv53 tags that are in the RuleLg examples correspond to mappings set in your agent mapping files. For example, if you open the windows_v2000_mapv*.csv or snort_v20_mapv*.csv file, the:

- Culture column corresponds to e.rv50
- Community columns corresponds to e.rv51
- Family column corresponds to e.rv52
- Event column corresponds to e.rv53

For example:

```
filter (e.rv52 = "Worm") flow trigger (3, 300)
```

This rule refers to NIDS taxonomy. If you look under the Family column in the snort mapping file, you will find over forty instances of the word Worm. This rule will trigger for over forty different worm attacks if they occur three times within a five minute time period.

The following attacks type example correlation rules are provided.

- Brute Force – same source and target
- Buffer Overflow - same source to same target
- Buffer Overflow – service stoppage
- Denial of Service
- Login Failures - any source to any destination
- Login Failures - same source to
- Microsoft– SQL Server
- Microsoft - NETBIOS
- Microsoft - Windows scripting
- Multiple Backdoor – different sources
- Multiple Backdoor – single source
- Trojan Horse
- UNIX - Apache Web server
- UNIX - BIND/DNS

same destination
- Microsoft - anonymous login
- Microsoft - general windows authentication
- Microsoft - IE
- Microsoft – IIS
- Microsoft - LAN Manager Authentication
- Microsoft – MDAC
- Microsoft - remote registry access
- UNIX - FTP
- UNIX - general UNIX
- UNIX - line printer daemon
- UNIX - remote procedure call
- UNIX - remote services
- UNIX - secure shell
- UNIX - sendmail
- UNIX – SNMP
- Virus Outbreak
- Worm Outbreak

Buffer Overflow Attack and Service Stoppage

This Rule will identify a potential security breach after a Buffer Overflow Attack. This rule will alert if the destination of a Buffer Overflow attack has a service stopped within 60 seconds of an attack. A host-based agent, HIDS/OS, can detect if a service is stopping. The Buffer Overflow attack may be detected by NIDS, HIDS or OS agent.

If a system has been affected by this Buffer Overflow Attack, this event should be investigated.

Rule Frequency	Rule Requirements	Rule Taxonomy
1 time	Define the following prior to implementing this rule: ▪ Network IDS Platforms that the e-Security taxonomy can translate (see NIDS Taxonomy table for more information) ▪ Host IDS Platforms that the e-Security taxonomy can translate (see HIDS & OS Taxonomy table for more information)	NIDS HIDS/OS

RuleLg for this Rule

```
filter ((e.rv51 = "Service" and e.rv52 = "Stop" ) and
   (e.st = "H")) flow window (w.dip = e.sip, filter
   (e.rv52 = "Buffer_Overflow"), 60) flow trigger(1,
   0)
```

Denial of Service Attack and Service Stoppage

This Rule will identify a potential security breach after a Denial of Service Attack. This rule will alert if the destination of a Denial of Service attack has a service stopped within 60 seconds of the attack. The service stopping is detected by a host-based agent, i.e. HIDS/OS. The Buffer Overflow attack may be detected by NIDS, HIDS or OS agents.

If a system has been affected by a Denial of Service Attack, this should be further investigated.

Rule Frequency	Rule Requirements	Rule Taxonomy
1 time	Define the following prior to implementing this rule: ▪ Network IDS Platforms that the e-Security taxonomy can translate (see NIDS Taxonomy table for more information) ▪ Host IDS Platforms that the e-Security taxonomy can translate (see HIDS & OS Taxonomy table for more information)	NIDS HIDS/OS

RuleLg for this Rule

```
filter ((e.rv51 = "Service" and e.rv52 = "Stop" ) and
  (e.st = "H")) flow window (w.dip = e.sip, filter
  (e.rv52 ="DoS" ), 60) flow trigger(1, 0)
```

Virus Outbreak Detection

This rule will identify if a known virus is attacking any system within an infrastructure.

When a virus attacks, typically a system or many systems get adversely affected requiring either a complete re-load of system and application data, or the complete loss of corporate assets. Identifying a virus while in progress can greatly reduce or prevent damage.

Rule Frequency	Rule Requirements	Rule Taxonomy
3 times in 5 minutes	Define the following prior to implementing this rule: ▪ Network IDS Platforms that the e-Security taxonomy can translate (see NIDS Taxonomy table for more information)	NIDS

RuleLg for this Rule

```
filter (e.rv52 ="Virus") flow trigger (3, 300)
```

Worm Outbreak Detection

This rule will identify if a known worm is attacking any system within an infrastructure.

When a worm attacks, typically a system or many systems get adversely affected requiring either a complete re-load of system and application data, or the complete loss of corporate assets. Identifying a worm while in progress can greatly reduce the liability of the company.

Rule Frequency	Rule Requirements	Rule Taxonomy
3 times in 5 minutes	Define the following prior to implementing this rule: ▪ Network IDS Platforms that the e-Security taxonomy can translate (see NIDS Taxonomy table for more information)	NIDS

RuleLg for this Rule

```
filter (e.rv52 = "Worm") flow trigger (3, 300)
```

Trojan Horse Detection

This rule will identify if a known Trojan horse has been planted on any system within the infrastructure.

When a Trojan horse is successful, the targeted system can be completely compromised.

Rule Frequency	Rule Requirements	Rule Taxonomy
3 times in 5 minutes	Define the following prior to implementing this rule: ▪ Network IDS Platforms that the e-Security taxonomy can translate (see NIDS Taxonomy table for more information) ▪ Host IDS Platforms that the e-Security taxonomy can translate (see HIDS & OS Taxonomy table for more information)	NIDS HIDS/OS

RuleLg for this Rule

```
filter (e.rv52 ="Trojan") flow trigger (3, 500)
```

Multiple Backdoor Attempts from a Single Source

This rule will correlate multiple attempts to insert or execute a Backdoor attack from a single source.

A Backdoor program is typically used to gain complete control of the target system and then is used to launch other attacks. Typically, this rule will identify attempts of an intruder searching for an infected system, or searching to infect a system.

Rule Frequency	Rule Requirements	Rule Taxonomy
5 times in 2 minutes	Define the following prior to implementing this rule: ▪ Network IDS Platforms that the e-Security taxonomy can translate (see NIDS Taxonomy table for more information) ▪ Host IDS Platforms that the e-Security taxonomy can translate (see HIDS & OS Taxonomy table for more information)	NIDS HIDS/OS

RuleLg for this Rule

```
filter (e.rv50 = "Attack" and e.rv52 = "Backdoor" )
   flow trigger(5, 120, discriminator (e.sip))
```

Multiple Backdoor Attempts from Different Sources

This rule will correlate multiple attempts to insert or execute a coordinated Backdoor attack from different systems targeting a single destination.

A Backdoor program is typically used to gain complete control of the target system and then is used to launch other attacks. Typically, this rule identifies that either:

- the target system has been compromised
- the attacker is attempting to exploit the compromised system
- the attacker is trying to mask themselves with a coordinated attack
- or the attacker has gained knowledge that the target is vulnerable to this type of attack. If this is the case, then this may indicate that the attacker has gained knowledge from an internal source.

Rule Frequency	Rule Requirements	Rule Taxonomy
5 times in 2 minutes	Define the following prior to implementing this rule: ▪ Network IDS Platforms that the e-Security taxonomy can translate (see NIDS Taxonomy table for more information) ▪ Host IDS Platforms that the e-Security taxonomy can translate (see HIDS & OS Taxonomy table for more information)	NIDS HIDS/OS

RuleLg for this Rule

```
filter (e.rv50 = "Attack" and e.rv52 = "Backdoor" )
  flow trigger( 5, 120, discriminator(e.dip))
```

Multiple Login Failures from Any Source to Any Destination

This rule will identify login failures to the same types of systems.

Login failures to the same type of account or system may indicate that the attacker has prior knowledge of the network and the critical systems located on the network. This should cause an alarm. The more information an attacker has, the easier it is for the attacker to find a system that is exploitable.

Rule Frequency	Rule Requirements	Rule Taxonomy
5 times in 2 minutes	Define the following prior to implementing this rule: ▪ Network IDS Platforms that the e-Security taxonomy can translate (see NIDS Taxonomy table for more information)	NIDS

RuleLg for this Rule

```
filter ((e.rv52 = "Access" or e.rv52 = "Brute_Force")
  and e.rv51 = "User" and e.rv50 = "Attack") flow
  trigger (5, 120)
```

Multiple Login Failures from Same Source to Same Destination

This rule will identify multiple login failures from the same source to the same destination.

Login failures to the same type of account or system may indicate that the attacker has prior knowledge of the network and the critical systems located on the network. This should cause an alarm. The more information an attacker has, the easier it is for the attacker to find a system that is exploitable.

Rule Frequency	Rule Requirements	Rule Taxonomy
3 times in 5 minutes	Define the following prior to implementing this rule: ▪ Network IDS Platforms that the e-Security taxonomy can translate (see NIDS Taxonomy table for more information)	NIDS

RuleLg for this Rule

```
filter ((e.rv52 = "Access" or e.rv52 = "Brute_Force")
  and e.rv51 = "User" and e.rv50 = "Attack") flow
  trigger (5, 120, discriminator (e.sip, e.dip))
```

Buffer Overflow Attack from Same Source to Same Target

This rule will identify a Buffer Overflow attack from the same source IP address to the same destination IP address.

A Buffer Overflow attack is the most common attack on the network and is used to disable a system. These types of attacks can only be blocked at the perimeter. Knowledge of an attacking system can help in blocking that system.

Rule Frequency	Rule Requirements	Rule Taxonomy
5 times in 3 minutes	Define the following prior to implementing this rule: ▪ Network IDS Platforms that the e-Security taxonomy can translate (see NIDS Taxonomy table for more information) ▪ Host IDS Platforms that the e-Security taxonomy can translate (see HIDS & OS Taxonomy table for more information)	NIDS HIDS/OS

RuleLg for this Rule

```
filter (e.rv52 ="Buffer_Overflow" ) flow trigger (5,
  180, discriminator (e.sip, e.dip))
```

Brute Force Attack Success where Source and Target Are the Same

This rule will identify a possible compromised system with a cracked password.

A constant attempt of using combinations of usernames and passwords to gain access, followed by an eventual successful login may indicate that an attacker has access via a brute force attack. If this attack is successful, the account accessed should be shut down.

Rule Frequency	Rule Requirements	Rule Taxonomy
1 time in 3 minutes	Define the following prior to implementing this rule: ▪ Network IDS Platforms that the e-Security taxonomy can translate (see NIDS Taxonomy table for more information) ▪ Host IDS Platforms that the e-Security taxonomy can translate (see HIDS & OS Taxonomy table for more information)	NIDS HIDS/OS

RuleLg for this Rule

```
filter (e.rv53="Other" and rv52="Access" e.rv51
  ="User" and e.rv50="Prob" and e.st = "H") flow
  window (w.dip = e.sip, filter (e.rv52="Brute Force"
  and e.rv50="Compromise"), 180) flow trigger(1, 180,
  discriminator(e.sip, e.dip))
```

Microsoft - Internet Information Services Attacks (IIS) Verification

This rule supports SANS Microsoft top 10 attacks on Internet Information Service (IIS) attack. If you are running Microsoft's IIS application, you may be vulnerable to attack.

Rule Frequency	Rule Requirements	Rule Taxonomy
1 time	Define the following prior to implementing this rule: ▪ Network IDS Platforms that the e-Security taxonomy can translate (see NIDS Taxonomy table for more information)	NIDS

RuleLg for this Rule

```
filter (e.rv53 = "Sans_MS_IIS") flow trigger(1,60)
```

Microsoft - Microsoft Data Access Connector (MDAC) Attack -- Remote Data Services Attack Verification

This rule supports SANS Microsoft top 10 attacks on MDAC. Using Microsoft products may leave you vulnerable to attacks. MDAC is as an underlying tool used to integrate Microsoft products.

Rule Frequency	Rule Requirements	Rule Taxonomy
1 time	Define the following prior to implementing this rule: ▪ Network IDS Platforms that the e-Security taxonomy can translate (see NIDS Taxonomy table for more information)	NIDS

RuleLg for this Rule

```
filter (e.rv53 = "Sans_MS_MDAC") flow trigger(1,60)
```

Microsoft– SQL Server Attacks – SQL Server Attack Verification

This rule supports SANS Microsoft top 10 attacks on Microsoft SQL Server. Using Microsoft SQL Server may be vulnerable to attack. There are several serious vulnerabilities that allow remote attackers to obtain sensitive information, alert database content, compromise SQL servers and compromise server hosts.

Rule Frequency	Rule Requirements	Rule Taxonomy
1 time	Define the following prior to implementing this rule: ▪ Network IDS Platforms that the e-Security taxonomy can translate (see NIDS Taxonomy table for more information)	NIDS

RuleLg for this Rule

```
filter (e.rv53 = "Sans_MS_SQLServer") flow
   trigger(1,60)
```

Microsoft - NETBIOS - Unprotected Windows Network Shares Attack Verification

This rule supports SANS Microsoft top 10 attacks on NETBIOS. Using Microsoft networking with NETBIOS may leave you vulnerable to an attack. NETBIOS was Microsoft's original networking communications software. Current Microsoft networks do not rely on NETBIOS as a transport medium.

Rule Frequency	Rule Requirements	Rule Taxonomy
1 time	Define the following prior to implementing this rule: ▪ Network IDS Platforms that the e-Security taxonomy can translate (see NIDS Taxonomy table for more information)	NIDS

RuleLg for this Rule

```
filter (e.rv53 = "Sans_MS_NETBIOS") flow trigger(1,60)
```

Microsoft - Anonymous Login - Null Sessions Attack Verification

This rule supports SANS Microsoft top 10 attacks on Null Sessions. If you are using Microsoft Null Session, you may be vulnerable to attack. The anonymous user can retrieve information over the network or connect without authentication.

Rule Frequency	Rule Requirements	Rule Taxonomy
1 time	Define the following prior to implementing this rule: ▪ Network IDS Platforms that the e-Security taxonomy can translate (see NIDS Taxonomy table for more information)	NIDS

RuleLg for this Rule

```
filter (e.rv53 = "Sans_MS_NullSessions") flow
   trigger(1,60)
```

Microsoft - LAN Manager (LM) Authentication - Weak LM Hashing Attacks Verification

This rule supports SANS Microsoft top 10 attacks on weak LM hashing. LM uses a much weaker encryption scheme than current Microsoft authentication protocols (NTLM and NTLMv2) and LM passwords can be broken in a short period of time.

Rule Frequency	Rule Requirements	Rule Taxonomy
1 time	Define the following prior to implementing this rule: ▪ Network IDS Platforms that the e-Security taxonomy can translate (see NIDS Taxonomy table for more information)	NIDS

RuleLg for this Rule

```
filter (e.rv53 = "Sans_MS_LM") flow trigger(1,60)
```

Microsoft - General Windows Authentication Attack Verification

This rule supports SANS Microsoft top 10 attacks on passwords. When weak passwords are discovered they should be changed.

Rule Frequency	Rule Requirements	Rule Taxonomy
1 time	Define the following prior to implementing this rule: ▪ Network IDS Platforms that the e-Security taxonomy can translate (see NIDS Taxonomy table for more information)	NIDS

RuleLg for this Rule

```
filter (e.rv53 = "Sans_MS_WeakPasswords") flow
   trigger(1,60)
```

Microsoft - Internet Explorer (IE) Attacks Verification

This rule supports SANS Microsoft top 10 attacks on IE. Later versions of Microsoft have embedded this application into the operating system user interface. Known attacks with IE could result in the compromise of any Microsoft environment later than Windows 2000.

Rule Frequency	Rule Requirements	Rule Taxonomy
1 time	Define the following prior to implementing this rule: ▪ Network IDS Platforms that the e-Security taxonomy can translate (see NIDS Taxonomy table for more information)	NIDS

RuleLg for this Rule

```
filter (e.rv53 = "Sans_MS_IE") flow trigger(1,60)
```

Microsoft - Remote Registry Access Attack Verification

This rule supports SANS Microsoft top 10 attacks on Microsoft’s Registry. The registry for a Microsoft operating system is the location of all system-defined variables. The ability to modify or replace this can severely affect the operation or security of a Microsoft platform.

Rule Frequency	Rule Requirements	Rule Taxonomy
1 time	Define the following prior to implementing this rule: ▪ Network IDS Platforms that the e-Security taxonomy can translate (see NIDS Taxonomy table for more information)	NIDS

RuleLg for this Rule

```
filter (e.rv53 = "Sans_MS_Registry") flow
   trigger(1,60)
```

Microsoft - Windows Scripting Attack Verification

This rule supports SANS Microsoft top 10 attacks on Windows scripting. A number of Microsoft applications are built using the Visual Basic programming language. The ability to execute commands by supplying a script allows an attacker to gain access and control of a Microsoft system.

Rule Frequency	Rule Requirements	Rule Taxonomy
1 time	Define the following prior to implementing this rule: ▪ Network IDS Platforms that the e-Security taxonomy can translate (see NIDS Taxonomy table for more information)	NIDS

RuleLg for this Rule

```
filter (e.rv53 = "Sans_MS_Scripting") flow
   trigger(1,60)
```

UNIX - Remote Procedure Call (RPC) Attack Verification

This rule supports SANS UNIX top 10 attacks on RPC. Remote Procedure Calls are a method within a UNIX environment to access or execute some application or file on a remote system without authentication. Leaving RPC open allows for any remote user to execute privileged commands on your system without authentication. RPC can allow remote attacks.

Rule Frequency	Rule Requirements	Rule Taxonomy
1 time	Define the following prior to implementing this rule: ▪ Network IDS Platforms that the e-Security taxonomy can translate (see NIDS Taxonomy table for more information)	NIDS

RuleLg for this Rule

```
filter (e.rv53 = "Sans_Unix_RPC") flow trigger(1,60)
```

UNIX - Apache Web Server Attack Verification

This rule supports SANS UNIX top 10 attacks on Apache Web Servers. The Apache Web server is a free application that will support Web servers. Running Apache Web server, may leave you vulnerable to this attack.

Rule Frequency	Rule Requirements	Rule Taxonomy
1 time	Define the following prior to implementing this rule: ▪ Network IDS Platforms that the e-Security taxonomy can translate (see NIDS Taxonomy table for more information)	NIDS

RuleLg for this Rule

```
filter (e.rv53 = "Sans_Unix_Apache") flow
   trigger(1,60)
```

UNIX - Secure Shell Attack Verification

This rule supports SANS UNIX top ten attacks on Secure Shell. With the number of problems with telnet and FTP, Secure Shell was developed to encrypt traffic between two machines. This application allows for the transfer of data or the interaction with a remote system through a secure method. However, a number of bugs have been identified with versions of this application that allow an attacker to take complete control of the targeted system.

Rule Frequency	Rule Requirements	Rule Taxonomy
1 time	Define the following prior to implementing this rule: ▪ Network IDS Platforms that the e-Security taxonomy can translate (see NIDS Taxonomy table for more information)	NIDS

RuleLg for this Rule

```
filter (e.rv53 = "Sans_Unix_SSH") flow trigger(1,60)
```

UNIX – Simple Network Management Protocol (SNMP) Attack Verification

This rule supports SANS UNIX top 10 attacks on SNMP. SNMP was originally designed to manage nodes in a network. Security was never implemented in SNMP V 1.0 and little security was implemented in SNMP V 3.0. Therefore, SNMP is subject to a number of attacks.

Rule Frequency	Rule Requirements	Rule Taxonomy
1 time	Define the following prior to implementing this rule: ▪ Network IDS Platforms that the e-Security taxonomy can translate (see NIDS Taxonomy table for more information)	NIDS

RuleLg for this Rule

```
filter (e.rv53 = "Sans_Unix_SNMP") flow trigger(1,60)
```

UNIX - File Transfer Protocol (FTP) Attack Verification

This rule supports SANS UNIX top 10 attacks on FTP. The File Transfer Protocol is a vital part of communication within the Internet. As such, it is a prime target for attackers to re-direct access to and from the Internet.

Rule Frequency	Rule Requirements	Rule Taxonomy
1 time	Define the following prior to implementing this rule: ▪ Network IDS Platforms that the e-Security taxonomy can translate (see NIDS Taxonomy table for more information)	NIDS

RuleLg for this Rule

```
filter (e.rv53 = "Sans_Unix_FTP") flow trigger(1,60)
```

UNIX - Remote Services Attack Verification

This rule supports the SANS UNIX top 10 attacks on Remote Services. Remote Services is a method within a UNIX environment to access or execute some application or file on a remote system without authentication. Leaving Remote Services open allows for any remote user to execute privileged commands on a system without authentication. This allows possible remote attacks.

Rule Frequency	Rule Requirements	Rule Taxonomy
1 time	Define the following prior to implementing this rule: ▪ Network IDS Platforms that the e-Security taxonomy can translate (see NIDS Taxonomy table for more information)	NIDS

RuleLg for this Rule

```
filter (e.rv53 = "Sans_Unix_RemoteServices") flow
  trigger(1,60)
```

UNIX - Line Printer Daemon Attack Verification

This rule supports SANS UNIX top 10 attacks on the Line Printer Daemon. The Line Printer Daemon is the mechanism that UNIX uses to print files. This application runs in a UNIX environment under the root account. Many bugs found in this application allow an attacker to gain complete control of the UNIX environment.

Rule Frequency	Rule Requirements	Rule Taxonomy
1 time	Define the following prior to implementing this rule: ▪ Network IDS Platforms that the e-Security taxonomy can translate (see NIDS Taxonomy table for more information)	NIDS

RuleLg for this Rule

```
filter (e.rv53 = "Sans_Unix_LPD") flow trigger(1,60)
```

UNIX - Sendmail Attack Verification

This rule supports SANS UNIX top 10 attacks on Sendmail. The Sendmail application uses the Simple Mail Transport Protocol (SMTP). This application is a vital part of communication within the Internet. As such, it is a prime target of attackers to re-direct access to and from the Internet.

Rule Frequency	Rule Requirements	Rule Taxonomy
1 time	Define the following prior to implementing this rule: ▪ Network IDS Platforms that the e-Security taxonomy can translate (see NIDS Taxonomy table for more information)	NIDS

RuleLg for this Rule

```
filter (e.rv53 = "Sans_Unix_SendMail") flow
  trigger(1,60)
```

UNIX - BIND/DNS Attack Verification

This rule supports SANS UNIX top 10 attacks on DNS Attacks. The Domain Name Service (DNS) is a vital part of communication within the Internet. As such, it is a prime target of attackers to re-direct access to and from the Internet.

Rule Frequency	Rule Requirements	Rule Taxonomy
1 time	Define the following prior to implementing this rule: ▪ Network IDS Platforms that the e-Security taxonomy can translate (see NIDS Taxonomy table for more information)	NIDS

RuleLg for this Rule

```
filter (e.rv53 = "Sans_Unix_DNS") flow trigger(1,60)
```

UNIX - General UNIX Authentication Attack Verified

This rule supports SANS UNIX top 10 attacks on weak passwords. When weak passwords are discovered they should be changed.

Rule Frequency	Rule Requirements	Rule Taxonomy
1 time	Define the following prior to implementing this rule: ▪ Network IDS Platforms that the e-Security taxonomy can translate (see NIDS Taxonomy table for more information)	NIDS

RuleLg for this Rule

```
filter (e.rv53 = "Sans_Unix_WeakPasswords") flow
  trigger(1,60)
```

Taxonomy Tables

This section contains two tables. These tables are:

- NIDS Taxonomy
- HIDS and OS Taxonomy

They list the different values for e.rv50 through e.rv53 for the RuleLg examples provided.

NIDS Taxonomy Table

Action – Level1 (e.rv50)	System – Level2 (e.rv51)	Detail – Level3 (e.rv52)	Results – Level4 (e.rv53)
Attack	Chat	Access	
		Buffer_Overflow	
		Backdoor	
		Brute_Force	
		DoS	
	DNS	Access	Sans_Unix_DNS
		Buffer_Overflow	Sans_Unix_DNS
		Backdoor	

Action – Level1 (e.rv50)	**System – Level2 (e.rv51)**	**Detail – Level3 (e.rv52)**	**Results – Level4 (e.rv53)**
		Brute_Force	
		DoS	
	Mail	Access	Sans_Unix_SendMail
		Buffer_Overflow	Sans_Unix_SendMail Sans_MS_IE
		Backdoor	
		Brute_Force	
		DoS	
	Telnet	Access	
		Buffer_Overflow	
		Backdoor	
		Brute_Force	
		DoS	
	File	Access	Sans_Unix_FTP Sans_MS_WeakPasswords Sans_MS_NETBIOS
		Buffer_Overflow	Sans_Unix_FTP
		Backdoor	Sans_Unix_FTP
		Brute_Force	
		DoS	
	Web	Access	Sans_Unix_Apache Sans_MS_NETBIOS Sans_MS_WeakPasswords Sans_MS_IIS Sans_MS_Scripting Sans_MS_SQLServer Sans_MS_IE SANS_MS_MDAC
		Buffer_Overflow	Sans_Unix_Apache Sans_MS_IIS
		Backdoor	
		Brute_Force	Sans_MS_IIS
		DoS	Sans_Unix_Apache Sans_MS_IIS
	PC	Virus	Sans_MS_IE Sans_MS_IIS
		Script	
		Worm	Sans_MS_SQLServer
		Trojan	
	Server	Access	Scan_MS_IIS Sans_MS_Registry Sans_MS_SQLServer Sans_MS_NETBIOS Sans_Unix_remoteServices Sans_Unix_RPC Sans_Unix_SSH
		Buffer_Overflow	Sans_Unix_RemoteServices

Action – Level1 (e.rv50)	**System – Level2 (e.rv51)**	**Detail – Level3 (e.rv52)**	**Results – Level4 (e.rv53)**
			Sans_Unix_WeakPasswords Sans_Unix_RPC Sans_Unix_LPD Sans_MS_SQLServer Sans_MS_MDAC Sans_MS_NETBIOS Sans_Unix_SSH
		Backdoor	Sans_Unix_RPC
		Brute_Force	Sans_MS_SQLServer Sans_MS_WeakPasswords
		DoS	
	Protocol	IP	
		TCP	
		UDP	
		ICMP	
		HTTP	
		Route	
		Talk	
		XFS	
		SSH	
		IGMP	
		Time	
		News	
		Windows	
		RIP	
		IDS	
		SNMP	Sans_Unix_SNMP
		BGP	
	User	Access	Sans_Unix_WeakPasswords Sans_Unix_RemoteServices
		Buffer_Overflow	Sans_Unix_RemoteServices Sans_MS_NETBIOS
		Backdoor	
		Brute_Force	
		DoS	
Probe	Chat		
	DNS		
	Mail		
	File		Sans_Unix_FTP
	Web		Sans_MS_IIS Sans_Unix_Apache
	PC		
	Server		Sans_MS_NullSessions Sans_MS_Registry
	Protocol	IP	
		TCP	

Action – Level1 (e.rv50)	System – Level2 (e.rv51)	Detail – Level3 (e.rv52)	Results – Level4 (e.rv53)
		RIP	
		SNMP	Sans_Unix_SNMP
		SSH	
		Talk	
		Time	
		Windows	
		UDP	
		ICMP	
		DHCP	
	Scan		
	Telnet		Sans_MS_LM
	User		Sans_MS_LM
	IDS		
Policy	Porn		
Compromise	Chat	Access	
		Buffer_Overflow	Sans_Unix_Weak_Passwords
		Backdoor	
		Brute_Force	
		DoS	
	DNS	Access	Sans_Unix_DNS
		Buffer_Overflow	
		Backdoor	
		Brute_Force	
		DoS	
	Mail	Access	
		Buffer_Overflow	
		Backdoor	Sans_Unix_SendMail
		Brute_Force	
		DoS	
	Telnet	Access	
		Buffer_Overflow	
		Backdoor	
		Brute_Force	
		DoS	
	File	Access	
		Buffer_Overflow	
		Backdoor	
		Brute_Force	
		DoS	
	Web	Access	Sans_Unix_Apache
		Buffer_Overflow	Sans_MS_IIS
		Backdoor	Sans_Unix_Apache Sans_MS_Registry
		Brute_Force	
		DoS	
	PC	Virus	

Action – Level1 (e.rv50)	System – Level2 (e.rv51)	Detail – Level3 (e.rv52)	Results – Level4 (e.rv53)
		Script	
		Worm	
		Trojan	
	Server	Access	Sans_MS_SQLServer
		Buffer_Overflow	Sans_Unix_RPC
		Backdoor	Sans_MS_WeakPasswords Sans_MS_Registry Sans_Unix_SNMP Sans_Unix_WeakPasswords
		Brute_Force	
		DoS	
	User	Access	
		Buffer_Overflow	
		Backdoor	
		Brute_Force	
		DoS	

HIDS & OS Taxonomy Table

Action – Level1 (e.rv50)	System – Level2 (e.rv51)	Detail – Level3 (e.rv52)	Results – Level4 (e.rv53)
Attack	File	Delete	App OS
		Execute	App OS
		Create	App OS
		Modify	App OS
		Access	App OS
	Service	Delete	App OS
		Stop	App OS
		Start	App OS
		Create	App OS
		Access	App OS Priv Mail ID Network File System

Action – Level1 (e.rv50)	**System – Level2 (e.rv51)**	**Detail – Level3 (e.rv52)**	**Results – Level4 (e.rv53)**
		Buffer_Overflow	
		Backdoor	
		DoS	
	Config	Delete	App OS
		Modify	App OS
		Create	App OS
		Enable	App OS
		Access	App OS
	User	Create	ID Auth Param Priv
		Modify	ID Auth Param Priv
		Delete	ID Auth Param Priv
		Access	Guest Priv Root Other
	Group	Create	Member Group
		Modify	Member Group
		Delete	Member Group
	System	Information	
		Memory	
		Debug	
	Anomoly		
	Telnet	Access	
		Buffer_Overflow	
		Backdoor	
		Brute_Force	
		DoS	
	Web	Access	
		Buffer_Overflow	
		Backdoor	
		Brute_Force	

Action – Level1 (e.rv50)	**System – Level2 (e.rv51)**	**Detail – Level3 (e.rv52)**	**Results – Level4 (e.rv53)**
		DoS	
	PC	Virus	
		Script	
		Backdoor	
		Worm	
		Trojan	
	DNS	Access	
		Buffer_Overflow	
		Backdoor	
		Brute_Force	
		DoS	
	Mail	Access	
		Buffer_Overflow	
		Backdoor	
		Brute_Force	
		DoS	
Probe	File	Delete	App OS
		Execute	App OS
		Create	App OS
		Modify	App OS
		Access	App OS
	Service	Delete	App OS
		Stop	App OS
		Start	App OS
		Create	App OS
		Access	App OS File ID Mail Priv Network System
	Config	Delete	App OS
		Modify	App OS
		Create	App

Action – Level1 (e.rv50)	**System – Level2 (e.rv51)**	**Detail – Level3 (e.rv52)**	**Results – Level4 (e.rv53)**
			OS
		Enable	App OS
		Access	App OS
	User	Create	ID Auth Param Priv
		Modify	ID Auth Param Priv
		Delete	ID Auth Param Priv
		Access	Guest Root Other
	Group	Create	Member Group
		Modify	Member Group
		Delete	Member Group
	System	Information	
		Memory	
		Debug	
	Anomoly		
	Web	Access	
		Buffer_Overflow	
		Backdoor	
		Brute_Force	
		DoS	
	Mail	Access	
		Buffer_Overflow	
		Backdoor	
		Brute_Force	
		DoS	
	Protocol	IP	
		TCP	
		UDP	
		ICMP	
		HTTP	
		Route	
		Talk	

Action – Level1 (e.rv50)	System – Level2 (e.rv51)	Detail – Level3 (e.rv52)	Results – Level4 (e.rv53)
		XFS	
		SSH	
		IGMP	
		Time	
		News	
		Windows	
		RIP	
		IDS	
		SNMP	
		BGP	

Correlation Output

The output structure for the Correlation Engine allows sorting, filtering and reporting on data generated as part of a Watchlist or Correlation rule.

Correlation Rule Output Structure

The default output values are:

- RES set to "Correlation" unless set by user
- SubRes set to "<rule>.<rulename> unless set by user
- Sev set to 4 unless set by user
- ST (sensor type - C)
- EI (rule pattern - SIP=’1.2.3.4.’ then semicolon, then rule threshold in format 3-2-m (count of 3 in 2 minutes, for example)
- RT2 (rule name)

Script Parameters Passed

The script parameters passed affect both Watchlist Rules and Correlation Rules. Script parameters are specified in the Perform Action input box of the Activation Criteria tab with the format %xyz% where xyz is the name of the parameter. The names of the parameters that represent meta-tags can be either a short name (such as sip) or a long name (such as SourceIP). Parameter names are case-sensitive.

Parameters

The first eleven parameters are special parameters. They are not meta-tags. They correspond to correlated events. Parameters twelve through forty-seven are meta-tag parameters.

1. `%RuleName%` - The name of the rule that triggered (Format is rule.rulename).
2. `%RuleType%` - The type of rule that has triggered. C for correlation. W for watchlist.
3. `%RuleDescription%` - The description that was entered when the rule was created.
4. `%RuleSeverity%` - The severity of the rule that has triggered.
5. `%RuleResource%` - The resource name of the rule that has triggered.

6. `%RuleSubResource%` - The subresource name of the rule that has triggered.
7. `%RuleLg%` - The rule in Correlation Engine rule language (RuleLg).
8. `%RuleCount%` - The count of the rule that has triggered.
9. `%RuleDuration%` - The duration (in seconds) of the rule that has triggered.
10. `%RulePattern%` - A list of all tags in the rule's language and the tag's value taken from the last event that triggered the rule. The format is tsn1='value1='value2'tsn3='value3', where:
    - tns1 is the tag short name 1
    - tns2 is the tag short name 2

    For example:

    ```
    sip='192.168.0.3'dip='2.168.0.2'
    ```
11. `%CorrelatedEventID%` - The event identifier of the correlated event generated by the rule that has triggered.
12. %MessageText% - The message text of the rule that has triggered.
13. %EventName% - The event name of the rule that has triggered.

The remaining tags correspond to the field of the last event that triggered the correlated event.

14. %sev% - Severity: The normalized severity of the event (0-5).
15. %vul% - Vulnerability: The vulnerability of the asset identified in this event.
16. %crt% - Criticality: The criticality of the asset identified in this event.
17. %dt% - DateTime: The normalized date and time of the event, as given by the agent.
18. %sip% - SourceIP: The source IP address from which the event originated.
19. %dip% - DestinationIP: The destination IP address to which the event was targeted.
20. %id% - EventID: Unique identifier (UUID) for this event.
21. %src% - SourceID: Unique identifier (UUID) for the e-Security process that generated this event.
22. %port% - WizardPort: e-Security agent port description.
23. %agent% - WizardAgent: e-Security agent port description.
24. %res% - Resource: The resource name.
25. %sres% - SubResource: The sub-resource name.
26. %evt% - EventName: the descriptive name of the event as reported (or given) by the sensor. Example: "Port Scan".
27. %sn% - SensorName: The name of the "ultimate detector" of the event when received in raw data. Example "FW1" for a firewall.
28. %st% - SensorType: The single character designator for the sensor type (N, H, O, V, C, W). H: host-based, N: network-based, O: Other, V: Anti-virus, C: Correlation and W: Watchlist.

29. %et% - EventTime: The normalized time of the event as reported by the sensor; parsed into the format: Y-M-D-H:M:S~AMPM24~TZ.
30. %prot% - Protocol: The network protocol of the event.
31. %shn% - SourceHostName: The source host name from which the event originated.
32. %sp% - SourcePort: The source port from which the event originated.
33. %dhn% - DestinationHostName: The destination host name to which the event was targeted.
34. %dp% - DestinationPort: The destination port to which the event was targetted.
35. %sun% - SourceUserName: The source user name used to initiate an event. Example "jdoe" during an attempt to "su".
36. %dun% - DestinationUserName: The destination user name on which an action was attempted. Example: Attempts to reset the password of root.
37. %fn% - FileName: The name of the program executed or the file accessed, modified or affected. Example: The name of a virus-infected file or a program detected by an IDS.
38. %ei% - ExtendedInformation: Stores additional agent-collected information. Values within this variable are separated by semi-colons (;). Example: A domain for an ID or file names.
39. %rn% - ReporterName: The host name or IP address of the device to which an event was logged or from which notification of the event is sent.
40. %pn% - ProductName: Indicates the type, vendor and product code name of the sensor from which the event was generated. Example: Check Point FireWall=CPFW.
41. %msg% - Message: Free-form message text for the event.
42. %rt1% - Reserved by e-Security for expansion. For use with Advisor (String).
43. %rt2% - Reserved by e-Security for expansion (String).
44. %ct1% - Reserved for use by customers for customer-specific data (String).
45. %ct2% - Reserved for use by customers for customer-specific data (String).
46. %rt3% - Reserved by e-Security for expansion (Number).
47. %ct3% - Reserved for use by customers for customer-specific data (Number).
48. Parameters 46 – 145

    %rv1% thru %rv100%

    These are meta-tags of current event representing reserved variables.
49. Parameters 146 - 245

    %cv1% thru %cv100%

    These are meta-tags of current event representing customer variables.

NOTE: For more information about Commands and Parameters, see Chapter 5 – Wizard and Sentinel Meta Tags in the User Reference Guide and Chapter 9 – Admin Tab, Correlation Rules in the User's Guide.

When using the %all% command:

- If a parameter value is blank or null, then the parameter value is E_NULL or <tag absent>. This way, there will always be 45 parameters regardless if some of the fields are blank.
- When configuring the correlation engine to kick off the HP OVO interface script, you should specify the name of the script along with the %all% parameter tag:

  ```
  esec_ovo %all%
  ```

- When configuring the correlation engine to kick off the BMC interface script, you should specify the name of the script along with the %all% parameter:

  ```
  bmc_interface.csh %all%
  ```

- When configuring the correlation engine to send an email, you should specify the name of the email script along with the %all% parameter and the email address and (optional) subject:

  ```
  email_interface.csh %all% <name>@<domain name> "My
     Subject"
  ```

- All scripts/applications that the correlation engine can execute must be located in the $ESEC_HOME/sentinel/exec (UNIX) %ESEC_HOME%\sentinel\bin (Windows) directory.
- By default, the correlation engine will NOT pass any parameters to the scripts that it executes. You must use the %tags% described above if you want any parameters passed to the scripts.
- When specifying parameters for a script, you may group them by using double-quotes. Here are some examples:

  `%sip% %dip%` - would be treated as two parameters.

  `"%sip% %dip%"` - would be treated as one parameter.

  `"Hello World" %sip%` - would be treated as two parameters.

  `"The message is %msg%"` - would be treated as one parameter.

  `%msg%` - would be treated as one parameter (even if the substituted message has spaces.)

  `"%msg%"` - would also be treated as one parameter (even if the substituted message has spaces.)

Chapter 8 – Sentinel Correlation Command Line Options

Command line options should be used by advanced users. Typical users should not make modifications based on using these options. To access the command line options, go to:

For UNIX:

```
$ESEC_HOME/sentinel/bin
```

For Windows

```
%ESEC_HOME%\sentinel\bin
```

To run the command line option, enter:

```
correlation_engine <correlation command line option>
```

Correlation Command-line Option	Description
-debug	Debug mode (print extensive debug information)
-noErrorLogging	Disable error logging to Windows Event Log.
-ruleFile <file>	Specify text file containing rules to be processed by Correlation Engine instance
-xmlruleFile <file>	Specify xml configurations file to store a local copy of the rules contained on the database. Default: startup_correlation_rules.xml
-inputChannel <string>	Specify communication layer input channel for Correlation Engine. Default: ewizard_binary_event
-outputChannel <string>	Specify communication layer output channel for Correlation Engine. Default: correlation_binary_event.
-outputUpdateChannel <string>	Specify communication layer output update channel for correlation engine. Default: correlation_binary_event_update
-outputExecuteChannel <string>	Specify communication layer output execute channel for Correlation Engine. Default: execute
-outputIncidentChannel <string>	Specify communication layer output incident channel for Correlation Engine. Default: app_incident_req

Correlation Command-line Option	**Description**
-service <string>	Specify communication service (configuration parameter) for Correlation Engine. Default: correlation_engine
-mgmtInputChannel <string>	Specify communication layer management input channel for Correlation Engine. Default: correlation_mgmt_input_channel
-mgmtOutputChannel <string>	Specify communication layer management output channel for Correlation Engine. Default: correlation_mgmt_output_channel
-mgmtService <string>	Specify communication management service (configuration parameter) for Correlation Engine. Default: correlation_engine_mgmt
-configurationFile <file>	Specify file to override Correlation Engine default configuration startup parameters. Default: ± 30 seconds of the Sentinel Server time.
-noStartupRules	Set Correlation Engine to run without retrieving rules stored in the database. The option -ruleFile also bypasses database retrieval.
-dbTimeout <timeout in milliseconds>	Set the timeout value for retrieving the rules stored in the database. Default: 5000 milliseconds
-dbRetries <number>	Set the number of retries to contact the database. Default: 6
-name <engine name>	Sets the reporter name of this correlation engine. Default: Correlation Engine.
-affinityOneProcessor	Set Correlation Engine to run only on one processor.
-useEventTime	This is for test and should not be used.
-useNullOutput	This is for test and should not be used.
-logFile <filename>	This directs the status to a file.
-logPeriod <seconds>	This controls how often the status is written to file.
-version	Display the build version and exit.
-help	Display this help and exit.

Chapter 9 – Sentinel Data Access Service

The Data Access Service (DAS) process is Sentinel Server's persistence service and provides a MOM (message bus) interface to the database. It provides data driven access to the backend database. It receives XML request from the different Sentinel processes, converts them to a query against the database, processes the result from the database and converts it that back to an XML reply. It supports requests to retrieve events for Quick Query and Event Drill Down, to retrieve vulnerability information and advisor information and to manipulate configuration information. DAS also handles logging of all events being received from the Wizard Agent Manager and requests to retrieve and store configuration information.

DAS Container Files

DAS is a container, composed of five different processes. Each process is responsible for different types of database operations. These processes are controlled by the following files:

For Windows and Solaris:

- das_binary.xml: used for event and correlated event insertion operation and aggregation operation
- das_query.xml: all other database operations
- activity_container.xml: used for executing and configuring the activity service
- workflow_container.xml: used for configuring the workflow (iTRAC) service
- das_rt.xml: used for configuring the Active Views function within the Sentinel Control Console

For Linux:

- das_binary.xml: used for event and correlated event insertion operation
- das_query.xml: all other database operations
- das_aggregation.xml: used for aggregation operation
- das_itrac.xml: used for executing and configuring the activity service and for configuring the workflow service
- das_rt.xml: used for configuring the Active Views function within the Sentinel Control Console

CAUTION: Do not manually edit the xml files. Use the dbconfig utility to change any values within the xml files.

Each of these processes has a live log file located in %ESEC_HOME%\Sentinel\log or $ESEC_HOME/Sentinel/log. They are:

For Windows and Solaris:

- das_query0.*.log - All das_query logs
- das_binary0.*.log - All das_binary logs
- activity_container_0.*.log – activity (iTRAC) logs
- workflow_container_0.*.log – workflow (iTRAC) logs
- das_rt0.*.log – Active View logs

For Linux:

- das_query0.*.log - All das_query logs
- das_binary0.*.log - All das_binary logs
- das_itrac0_*.log – activity and workflow logs
- das_aggregation0.*.log – aggregation logs
- das_rt0.*.log – Active View logs

The xml files specify:

- ConnectionManager
  - username
  - password
  - hostname
  - portnumber
  - database (database name)
  - server (oracle or mssql)
  - maxConnections
  - batchSize
  - loadSize
- DispatchManager Specifies the channels in the message bus for DAS to listen to. It also specifies which java class to use to convert xml requests into java objects and specifies which handler to send the java object for processing of the message. For example: an event query request is converted to a java object via the esecurity.cracker.QuickQueryRequestCracker. The cracker then sends it to the esecurity.event.request handler. The handler sends this to one of the services for fulfillment.
- And other components that provide relevant DAS services.

Use the dbconfig utility for Reconfiguring Database Connection Properties for Windows.

Reconfiguring Database Connection Properties

This procedure has to be run for each of the following container file names (containerFilename):

Windows and Solaris	**Linux**
▪ das_binary.xml	▪ das_binary.xml
▪ das_query.xml	▪ das_query.xml
▪ activity_container.xml	▪ das_rt.xml
▪ workflow_container.xml	▪ das_aggregation.xml
▪ das_rt.xml	▪ das_itrac.xml

Reconfiguring Database Connection Properties for Windows

> **NOTE**: At 10 second intervals, the logging properties file will be checked to see if any changes have occurred since it was last read. If the file has changed, the LogManagerRefreshService will re-read the logging properties file.

1. Login as a user with administrative rights where the database is installed.
2. Go to:

   For Windows:

   ```
   %ESEC_HOME%\sentinel\config
   ```

   For UNIX:

```
$ESEC_HOME/sentinel/config
```

3. Enter the following command:

```
dbconfig -n <containerFilename> [-u username] [-p
   password] [-h hostname] [-t port number] [-d
   database] [-s server(mssql or oracle)] [-help]
   [-version]
```

DAS Configuration Files

The following files are used to configure logging of the DAS process.

Windows and Solaris	**Linux**
▪ das_query_log.prop	▪ das_query_log.prop
▪ das_binary_log.prop	▪ das_binary_log.prop
▪ das_rt_log.prop	▪ das_rt_log.prop
▪ activity_container_log.properties	▪ das_itrac_log.prop
▪ workflow_server_log.properties	▪ das_aggregation_log.prop

They are located:

For Windows:

```
%ESEC_HOME%\sentinel\config
```

For UNIX:

```
$ESEC_HOME/sentinel/config
```

These files contain the configuration information for the console handler, which prints messages to a standard output and the file handler, which prints messages to a file. The configuration of each handler allows one to specify the available options for each. These files allow you to specify the configuration of which logging messages should get printed. The levels are:

- OFF – disables all logging
- SEVERE (highest value) - indication that a component has malfunctioned or there is a loss/corruption of critical data
- WARNING - if an action may cause a component to malfunction in the future or if there is non-critical data loss/corruption
- INFO – audit information
- CONFIG
- FINE – for debugging
- FINER – for debugging
- FINEST (lowest value) – for debugging
- ALL – will log all levels

When one specifies a logging level, all log messages of that level and higher (in the above list) will actually be logged. For example, if one specifies the INFO level, then all INFO, WARNING and SEVERE message will be logged.

If you make a change to the files, you'll need to restart DAS in order for the changes to take effect.

Logging is written to:

For Windows:

```
%ESEC_HOME%\sentinel\log\das_query_0.*.log
```

```
%ESEC_HOME%\sentinel\log\das_binary_0.*.log
%ESEC_HOME%\sentinel\log\activity_container_0.0.log
%ESEC_HOME%\sentinel\log\workflow_server_0.0.log
```

For Solaris:

```
$ESEC_HOME/sentinel/log/das_query0.*.log
$ESEC_HOME/sentinel/log/das_binary0.*.log
$ESEC_HOME/sentinel/log/activity_container_0.0.log
$ESEC_HOME/sentinel/log/workflow_server_0.0.log
```

For Linux:

```
$ESEC_HOME/sentinel/log/das_query0.*.log
$ESEC_HOME/sentinel/log/das_binary0.*.log
$ESEC_HOME/sentinel/log/das_itrac_0.*.log
$ESEC_HOME/sentinel/log/das_aggregation0.*.log
```

The * indicates the unique number to resolve conflicts and the generation number to distinguish rotated logs. For example, das_query0.0.log is the log with index 0 (first) file in a rotated set of log files for the DAS process.

Native DB Connectors for Event insertion

The native DB connectors offer increased event insertion performance. The connector you should use depends on the database platform you are using.

MS SQL Native DB Connector

Use the ADO.Net native event store.

How to configure MS SQL Native Connector

1. On the machine where DAS is installed, Install .Net framework.
2. In the das_binary.xml file, change the "insert.strategy" property of the EventStoreService > Persistor to:

```
esecurity.ccs.comp.event.jdbc.ADOLoadStrategy
```

Oracle Native DB Connector

Use the OCI native event store. At a minimum, the Oracle client must be installed on the DAS machine.

How to configure Oracle Native Connector

1. Create a “.profile” file in esecadm’s home dir. Put the following text in that file (modify ORACLE_HOME for your installation):

```
ORACLE_HOME=/build/home/oracle/OraHome

export ORACLE_HOME

LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$ORACLE_HOME/lib

export LD_LIBRARY_PATH
```

2. In das_binary.xml, change the "insert.strategy" property of the EventStoreService > Persistor to:

`esecurity.ccs.comp.event.jdbc.OCILoadStrategy`

Chapter 10 – Changing Default User Passwords

This chapter discusses how to change the passwords for Sentinel default users:

Oracle and MS SQL Authentication:	Windows Authentication:
▪ esecadm	▪ e-Security Administrator
▪ esecapp	▪ e-Security Application DB User
▪ esecdba	▪ e-Security DB Administrator
▪ esecrpt	▪ e-Security Report User

Changing Default User Passwords for Oracle and MS SQL Authentication

NOTE: To change passwords, you must have administrative rights.

Changing esecadm Password

Changing esecadm password

1. Login to the Sentinel Control Console, click on the Admin tab.
2. Open the User Manager window.
3. Double-click on esecadm user account or right-click > User Details.
4. Modify the account password.
5. Click Ok.

Changing esecapp Password

Changing esecapp password

1. For MS SQL, use MS SQL Enterprise Manager and change the esecapp password.
2. For Oracle, use Oracle Enterprise Manager and change the esecapp password.
3. Using the dbconfig utility, update all container xml files. This is required because these xml files store the (encrypted) esecapp password to allow DAS and Advisor to connect to the database.

   - das_binary.xml
   - das_query.xml
   - activity_container.xml
   - workflow_container.xml
   - das_rt.xml

   The container xml files are located:

   For Windows:

   ```
   %ESEC_HOME%\sentinel\config
   ```

   For Oracle:

   ```
   $ESEC_HOME/sentinel/config
   ```

   For more information on usage of the dbconfig utility, go to Sentinel Reference Guide, Chapter 9 - Sentinel Data Access Service.

```
dbconfig –a <containerDirectory> -p <password>
```

Changing esecdba Password

Changing esecdba password

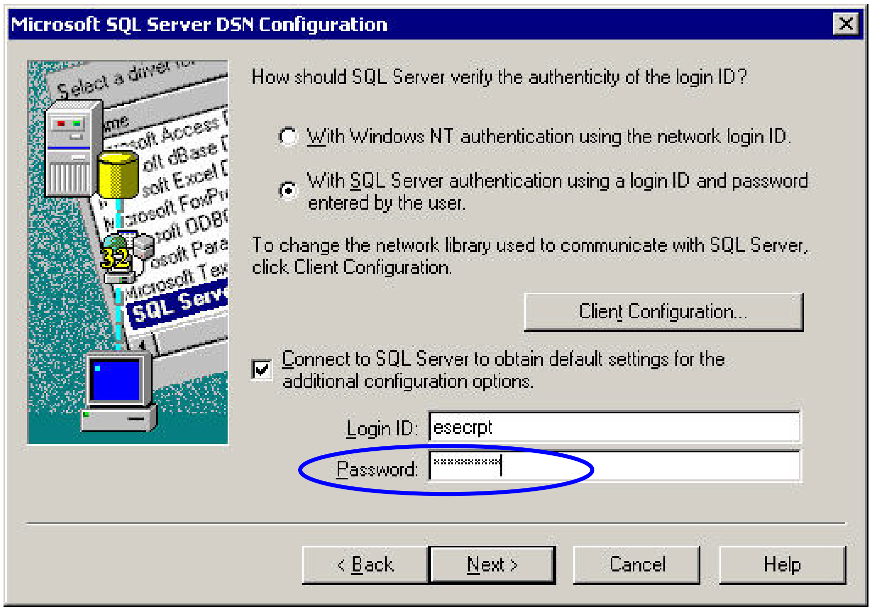
1. For MS SQL, use MS SQL Enterprise Manager and change the esecdba password.
2. For Oracle, use Oracle Enterprise Manager and change the esecdba password.
3. In order for automated SDM tasks to continue to work (e.g. – add partition, archive partition) update the dbPass in the sdm.connect file with the new esecdba password using the SDM GUI or command line. For more information, go to Sentinel User's Guide, Chapter 10 – Sentinel Data Manager.

```
sdm -action saveConnection -server <oracle/mssql> -
   host <hostIp/hostName> -port <portnum> -database
   <databaseName/SID> [-driverProps
   <propertiesFile>] {-user <dbUser> -password
   <dbPass>} -connectFile <filenameToSaveConnection>
```

Changing esecrpt Password

Changing esecrpt password

1. For Sentinel MS SQL Database, use MS SQL Enterprise Manager and change the esecrpt password.
2. For Sentinel Oracle Database, use Oracle Enterprise Manager and change the esecrpt password.
3. Crystal Server for Sentinel MS SQL, if applicable, at the Crystal Server machine update the ODBC DSN (Control Panel > Administrative Tools > Data Sources (ODBC)).
   a. Under the System DSN tab, highlight esecuritydb and click Configure.
   b. Click Next. Update the password.
   c. Click Next until you get a Finish button. Click Finish.



Microsoft SQL Server DSN Configuration

How should SQL Server verify the authenticity of the login ID?

○ With Windows NT authentication using the network login ID.

● With SQL Server authentication using a login ID and password entered by the user.

To change the network library used to communicate with SQL Server, click Client Configuration.

☑ Connect to SQL Server to obtain default settings for the additional configuration options.

Login ID: esecrpt

Password: **********

4. Crystal Server for Sentinel Oracle, no changes needed.

Changing Default User Passwords for Windows Authentication

Changing the e-Security Administrator Password

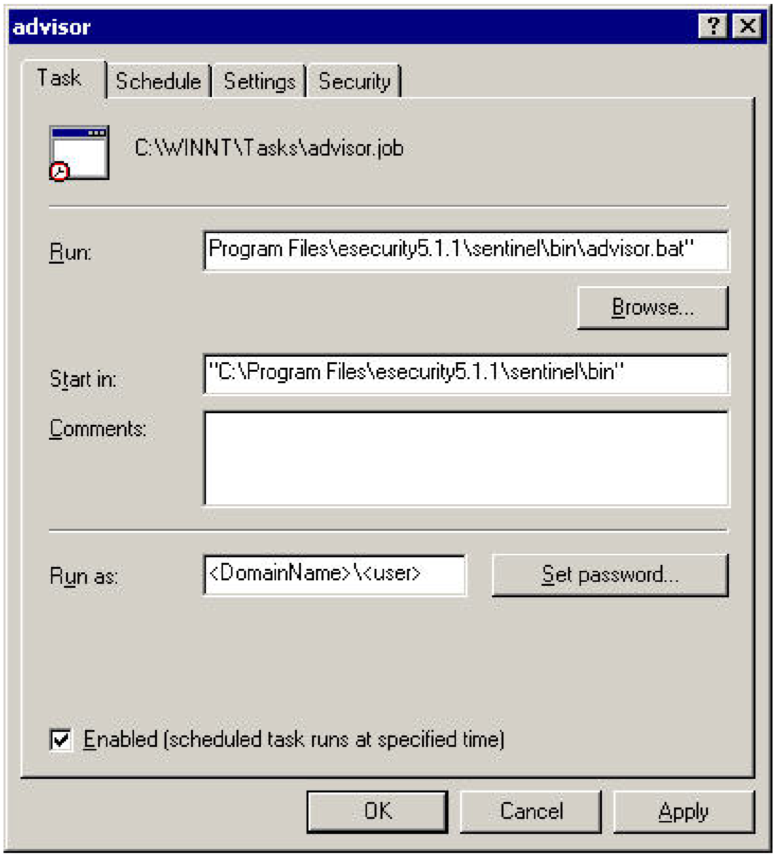
Changing e-Security Administrator password

1. Use the Windows Operating System to change the password.

Changing the e-Security DB Administrator Password

Changing e-Security DB Administrator password

1. Use the Windows Operating System to change the password.
2. If you are running any SDM scheduled tasks (e.g. – for adding or archiving partitions), you will need to update the “Run as” property (Control Panel > Scheduled Tasks > right-click Properties).



C:\WINNT\Tasks\advisor.job

Run: Program Files\esecurity5.1.1\sentinel\bin\advisor.bat"

Start in: "C:\Program Files\esecurity5.1.1\sentinel\bin"

Comments:

Run as: <DomainName>\<user>

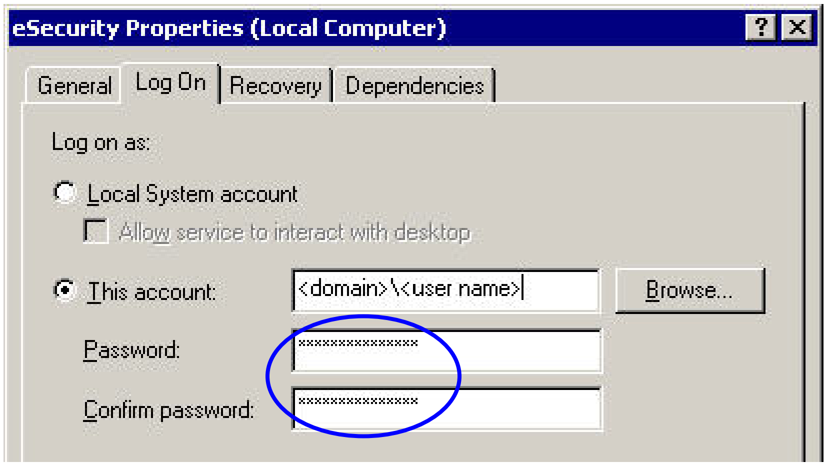
☑ Enabled (scheduled task runs at specified time)

3. Click Set password. Enter the new password twice and click OK. Click Apply and click OK.

Changing the e-Security Application DB Administrator Password

Changing e-Security Application DB Administrator password

1. Use the Windows Operating System to change the password.
2. On your DAS machine, open Windows Services (Control Panel > Administrative Tools > Services).
3. Right-click on eSecurity > Properties. Click the “Log On” tab and update “log on as” password. Click Apply and click OK.



Log on as:

○ Local System account

☐ Allow service to interact with desktop

● This account: <domain>\<user name>

Password: ***************

Confirm password: ***************

4. If you have Advisor installed, you will need to update the “Run as” property (Control Panel > Scheduled Tasks > right-click Properties) of the Advisor Scheduled task(s).
5. Click Set password. Enter the new password twice and click OK. Click Apply and click OK.

Changing the e-Security Report User Password

Changing e-Security Report User password

1. Use the Windows Operating System to change the password.

Chapter 11 - e-Security Database Views for Oracle

This chapter lists the e-Security Schema Views for Oracle. The views provide information for developing your own reports (Crystal Reports).

Views

ADV_ALERT_CVE_RPT_V

View references ADV_ALERT_CVE table that stores the Advisor alert identification number.

Column Name	Datatype	Comment
ALERT_ID	number	Annotation identifier - sequence number.
CVE	varchar2	
DATE_CREATED	date	Insert date
DATE_MODIFIED	date	Last update date
CREATED_BY	number	By user ID
MODIFIED_BY	number	By user ID

ADV_ALERT_PRODUCT_RPT_V

View references ADV_ALERT_PRODUCT table that stores Advisor product information, such as service pack ID number, version and date created.

Column Name	Datatype	Comment
ALERT_ID	number	Annotation identifier - sequence number.
SERVICE_PACK_ID	number	
VENDOR	varchar2	
PRODUCT	varchar2	
VERSION	varchar2	Contains the version number
SERVICE_PACK	varchar2	
PRIMARY_FLAG	number	
DATE_CREATED	date	Insert date
DATE_MODIFIED	date	Last update date
CREATED_BY	number	By user ID
MODIFIED_BY	number	By user ID

ADV_ALERT_RPT_V

View references ADV_ALERT table that stores Advisor alert information, such as name, threat type and date published.

Column Name	Datatype	Comment
ALERT_ID	number	Annotation identifier - sequence number.
VERSION	number	Contains the version number
TEMPLATE_ID	number	
TEMPLATE_NAME	varchar2	

Column Name	Datatype	Comment
THREAT_CATEGORY_NAME	varchar2	
THREAT_TYPE_NAME	varchar2	
HEADLINE	clob	
FIRST_PUBLISHED	date	
LAST_PUBLISHED	date	
STATUS	varchar2	
URGENCY_ID	number	
CREDIBILITY_ID	number	
SEVERITY_ID	number	
SUMMARY	clob	
LEGAL_DISCLAIMER	clob	
COPYRIGHT	varchar2	
BEGIN_EFFECTIVE_DATE	date	
END_EFFECTIVE_DATE	date	
DATE_CREATED	date	Insert date
DATE_MODIFIED	date	Last update date
CREATED_BY	number	By user ID
MODIFIED_BY	number	By user ID

ADV_ATTACK_ALERT_RPT_V

View references ADV_ATTACK_ALERT table that stores Advisor attack information, such as name, threat type and date published.

Column Name	Datatype	Comment
ATTACK_ID	number	
ALERT_ID	number	
DATE_CREATED	date	Insert date
DATE_MODIFIED	date	Last update date
CREATED_BY	number	By user ID
MODIFIED_BY	number	By user ID

ADV_ATTACK_CVE_RPT_V

View references ADV_ATTACK_CVE table that stores Advisor CVE information.

Column Name	Datatype	Comment
ATTACK_ID	number	
CVE	varchar2	
DATE_CREATED	date	Insert date
DATE_MODIFIED	date	Last update date
CREATED_BY	number	By user ID
MODIFIED_BY	number	By user ID

ADV_ATTACK_MAP_RPT_V

View references ADV_ATTACK_MAP table that stores Advisor map information.

Column Name	Datatype	Comment
ATTACK_KEY	number	
ATTACK_ID	number	

Column Name	Datatype	Comment
SERVICE_PACK_ID	number	
ATTACK_NAME	varchar2	
ATTACK_CODE	varchar2	
DATE_CREATED	date	Insert date
DATE_MODIFIED	date	Last update date
CREATED_BY	number	By user ID
MODIFIED_by	number	By user ID

ADV_ATTACK_PLUGIN_RPT_V

View references ADV_ATTACK_PLUGIN table that stores Advisor plug-in information.

Column Name	Datatype	Comment
PLUGIN_KEY	number	
ATTACK_ID	number	
SERVICE_PACK_ID	number	
PLUGIN_ID	varchar2	
PLUGIN_NAME	varchar2	
DATE_CREATED	date	Insert date
DATE_MODIFIED	date	Last update date
CREATED_BY	number	By user ID
MODIFIED_BY	number	By user ID

ADV_ATTACK_RPT_V

View references ADV_ATTACK table that stores Advisor attack information.

Column Name	Datatype	Comment
ALERT_ID	number	
TRUSECURE_ATTACK_NAME	number	
FEED_DATE_CREATED	date	
FEED_DATE_UPDATED	date	
ATTACK_CATEGORY	varchar2	
URGENCY_ID	number	
SEVERITY_ID	number	
LOCAL	number	
REMOTE	number	
BEGIN_EFFECTIVE_DATE	date	
END_EFFECTIVE_DATE	date	
DESCRIPTION	clob	
SCENARIO	clob	
IMPACT	clob	
SAFEGUARDS	clob	
PATCHES	clob	
FALSE_POSITIVES	clob	
DATE_CREATED	date	Insert date
DATE_MODIFIED	date	Last update date

Column Name	Datatype	Comment
CREATED_BY	number	By user ID
MODIFIED_BY	number	By user ID

ADV_CREDIBILITY_RPT_V

View references ADV_CREDIBILITY table that stores Advisor credibility information.

Column Name	Datatype	Comment
CREDIBILITY_ID	number	
CREDIBILITY_RATING	varchar2	
CREDIBILITY_EXPLANATION	varchar2	
DATE_CREATED	date	Insert date
DATE_MODIFIED	date	Last update date
CREATED_BY	number	By user ID
MODIFIED_BY	number	By user ID

ADV_FEED_RPT_V

View references ADV_FEED table that stores Advisor feed information, such as feed name and date.

Column Name	Datatype	Comment
FEED_NAME	varchar2	
FEED_FILE	varchar2	
BEGIN_DATE	date	
END_DATE	date	
FEED_INSERT	number	
FEED_UPDATE	number	
FEED_EXPIRE	number	

ADV_PRODUCT_RPT_V

View references ADV_PRODUCT table that stores Advisor product information such as vendor and product ID.

Column Name	Datatype	Comment
PRODUCT_ID	number	
VENDOR_ID	number	
PRODUCT_CATEGORY_ID	number	
PRODUCT_CATEGORY_NAME	varchar2	
PRODUCT_TYPE-ID	number	
PRODUCT_TYPE_NAME	varchar2	
PRODUCT_NAME	varchar2	
PRODUCT_DESCRIPTION	varchar2	
FEED_DATE_CREATED	date	
FEED_DATE_UPDATED	date	
ACTIVE_FLAG	number	
DATE_CREATED	date	Insert date
DATE_MODIFIED	date	Last update date
CREATED_BY	number	By user ID

Column Name	Datatype	Comment
MODIFIED_BY	number	By user ID

ADV_PRODUCT_SERVICE_PACK_RPT_V

View references ADV_PRODUCT_SERVICE _PACK table that stores Advisor service pack information, such as service pack name, version ID and date.

Column Name	Datatype	Comment
SERVICE_PACK_ID	number	
VERSION_ID	number	Contains the version ID number
SERVICE_PACK_NAME	varchar2	
FEED_DATE_CREATED	date	
FEED_DATE_UPDATED	date	
ACTIVE_FLAG	number	
BEGIN_EFFECTIVE_DATE	date	
END_EFFECTIVE_DATE	date	
DATE_CREATED	date	Insert date
DATE_MODIFIED	date	Last update date
CREATED_BY	number	By user ID
MODIFIED_BY	number	By user ID

ADV_PRODUCT_VERSION_RPT_V

View references ADV_PRODUCT_VERSION table that stores Advisor product version information, such as version name, product and version ID.

Column Name	Datatype	Comment
VERSION_ID	number	Contains the version ID number
PRODUCT_ID	number	
VERSION_NAME	varchar2	
FEED_DATE_CREATED	date	
FEED_DATE_UPDATED	date	
ACTIVE_FLAG	number	
DATE_CREATED	date	Insert date
DATE_MODIFIED	number	Last update date
CREATED_BY	number	By user ID
MODIFIED_BY	number	By user ID

ADV_SEVERITY_RPT_V

View references ADV_SEVERITY table that stores Advisor severity rating information.

Column Name	Datatype	Comment
SEVERITY_ID	number	
SEVERITY_RATING	varchar2	
SEVERITY_EXPLANATION	varchar2	
DATE_CREATED	date	Insert date
DATE_MODIFIED	date	Last update date
CREATED_BY	number	By user ID
MODIFIED_BY	number	By user ID

ADV_SUBALERT_RPT_V

View references ADV_SUBALERT table.

Column Name	Datatype	Comment
ALERT_ID	number	
SUBALERT_ID	number	
CHANGED_SECTIONS	varchar2	
VARIANTS	clob	
VIRUS_NAME	clob	
DESCRIPTION	clob	
IMPACT	clob	
WARNING_INDICATORS	clob	
TECHNICAL_INFO	clob	
TRUSECURE_COMMENTS	clob	
VENDOR_ANNOUNCEMENTS	clob	
SAFEGUARDS	clob	
PATCHES_SOFTWARE	clob	
ALERT_HISTORY	clob	
BACKGROUND_INFO	clob	
DATE_CREATED	date	Insert date
DATE_MODIFIED	date	Last update date
CREATED_BY	number	By user ID
MODIFIED_BY	number	By user ID

ADV_URGENCY_RPT_V

View references ADV_URGENCY table.

Column Name	Datatype	Comment
URGENCY_ID	number	
URGENCY_RATING	varchar2	
URGENCY_EXPLANATION	varchar2	
DATE_CREATED	date	Insert date
DATE_MODIFIED	date	Last update date
CREATED_BY	number	By user ID
MODIFIED_BY	number	By user ID

ADV_VENDOR_RPT_V

View references ADV_VENDOR table that stores Advisor address information.

Column Name	Datatype	Comment
VENDOR_ID	number	
VENDOR_NAME	varchar2	
CONTACT_PERSON	varchar2	
ADDRESS_LINE_1	varchar2	
ADDRESS_LINE_2	varchar2	
ADDRESS_LINE_3	varchar2	
ADDRESS_LINE_4	varchar2	
CITY	varchar2	
STATE	varchar2	

Column Name	Datatype	Comment
COUNTRY	varchar2	
ZIP_CODE	varchar2	
URL	varchar2	
PHONE	varchar2	
FAX	varchar2	
EMAIL	varchar2	
PAGER	varchar2	
FEED_DATE_CREATED	date	
FEED_DATE_UPDATED	date	
ACTIVE_FLAG	number	
DATE_CREATED	date	Insert date
DATE_MODIFIED	date	Last update date
CREATED_BY	number	By user ID
MODIFIED_BY	number	By user ID

ADV_VULN_PRODUCT_RPT_V

View references ADV_VULN_PRODUCT table that stores Advisor vulnerability attack ID and service pack ID.

Column Name	Datatype	Comment
ATTACK_ID	number	
SERVICE_PACK_ID	number	
DATE_CREATED	date	Insert date
DATE_MODIFIED	date	Last update date
CREATED_BY	number	By user ID
MODIFIED_BY	number	By user ID

ANNOTATIONS_RPT_V

View references ANNOTATIONS table that stores documentation or notes that can be associated with objects in the Sentinel system such as incidents.

Column Name	Datatype	Comment
ANN_ID	NUMBER	Annotation identifier - sequence number.
TEXT	VARCHAR2(4000)	Documentation or notes.
DATE_CREATED	DATE	Insert date
DATE_MODIFIED	DATE	Last update date
MODIFIED_BY	NUMBER	Last updating user ID
CREATED_BY	NUMBER	Inserting user ID
ACTION	Varchar2(255)	Action

ASSET_CTGRY_RPT_V

View references ASSET_CTGRY table that stores information about asset categories (e.g. hardware, software, OS, database, etc...).

Column Name	Datatype	Comment
ASSET_CATAGORY_ID	number	Asset category identifier
ASSET_CATAGORY_NAME	varchar2(100)	Asset category name

Column Name	Datatype	Comment
DATE_CREATED	date	Insert date
DATE_MODIFIED	date	Last update date
CREATED_BY	number	Inserting user ID
MODIFIED_BY	number	Last updating user ID

ASSET_HOSTNAME_RPT_V

View references ASSET_HOSTNAME table that stores information about alternate host names for assets.

Column Name	Datatype	Comment
ASSET_HOSTNAME_ID	Varchar2(36)	Asset alternate hostname identifier
PHYSICAL_ASSET_ID	varchar2(36)	Physical asset identifier
HOST_NAME	Varchar2(255)	Host name
CUSTOMER_ID	number	Customer identifier
DATE_CREATED	date	Last update date
DATE_MODIFIED	date	Last updating user ID
CREATED_BY	number	Inserting user ID
MODIFIED_BY	number	Last updating user ID

ASSET_IP_RPT_V

View references ASSET_IP table that stores information about alternate IP addresses for assets.

Column Name	Datatype	Comment
ASSET_IP_ID	Varchar2(36)	Asset alternate IP identifier
PHYSICAL_ASSET_ID	varchar2(36)	Physical asset identifier
IP_ADDRESS	number	Asset IP address
CUSTOMER_ID	number	Customer identifier
DATE_CREATED	date	Insert date
DATE_MODIFIED	date	Last update date
CREATED_BY	number	Inserting user ID
MODIFIED_BY	number	Last updating user ID

ASSET_LOCATION_RPT_V

View references ASSET_LOC table that stores information about asset locations.

Column Name	Datatype	Comment
LOCATION_ID	number	Location identifier
CUSTOMER_ID	number	Customer identifier
BUILDING_NAME	varchar2(255)	Building name
ADDRESS_LINE_1	varchar2(255)	Address line 1
ADDRESS_LINE_2	varchar2(255)	Address line 2
CITY	varchar2(100)	City
STATE	varchar2(100)	State
COUNTRY	varchar2(100)	Country
ZIP_CODE	varchar2(50)	Zip code
DATE_CREATED	date	Insert date

Column Name	Datatype	Comment
DATE_MODIFIED	date	Last update date
CREATED_BY	number	Inserting user ID
MODIFIED_BY	number	Last updating user ID

ASSET_RPT_V

View references ASSET table that stores information about the physical and soft assets.

Column Name	Datatype	Comment
ASSET_ID	varchar2(36)	Asset identifier
CUSTOMER_ID	number	Customer identifier
ASSET_NAME	varchar2(255)	Asset name
PHYSICAL_ASSET_ID	varchar2(36)	Physical asset identifier
PRDT_ID	number	Product identifier
ASSET_CATEGORY_ID	number	Asset category identifier
ENVIRONMENT_IDENTITY_CD	varchar2(5)	Environment identify code
PHYSICAL_ASSET_IND	number(1)	Physical asset indicator
ASSET_VALUE_CODE	varchar2(5)	Asset value code
CRITICALITY_CODE	varchar2(5)	Asset criticality code
SENSITIVITY_CODE	varchar2(5)	Asset sensitivity code
DATE_CREATED	date	Insert date
DATE_MODIFIED	date	Last update date
CREATED_BY	number	Inserting user ID
MODIFIED_BY	number	Last updating user ID

ASSET_VALUE_RPT_V

View references ASSET_VAL_LKUP table that stores information about the asset value.

Column Name	Datatype	Comment
ASSET_VALUE_CODE	varchar2(5)	Asset value code
ASSET_VALUE_NAME	varchar2(50)	Asset value name
DATE_CREATED	date	Insert date
DATE_MODIFIED	date	Last update date
CREATED_BY	number	Inserting user ID
MODIFIED_BY	number	Last updating user ID

ASSET_X_ENTITY_X_ROLE_RPT_V

View references ASSET_X_ENTITY_X_ROLE table that associates a person or an organization to an asset.

Column Name	Datatype	Comment
PERSON_ID	varchar2(36)	Person identifier
ORGANIZATION_ID	varchar2(36)	Organization identifier
ROLE_CODE	varchar2(5)	Role code
ASSET_ID	varchar2(36)	Asset identifier
ENTITY_TYPE_CODE	varchar2(5)	Entity type code
PERSON_ROLE_SEQ	number	Order of persons under a particular role

Column Name	Datatype	Comment
UENCE		
DATE_CREATED	date	Insert date
DATE_MODIFIED	date	Last update date
CREATED_BY	number	Inserting user ID
MODIFIED_BY	number	Last updating user

ASSOCIATIONS_RPT_V

View references ASSOCIATIONS table that associates users to incidents, incidents to annotations, etc.

Column Name	Datatype	Comment
TABLE1	VARCHAR2(64)	Table name 1. For example, incidents
ID1	VARCHAR2(36)	ID1. For example, incident ID.
TABLE2	VARCHAR2(64)	Table name 2. For example, users.
ID2	VARCHAR2(36)	ID2. For example, user ID.
DATE_CREATED	DATE	Insert date.
DATE_MODIFIED	DATE	Last update date
CREATED_BY	NUMBER	Inserting user ID
MODIFIED_BY	NUMBER	Last updating user ID

ATTACHMENTS_RPT_V

View references ATTACHMENTS table that stores attachment data.

Column Name	Datatype	Comment
ATTACHMENT_ID	number	Attachment identifier
NAME	varchar2(255)	Attachment name
SOURCE_REFERENCE	varchar2(64)	Source reference
TYPE	varchar2(32)	Attachment type
SUB_TYPE	varchar2(32)	Attachment subtype
FILE_EXTENSION	varchar2(32)	File extension
ATTACHMENT_DESCRIPTION	varchar2(255)	Attachment description
DATA	clob	Attachment data
DATE_CREATED	date	Insert date
DATE_MODIFIED	date	Last update date
CREATED_BY	number	Inserting by ID
MODIFIED_BY	number	Last updating user ID

CONFIGS_RPT_V

View references CONFIGS table that stores general configuration information of the application.

Column Name	Datatype	Comment
USR_ID	VARCHAR2(32)	User name.
APPLICATION	VARCHAR2(255)	Application identifier
UNIT	VARCHAR2(64)	Application unit
VALUE	VARCHAR2(255)	Text value if any
DATA	CLOB	XML data

Column Name	Datatype	Comment
DATE_CREATED	DATE	Insert date.
DATE_MODIFIED	DATE	Last update date.
CREATED_BY	NUMBER	Inserting user ID.
MODIFIED_BY	NUMBER	Last updating user ID.

CONTACTS_RPT_V

View references CONTACTS table that stores contact information.

Column Name	Datatype	Comment
CNT_ID	NUMBER	Contact ID - Sequence number
FIRST_NAME	VARCHAR2(20)	Contact first name.
LAST_NAME	VARCHAR2(30)	Contact last name.
TITLE	VARCHAR2(128)	Contact title
DEPARTMENT	VARCHAR2(128)	Department
PHONE	VARCHAR2(64)	Contact phone
EMAIL	VARCHAR2(255)	Contact email
PAGER	VARCHAR2(64)	Contact pager
CELL	VARCHAR2(64)	Contact cell phone
DATE_CREATED	DATE	Insert date
DATE_MODIFIED	DATE	Last update date
CREATED_BY	NUMBER	Inserting user ID
MODIFIED_BY	NUMBER	Last updating user ID

CORRELATED_EVENTS_RPT_V

View references CORRELATED_EVENTS_* tables that store correlated event information.

Column Name	Datatype	Comment
PARENT_EVT_ID	varchar2	Event Universal Unique Identifier (UUID) of parent event
CHILD_EVT_ID	varchar2	Event Universal Unique Identifier (UUID) of child event
PARENT_EVT_TIME	DATE	Parent event time
CHILD_EVT_TIME	DATE	Child event time
DATE_CREATED	DATE	Insert date created by DAS
DATE_MODIFIED	DATE	Last update date
CREATED_BY	NUMBER	Inserting user ID
MODIFIED_BY	NUMBER	Last updating user ID

CORRELATED_EVENTS_RPT_V1

View contains current and historical correlated events (correlated events imported from archives).

Column Name	Datatype	Comment
PARENT_EVT_ID	varchar2	Event Universal Unique Identifier (UUID) of parent event
CHILD_EVT_ID	varchar2	Event Universal Unique Identifier (UUID) of child event

Column Name	Datatype	Comment
PARENT_EVT_TIME	DATE	Parent event time
CHILD_EVT_TIME	DATE	Child event time
DATE_CREATED	DATE	Insert date created by DAS
DATE_MODIFIED	DATE	Last update date
CREATED_BY	NUMBER	Inserting user ID
MODIFIED_BY	NUMBER	Last updating user ID

CRITICALITY_RPT_V

View references CRIT_LKUP table that contains information about asset criticality.

Column Name	Datatype	Comment
CRITICALITY_CODE	varchar2(5)	Asset criticality code
CRITICALITY_NAME	varchar2(50)	Asset criticality name
DATE_CREATED	date	Insert date
DATE_MODIFIED	date	Last update date
CREATED_BY	number	By user ID
MODIFIED_BY	number	By user ID

CUST_RPT_V

View references CUST table that stores customer information for MSSPs.

Column Name	Datatype	Comment
CUSTOMER_ID	number	Customer identifier
CUSTOMER_NAME	varchar2(255)	Customer name
DATE_CREATED	date	Insert date
DATE_MODIFIED	date	Last update date
CREATED_BY	number	Inserting user ID
MODIFIED_BY	number	Last updating user ID

ENTITY_TYPE_RPT_V

View references ENTITY_TYP table that stores information about entity types (person, organization).

Column Name	Datatype	Comment
ENTITY_TYPE_CODE	varchar2(5)	Entity type code
ENTITY_TYPE_NAME	varchar2(50)	Entity type name
DATE_CREATED	date	Insert date
DATE_MODIFIED	date	Last update date
CREATED_BY	number	Inserting by user ID
MODIFIED_BY	number	Last updating user ID

ENV_IDENTITY_RPT_V

View references ENV_IDENTITY_LKUP table that stores information about asset environment identity.

Column Name	Datatype	Comment
ENVIRONMENT_IDENTITY_CODE	varchar2(5)	Environment identity code
ENVIRONMENT_IDENTITY_NAME	varchar2(255)	Environment identity name
DATE_CREATED	date	Insert date
DATE_MODIFIED	date	Last update date
CREATED_BY	number	Inserting user by ID
MODIFIED_BY	number	Last updating user ID

ESEC_DISPLAY_RPT_V

View references ESEC_DISPLAY table that stores displayable properties of objects. Currently used in renaming meta-tags. Used with Event Configuration (Business Relevance).

Column Name	Datatype	Comment
DISPLAY_OBJECT	VARCHAR2(32)	The parent object of the property
TAG	VARCHAR2(32)	The native tag name of the property
LABEL	VARCHAR2(32)	The display string of tag.
POSITION	NUMBER	Position of tag within display.
WIDTH	NUMBER	The column width
ALIGNMENT	NUMBER	The horizontal alignment
FORMAT	NUMBER	The enumerated formatter for displaying the property
ENABLED	VARCHAR2(1)	Indicates if the tag is shown.
TYPE	NUMBER	Indicates datatype of tag. 1 = string 2 = ulong 3 = date 4 = uuid 5 = ipv4
DESCRIPTION	VARCHAR2(255)	Textual description of the tag
DATE_CREATED	DATE	Insert date.
DATE_MODIFIED	DATE	Last update date.
CREATED_BY	NUMBER	Inserting user id.
MODIFIED_BY	NUMBER	Last updating user id.
REF_CONFIG	VARCHAR2(4000)	Referential Data Configuration

ESEC_PORT_REFERENCE_RPT_V

View references ESEC_PORT_REFERENCE table that stores industry standard assigned port numbers.

Column Name	**Datatype**	**Comment**
PORT_NUMBER	NUMBER	Per http://www.iana.org/assignments/port-numbers, the numerical representation of the port. This port number is typically associated with the Transport Protocol level in the TCP/IP stack.
PROTOCOL_NUMBER	NUMBER	Per http://www.iana.org/assignments/protocol-numbers, the numerical identifiers used to represent protocols that are encapsulated in an IP packet.
PORT_KEYWORD	VARCHAR2(64)	Per http://www.iana.org/assignments/port-numbers, the keyword representation of the port.
PORT_DESCRIPTION	VARCHAR2(512)	Port description.
DATE_CREATED	DATE	Insert date.
DATE_MODIFIED	DATE	Last updating date.
CREATED_BY	NUMBER	Inserting User ID.
MODIFIED_BY	NUMBER	Last modifying User ID.

ESEC_PROTOCOL_REFERENCE_RPT_V

View references ESEC_PROTOCOL_REFERENCE table that stores industry standard assigned protocol numbers.

Column Name	**Datatype**	**Comment**
PROTOCOL_NUMBER	NUMBER	Per http://www.iana.org/assignments/protocol-numbers, the numerical identifiers used to represent protocols that are encapsulated in an IP packet.
PROTOCOL_KEYWORD	VARCHAR2(64)	Per http://www.iana.org/assignments/protocol-numbers, the keyword used to represent protocols that are encapsulated in an IP packet.

Column Name	Datatype	Comment
PROTOCOL_DESCRIPTION	VARCHAR2(512)	IP packet protocol description.
DATE_CREATED	DATE	Insert date.
DATE_MODIFIED	DATE	Last update date.
CREATED_BY	NUMBER	Inserting User ID.
MODIFIED_BY	NUMBER	Last updating User ID.

ESEC_SEQUENCE_RPT_V

View references ESEC_SEQUENCE table that's used to generate primary key sequence numbers for e-Security tables.

Column Name	Datatype	Comment
TABLE_NAME	VARCHAR2(32)	Name of the table.
COLUMN_NAME	VARCHAR2(32)	Name of the column
SEED	NUMBER	Current value of primary key field.
DATE_CREATED	DATE	Insert date.
DATE_MODIFIED	DATE	Last update date.
CREATED_BY	NUMBER	Inserting user ID.
MODIFIED_BY	NUMBER	Last updating user ID.

EVENTS_ALL_RPT_V (Provided for backward compatibility purpose)

View contains current and historical events (events imported from archives).

Column Name	Datatype	Comment
EVENT_ID	varchar2	Event identifier
RESOURCE_NAME	varchar2(255)	Resource name
SUB_RESOURCE	varchar2(255)	Subresource name
SEVERITY	number	Event severity
EVENT_PARSE_TIME	date	Event time
EVENT_DATE_TIME	date	Event time
BASE_MESSAGE	varchar2(4000)	Base message
EVENT_NAME	varchar2(255)	Name of the event as reported by the sensor
EVENT_TIME	varchar2(255)	Event time as reported by the sensor
SENSOR_NAME	varchar2(255)	Sensor name
SENSOR_TYPE	varchar2(5)	Sensor type: H – host-based N – network-based V – virus O – other
PROTOCOL	varchar2(255)	Protocol Name
SOURCE-IP	number	Source IP address in numeric format
SOURCE_HOST_NAME	varchar2(255)	Source host name
SOURCE_PORT	varchar2(32)	Source port
DESTINATION_IP	number	Destination IP address in numeric format

Column Name	**Datatype**	**Comment**
DESTINATION_HOST_NAME	varchar2(255)	Destination host name
DESTINATION_PORT	varchar2(32)	Destination port
SOURCE_USER_NAME	varchar2(255)	Source user name
DESTINATION_USER_NAME	varchar2(255)	Destination user name
FILE_NAME	varchar2(1000)	File name
EXTENDED_INFO	varchar2(1000)	Extended information
REPORT_NAME	varchar2(255)	Reporter name
PRODUCT_NAME	varchar2(255)	Reporting product name
CUSTOM_TAG_1	varchar2(255)	Customer Tag 1
CUSTOM_TAG_2	varchar2(255)	Customer Tag 2
CUSTOM_TAG_3	number	Customer Tag 3
RESERVED_TAG_1	VARCHAR2(255)	Reserved Tag 1 Reserved for future use by e-Security. This field is used for Advisor information concerning attack descriptions.
RESERVED_TAG_2	varchar2(255)	Reserved for future use by e-Security. Use of this field for any other purpose may result in data being overwritten by future functionality.
RESERVED_TAG_3	number	Reserved for future use by e-Security. Use of this field for any other purpose may result in data being overwritten by future functionality.
SOURCE_UUID	varchar(36)	Source UUID
PORT	varchar(64)	Agent port
AGENT	varchar2(64)	Agent name
VULNERABILITY_RATING	number	Vulnerability rating
CRITICALITY_RATING	number	Criticality rating
DATE_CREATED	date	Insert date
DATE_MODIFIED	date	Last update date
CREATED_BY	number	By user ID
MODIFIED_BY	number	By user ID
RV01 - 10	NUMBER	Reserved Value 1 - 10 Reserved for future use by e-Security. Use of this field for any other purpose may result in data being overwritten by future functionality.
RV11 - 20	DATE	Reserved Value 11 - 20 Reserved for future use by e-Security. Use of this field for any other purpose may result in data being overwritten by future functionality.

Column Name	**Datatype**	**Comment**
RV21 - 25	varchar2	Reserved Value 21 - 25 Reserved for future use by e-Security to store UUIDs. Use of this field for any other purpose may result in data being overwritten by future functionality.
RV26 - 31	VARCHAR2(255)	Reserved Value 26 - 31 Reserved for future use by e-Security. Use of this field for any other purpose may result in data being overwritten by future functionality.
RV32	VARCHAR2(255)	Reserved Value 32 Reserved for DeviceCategory Use of this field for any other purpose may result in data being overwritten by future functionality.
RV33	VARCHAR2(255)	Reserved Value 33 Reserved for EventContex Use of this field for any other purpose may result in data being overwritten by future functionality.
RV34	VARCHAR2(255)	Reserved Value 34 Reserved for SourceThreatLevel Use of this field for any other purpose may result in data being overwritten by future functionality.
RV35	VARCHAR2(255)	Reserved Value 35 Reserved for SourceUserContext. Use of this field for any other purpose may result in data being overwritten by future functionality.
RV36	VARCHAR2(255)	Reserved Value 36 Reserved for DataContext. Use of this field for any other purpose may result in data being overwritten by future functionality.

Column Name	Datatype	Comment
RV37	VARCHAR2(255)	Reserved Value 37 Reserved for SourceFunction. Use of this field for any other purpose may result in data being overwritten by future functionality.
RV38	VARCHAR2(255)	Reserved Value 38 Reserved for SourceOperationalContext. Use of this field for any other purpose may result in data being overwritten by future functionality.
RV39	VARCHAR2(255)	Reserved Value 39 Reserved for MSSPCustomerName. Use of this field for any other purpose may result in data being overwritten by future functionality.
RV40 - 43	VARCHAR2(255)	Reserved Value 40 - 43 Reserved for future use by e-Security. Use of this field for any other purpose may result in data being overwritten by future functionality.
RV44	VARCHAR2(255)	Reserved Value 44 Reserved for DestinationThreatLevel. Use of this field for any other purpose may result in data being overwritten by future functionality.
RV45	VARCHAR2(255)	Reserved Value 45 Reserved for DestinationUserContext. Use of this field for any other purpose may result in data being overwritten by future functionality.
RV46	VARCHAR2(255)	Reserved Value 46 Reserved for VirusStatus. Use of this field for any other purpose may result in data being overwritten by future functionality.

Column Name	Datatype	Comment
RV47	VARCHAR2(255)	Reserved Value 47 Reserved for future use by e-Security. Use of this field for any other purpose may result in data being overwritten by future functionality.
RV48	VARCHAR2(255)	Reserved Value 48 Reserved for DestinationOperationalContext. Use of this field for any other purpose may result in data being overwritten by future functionality.
RV49	VARCHAR2(255)	Reserved Value 49 Reserved for future use by e-Security. Use of this field for any other purpose may result in data being overwritten by future functionality.
RV50	VARCHAR2(255)	Taxonomy level 1
RV51	VARCHAR2(255)	Taxonomy level 2
RV52	VARCHAR2(255)	Taxonomy level 3
RV53	VARCHAR2(255)	Taxonomy level 4
CV01 - 10	NUMBER	Custom Value 1 - 10 Reserved for use by Customer, typically for association of Business relevant data
CV11 - 20	DATE	Custom Value 11 - 20 Reserved for use by Customer, typically for association of Business relevant data
CV21 - 100	VARCHAR2(255)	Custom Value 21 - 100Reserved for use by Customer, typically for association of Business relevant data

EVENTS_ALL_RPT_V1 (Provided for backward compatibility purpose)

View contains current events. It has the same columns as EVENT_ALL_RPT_V.

EVENTS_RPT_V (Provided for backward compatibility purpose)

View contains current and historical events. It has the same columns as EVENT_ALL_RPT_V.

EVENTS_RPT_V1 (Provided for backward compatibility purpose)

View contains current events. It has the same columns as EVENT_ALL_RPT_V.

EVENTS_RPT_V2 (All new Sentinel 5 reports should use this view)

View contains current event and historical events.

Column Name	Datatype	Comment
EVENT_ID	varchar2	Event identifier
RESOURCE_NAME	varchar2(255)	Resource name
SUB_RESOURCE	varchar2(255)	Subresource name
SEVERITY	number	Event severity
EVENT_PARSE_TIME	date	Event time
EVENT_DATETIME	date	Event time
BASE_MESSAGE	varchar2(4000)	Base message
EVENT_NAME	varchar2(255)	Name of the event as reported by the sensor
EVENT_TIME	varchar2(255)	Event time as reported by the sensor
TAXONOMY_ID	number	Taxonomy identifier
PROTOCOL_ID	number	Protocol identifier
AGENT_ID	number	Agent identifier
SOURCE_IP	number	Source IP address in numeric format
SOURCE_HOST_NAME	varchar2(255)	Source host name
SOURCE_PORT	varchar2(32)	Source port
DESTINATION_IP	number	Destination IP address in numeric format
DESTINATION_HOST_NAME	varchar2(255)	Destination host name
DESTINATION_PORT	varchar2(32)	Destination port
SOURCE_USER_NAME	varchar2(255)	Source user name
DESTINATION_USER_NAME	varchar2(255)	Destination user name
FILE_NAME	varchar2(1000)	File name
EXTENDED_INFO	varchar2(1000)	Extened information
CUSTOM_TAG_1	varchar2(255)	Customer Tag 1
CUSTOM_TAG 2	varchar2(255)	Customer Tag 2
CUSTOM_TAG 3	number	Customer Tag 3
RESERVED_TAG_1	VARCHAR2(255)	Reserved Tag 1 Reserved for future use by e-Security, Inc. This field is used for Advisor information concerning attack descriptions.

Column Name	**Datatype**	**Comment**
RESERVED_TAG_2	varchar2(255)	Reserved for future use by e-Security, Inc. Use of this field for any other purpose may result in data being overwritten by future functionality.
RESERVED_TAG_3	number	Reserved for future use by e-Security, Inc. Use of this field for any other purpose may result in data being overwritten by future functionality.
VULNERABILITY_RATING	number	Vulnerability rating
CRITICALITY_RATING	number	Criticality rating
DATE_CREATED	date	Insert date
DATE_MODIFIED	date	Last update date
CREATED_BY	number	Inserting user ID.
MODIFIED_BY	number	Last updating user ID.
RV01 - 10	NUMBER	Reserved Value 1 - 10 Reserved for future use by e-Security, Inc. Use of this field for any other purpose may result in data being overwritten by future functionality.
RV11 - 20	DATE	Reserved Value 1 - 31 Reserved for future use by e-Security, Inc. Use of this field for any other purpose may result in data being overwritten by future functionality.
RV21 - 25	varchar2	Reserved Value 21 - 25 Reserved for future use by e-Security, Inc to store UUIDs. Use of this field for any other purpose may result in data being overwritten by future functionality.
RV26 - 31	VARCHAR2(255)	Reserved Value 26 - 31 Reserved for future use by e-Security, Inc. Use of this field for any other purpose may result in data being overwritten by future functionality.

Column Name	Datatype	Comment
RV33	VARCHAR2(255)	Reserved Value 33 Reserved for EventContex Use of this field for any other purpose may result in data being overwritten by future functionality.
RV34	VARCHAR2(255)	Reserved Value 34 Reserved for SourceThreatLevel Use of this field for any other purpose may result in data being overwritten by future functionality.
RV35	VARCHAR2(255)	Reserved Value 35 Reserved for SourceUserContext. Use of this field for any other purpose may result in data being overwritten by future functionality.
RV36	VARCHAR2(255)	Reserved Value 36 Reserved for DataContext. Use of this field for any other purpose may result in data being overwritten by future functionality.
RV37	VARCHAR2(255)	Reserved Value 37 Reserved for SourceFunction. Use of this field for any other purpose may result in data being overwritten by future functionality.
RV38	VARCHAR2(255)	Reserved Value 38 Reserved for SourceOperationalContext. Use of this field for any other purpose may result in data being overwritten by future functionality.
RV40 - 43	VARCHAR2(255)	Reserved Value 40 - 43 Reserved for future use by e-Security, Inc. Use of this field for any other purpose may result in data being overwritten by future functionality.

Column Name	Datatype	Comment
RV44	VARCHAR2(255)	Reserved Value 44 Reserved for DestinationThreatLevel. Use of this field for any other purpose may result in data being overwritten by future functionality.
RV45	VARCHAR2(255)	Reserved Value 45 Reserved for DestinationUserContext. Use of this field for any other purpose may result in data being overwritten by future functionality.
RV46	VARCHAR2(255)	Reserved Value 46 Reserved for VirusStatus. Use of this field for any other purpose may result in data being overwritten by future functionality.
RV47	VARCHAR2(255)	Reserved Value 47 Reserved for future use by e-Security, Inc. Use of this field for any other purpose may result in data being overwritten by future functionality.
RV48	VARCHAR2(255)	Reserved Value 48 Reserved for DestinationOperationalContext. Use of this field for any other purpose may result in data being overwritten by future functionality.
RV49	VARCHAR2(255)	Reserved Value 49 Reserved for future use by e-Security, Inc. Use of this field for any other purpose may result in data being overwritten by future functionality.
REFERENCE_ID 01 - 20	number	Reserved for future use by e-Security, Inc. Use of this field for any other purpose may result in data being overwritten by future functionality.

Column Name	Datatype	Comment
CV01 - 10	NUMBER	Custom Value 1 - 10 Reserved for use by Customer, typically for association of Business relevant data
CV11 - 20	DATE	Custom Value 11 - 20 Reserved for use by Customer, typically for association of Business relevant data
CV21 - 100	VARCHAR2(255)	Custom Value 21 - 100Reserved for use by Customer, typically for association of Business relevant data

EVT_AGENT_RPT_V

View references EVT_AGENT table that stores information about agents.

Column Name	Datatype	Comment
AGENT_ID	number	Agent identifier
AGENT	varchar2(64)	Agent name
PORT	varchar2(64)	Agent port
REPORT_NAME	varchar2(255)	Reporter name
PRODUCT_NAME	varchar2(255)	Product name
SENSOR_NAME	varchar2(255)	Sensor name
SENSOR_TYPE	varchar2(5)	Sensor type: H - host-based N - network-based V - virus O - other
DEVICE_CTGRY	varchar2(255)	Device category
SOURCE_UUID	varchar2	Source component Universal Unique Identifier (UUID)
DATE_CREATED	date	Insert date
DATE_MODIFIED	date	Last update date
CREATED_BY	number	Inserting user ID
MODIFIED_BY	number	Last updating user ID

EVT_ASSET_RPT_V

View references EVT_ASSET table that stores asset information.

Column Name	Datatype	Comment
EVENT_ASSET_ID	number	Event asset identifier
ASSET_NAME	varchar2(255)	Asset name
PHYSICAL_ASSET_NAME	varchar2(255)	Physical asset name
REFERENCE_ASSET_ID	varchar2(100)	Reference asset identifier, links to source asset management system.

Column Name	Datatype	Comment
MAC_ADDRESS	varchar2(100)	MAC address
RACK_NUMBER	varchar2(50)	Rack number
ROOM_NAME	varchar2(100)	Room name
BUILDING_NAME	varchar2(255)	Building name
CITY	varchar2(100)	City
STATE	varchar2(100)	State
COUNTRY	varchar2(100)	Country
ZIP_CODE	varchar2(50)	Zip code
ASSET_CATEGORY_NAME	varchar2(100)	Asset category name
NETWORK_IDENTITY_NAME	varchar2(255)	Asset network identity name
ENVIRONMENT_IDENTITY_NAME	varchar2(255)	Environment name
ASSET_VALUE_NAME	varchar2(50)	Asset value name
CRITICALITY_NAME	varchar2(50)	Asset criticality name
SENSITIVITY_NAME	varchar2(50)	Asset sensitivity name
CONTACT_NAME_1	varchar2(255)	Name of contact person/organization 1
CONTACT_NAME_2	varchar2(255)	Name of contact person/organization 2
ORGANIZATION_NAME_1	varchar2(100)	Asset owner organization level 1
ORGANIZATION_NAME_2	varchar2(100)	Asset owner organization level 2
ORGANIZATION_NAME_3	varchar2(100)	Asset owner organization level 3
ORGANIZATION_NAME_4	varchar2(100)	Asset owner organization level 4
DATE_CREATED	date	Insert date
DATE_MODIFIED	date	Last update date
CREATED_BY	number	Inserting user ID
MODIFIED_BY	number	Last updating user ID

EVT_DEST_EVT_NAME_SMRY_1_RPT_V

View summarizes event count by destination, taxonomy, event name, severity and event time.

Column Name	Datatype	Comment
DESTINATION_IP	number	Destination IP address
DESTINATION_EVENT_ASSET_ID	number	Event asset identifier
TAXONOMY_ID	number	Taxonomy identifier
EVENT_NAME_ID	number	Event name identifier
SEVERITY	number	Event severity
CUSTOMER_ID	number	Customer identifier
EVT_TIME	date	Event time
EVT_COUNT	number	Event count
DATE_CREATED	date	Insert date

Column Name	Datatype	Comment
DATE_MODIFIED	date	Last update date
CREATED_BY	number	Inserting user ID
MODIFIED_BY	number	Last updating user ID

EVT_DEST_SMRY_1_RPT_V

View contains event destination summary information.

Column Name	Datatype	Comment
DESTINATION_IP	number	Destination IP address
DESTINATION_EVENT_ASSET_ID	number	Event asset identifier
DESTINATION_PORT	varchar2(32)	Destination port
DESTINATION_USR_ID	number	Destination user identifier
TAXONOMY_ID	number	Taxonomy identifier
EVENT_NAME_ID	number	Event name identifier
RESOURCE_ID	number	Resource identifier
AGENT_ID	number	Agent identifier
PROTOCOL_ID	number	Protocol identifier
SEVERITY	number	Event severity
CUSTOMER_ID	number	Customer identifier
EVENT_TIME	date	Event time
EVENT_CNT	number	Event count
DATE_CREATED	date	Insert date
DATE_MODIFIED	date	Last update date
CREATED_BY	number	Inserting user ID
MODIFIED_BY	number	Last updating user ID

EVT_DEST_TXNMY_SMRY_1_RPT_V

View summarizes event count by destination, taxonomy, severity and event time.

Column Name	Datatype	Comment
DESTINATION_IP	number	Destination IP address
DESTINATION_EVENT_ASSET_ID	number	Event asset identifier
TAXONOMY_ID	number	Taxonomy identifier
SEVERITY	number	Event severity
CUSTOMER_ID	number	Customer identifier
EVENT_TIME	date	Event time
EVENT_COUNT	number	Event count
DATE_CREATED	date	Insert date
DATE_MODIFIED	date	Last update date
CREATED_BY	number	Inserting user ID
MODIFIED_BY	number	Last updating user ID

EVT_NAME_RPT_V

View references EVT_NAME table that stores event name information.

Column Name	Datatype	Comment
EVENT_NAME_ID	number	Event name identifier
EVENT_NAME	varchar2(255)	Event name
DATE_CREATED	date	Insert date
DATE_MODIFIED	date	Last update date
CREATED_BY	number	Inserting user ID
MODIFIED_BY	number	Last updating user ID

EVT_PORT_SMRY_1_RPT_V

View summarizes event count by destination port, severity and event time.

Column Name	Datatype	Comment
DESTINATION_PORT	Varchar2(32)	Destination port
SEVERITY	number	Event severity
CUSTOMER_ID	number	Customer identifier
EVENT_TIME	date	Event time
EVENT_COUNT	number	Event count
DATE_CREATED	date	Insert date
DATE_MODIFIED	date	Last update date
CREATED_BY	number	Inserting user ID
MODIFIED_BY	number	Last updating user ID

EVT_PRTCL_RPT_V

View references EVT_PRTCL table that stores event protocol information.

Column Name	Datatype	Comment
PROTOCOL_ID	number	Protocol identifier
PROTOCOL_NAME	varchar2(255)	Protocol name
DATE_CREATED	date	Insert date
DATE_MODIFIED	date	Last update date
CREATED_BY	number	Inserting user ID
MODIFIED_BY	number	Last updating user ID

EVT_RSRC_RPT_V

View references EVT_RSRC table that stores event resource information.

Column Name	Datatype	Comment
RESOURCE_ID	number	Resource identifier
RESOURCE_NAME	varchar2(255)	Resource name
SUBRESOURCE_NAME	varchar2(255)	Subresource name
DATE_CREATED	date	Insert date
DATE_MODIFIED	date	Last update date
CREATED_BY	number	Inserting user ID
MODIFIED_BY	number	Last updating user ID

EVT_SEV_SMRY_1_RPT_V

View summarizes event count by severity and event time.

Column Name	Datatype	Comment
SEVERITY	number	Event severity
CUSTOMER_ID	number	Customer identifier
EVENT_TIME	date	Event time
EVENT_COUNT	number	Event count
DATE_CREATED	date	Insert date
DATE_MODIFIED	date	Last update date
CREATED_BY	number	Inserting user ID
MODIFIED_BY	number	Last updating user ID

EVT_SRC_SMRY_1_RPT_V

View contains event source and destination summary information.

Column Name	Datatype	Comment
SOURCE_IP	number	Source IP address
SOURCE_EVENT_ASSET_ID	number	Source event asset identifier
SOURCE_PORT	varchar2(32)	Source port
SOURCE_USER_ID	number	Source user identifier
TAXONOMY_ID	number	Taxonomy identifier
EVENT_NAME_ID	number	Event name identifier
RESOURCE_ID	number	Resource identifier
AGENT_ID	number	Agent identifier
PROTOCOL_ID	number	Protocol identifier
SEVERITY	number	Event severity
CUSTOMER_ID	number	Customer identifier
EVENT_TIME	date	Event time
EVENT_COUNT	number	Event count
DATE_CREATED	date	Insert date
DATE_MODIFIED	date	Last update date
CREATED_BY	number	Inserting user ID
MODIFIED_BY	number	Last updating user ID

EVT_TXNMY_RPT_V

View references EVT_TXNMY table that stores event taxonomy information.

Column Name	Datatype	Comment
TAXONOMY_ID	number	Taxonomy identifier
TAXONOMY_LEVEL_1	varchar2(100)	Taxonomy level 1
TAXONOMY_LEVEL_2	varchar2(100)	Taxonomy level 2
TAXONOMY_LEVEL_3	varchar2(100)	Taxonomy level 3
TAXONOMY_LEVEL_4	varchar2(100)	Taxonomy level 4
DATE_CREATED	date	Insert date
DATE_MODIFIED	date	Last update date
CREATED_BY	number	Inserting user ID
MODIFIED_BY	number	Last updating user ID

EVT_USR_RPT_V

View references EVT_USR table that stores event user information.

Column Name	Datatype	Comment
USER_ID	number	User identifier
USER_NAME	varchar2(255)	User name
DATE_CREATED	date	Insert date
DATE_MODIFIED	date	Last update date
CREATED_BY	number	Inserting user ID
MODIFIED_BY	number	Last updating user ID

EXTERNAL_DATA_RPT_V

View references EXTERNAL_DATA table that stores external data.

Column Name	Datatype	Comment
EXTERNAL_DATA_ID	number	External data identifier
SOURCE_NAME	varchar2(50)	Source name
SOURCE_DATA_ID	varchar2(255)	Source data identifier
EXTERNAL_DATA	text	External data
EXTERNAL_DATA_TYPE	varchar2(10)	External data type
DATE_CREATED	date	Insert date
DATE_MODIFIED	date	Last update date
CREATED_BY	number	Inserting user ID
MODIFIED_BY	number	Last updating user ID

HIST_ EVENTS_RPT_V

View historical events (events restored from archives).

HIST_ INCIDENTS_RPT_V

View historical events (events restored from archives).

IMAGES_RPT_V

View references IMAGES table that stores system overview image information.

Column Name	Datatype	Comment
NAME	VARCHAR2(128)	Image name
TYPE	VARCHAR2(64)	Image type
DATA	CLOB	Image data
DATE_CREATED	DATE	Insert date
DATE_MODIFIED	DATE	Last update date
CREATED_BY	NUMBER	Inserting user ID
MODIFIED_BY	NUMBER	Last updating user ID

INCIDENTS_ASSETS_RPT_V

View references INCIDENTS_ASSETS table that stores information about the assets that makeup incidents created in the Sentinel Console.

Column Name	Datatype	Comment
INC_ID	NUMBER	Incident identifier – sequence number
ASSET_ID	varchar2	Asset Universal Unique Identifier (UUID)

Column Name	Datatype	Comment
DATE_CREATED	DATE	Insert date
DATE_MODIFIED	DATE	Last update date
CREATED_BY	NUMBER	Inserting user ID
MODIFIED_BY	NUMBER	Last updating user ID

INCIDENTS_EVENTS_RPT_V

View references INCIDENTS_EVENTS table that stores information about the events that makeup incidents created in the Sentinel Console.

Column Name	Datatype	Comment
INC_ID	NUMBER	Incident identifier – sequence number
EVT_ID	varchar2	Event Universal Unique Identifier (UUID)
EVT_TIME	DATE	Event time
DATE_CREATED	DATE	Insert date
DATE_MODIFIED	DATE	Last update date
CREATED_BY	NUMBER	Inserting user ID
MODIFIED_BY	NUMBER	Last updating user ID

INCIDENTS_RPT_V

View references INCIDENTS table that stores information describing the details of incidents created in the Sentinel Console.

Column Name	Datatype	Comment
INC_ID	NUMBER	Incident identifier – sequence number
NAME	VARCHAR2(255)	Incident name
SEVERITY	NUMBER	Incident severity
STT_ID	NUMBER	Incident State ID
SEVERITY_RATING	VARCHAR2(32)	Average of all the event severities that comprise an incident.
VULNERABILITY_RATING	VARCHAR2(32)	Reserved for future use by e-Security. Use of this field for any other purpose may result in data being overwritten by future functionality.
CRITICALITY_RATING	VARCHAR2(32)	Reserved for future use by e-Security. Use of this field for any other purpose may result in data being overwritten by future functionality.
DATE_CREATED	DATE	Insert date
DATE_MODIFIED	DATE	Last update date
CREATED_BY	NUMBER	Inserting user ID
MODIFIED_BY	NUMBER	Last updating user ID
INC_DESC	varchar2(4000)	Incident description

Column Name	Datatype	Comment
INC_PRIORITY	number	Incident priority
INC_CAT	varchar2(255)	Incident category
INC_RES	varchar2(4000)	Incident resolution

INCIDENTS_VULN_RPT_V

View references INCIDENTS_VULN table that stores information about the vulnerabilities that makeup incidents created in the Sentinel Console.

Column Name	Datatype	Comment
INC_ID	NUMBER	Incident identifier – sequence number
VULN_ID	varchar2(36)	Vulnerability Universal Unique Identifier (UUID)
DATE_CREATED	DATE	Insert date
DATE_MODIFIED	DATE	Last update date
CREATED_BY	NUMBER	Inserting user ID
MODIFIED_BY	NUMBER	Last updating user ID

L_STAT_RPT_V

View references L_STAT table that stores statistical information.

Column Name	Datatype	Comment
RES_NAME	VARCHAR2(32)	Resource name
STATS_NAME	VARCHAR2(32)	Statistic name
STATS_VALUE	VARCHAR2(32)	Value of the statistic
OPEN_TOT_SECS	NUMERIC	Number of seconds since 1970.

LOGS_RPT_V

View references LOGS_RPT table that stores logging information.

LOGS Table		
Column Name	**Datatype**	**Comment**
LOG_ID	NUMBER	Sequence number
TIME	DATE	Date of Log
MODULE	VARCHAR2(64)	Module log is for
TEXT	VARCHAR2(4000)	Log text

NETWORK_IDENTITY_RPT_V

View references NETWORK_IDENTITY_LKUP table that stores asset network identity information.

Column Name	Datatype	Comment
NETWORK_IDENTITY_CD	varchar2(5)	Network identity code
NETWORK_IDENTITY_NAME	varchar2(255)	Network identify name
DATE_CREATED	date	Insert date
DATE_MODIFIED	date	Last update date
CREATED_BY	number	Inserting user ID
MODIFIED_BY	number	Last updating user ID

ORGANIZATION_RPT_V

View references ORGANIZATION table that stores organization (asset) information.

Column Name	**Datatype**	**Comment**
ORGANIZATION_ID	varchar2	Organization identifier
ORGANIZATION_NAME	varchar2(100)	Organization name
CUSTOMER_ID	number	Customer identifier
DATE_CREATED	date	Insert date
DATE_MODIFIED	date	Last update date
CREATED_BY	number	Inserting user ID
MODIFIED_BY	number	Last updating user ID

PERSON_RPT_V

View references PERSION table that stores personal (asset) information.

Column Name	**Datatype**	**Comment**
PERSON_ID	varchar2	Person identifier
FIRST_NAME	varchar2(255)	First name
LAST_NAME	varchar2(255)	Last name
CUSTOMER_ID	number	Customer identifier
PHONE_NUMBER	varchar2(50)	Phone number
EMAIL_ADDRESS	varchar2(255)	Email address
DATE_CREATED	date	Insert date
DATE_MODIFIED	date	Last update date
CREATED_BY	number	Inserting user ID
MODIFIED_BY	number	Last updating user ID

PHYSICAL_ASSET_RPT_V

View references PHYSICAL_ASSET table that stores physical asset information.

Column Name	**Datatype**	**Comment**
PHYSICAL_ASSET_ID	varchar2	Physical asset identifier
CUSTOMER_ID	number	Customer identifier
LOCATION_ID	number	Location identifier
HOST_NAME	varchar2(255)	Host name
IP_ADDRESS	number	IP address
NETWORK_IDENTITY_CD	varchar2(5)	Network identity code
MAC_ADDRESS	varchar2(100)	MAC address
RACK_NUMBER	varchar2(50)	Rack number
ROOM_NAME	varchar2(100)	Room name
DATE_CREATED	date	Insert date
DATE_MODIFIED	date	Last update date
CREATED_BY	number	Inserting user ID
MODIFIED_BY	number	Last updating user ID

PRODUCT_RPT_V

View references PRDT table that stores asset product information.

Column Name	Datatype	Comment
PRODUCT_ID	number	Product identifier
PRODUCT_NAME	varchar2(255)	Product name
PRODUCT_VERSION	varchar2(100)	Product version
VENDOR_ID	number	Vendor identifier
DATE_CREATED	date	Insert date
DATE_MODIFIED	date	Last update date
CREATED_BY	number	Inserting user ID
MODIFIED_BY	number	Last updating user ID

ROLE_RPT_V

View references ROLE_LKUP table that stores user role (asset) information.

Column Name	Datatype	Comment
ROLE_CODE	varchar2(5)	Role code
ROLE_NAME	varchar2(255)	Role name
DATE_CREATED	date	Insert date
DATE_MODIFIED	date	Last update date
CREATED_BY	number	Inserting user ID
MODIFIED_BY	number	Last updating user ID

SENSITIVITY_RPT_V

View references SENSITIVITY_LKUP table that stores asset sensitivity information.

Column Name	Datatype	Comment
SENSITIVITY_CODE	varchar2(5)	Asset sensitivity code
SENSITIVITY_NAME	varchar2(50)	Asset sensitivity name
DATE_CREATED	date	Insert date
DATE_MODIFIED	date	Last update date
CREATED_BY	number	By user ID
MODIFIED_BY	number	By user ID

STATES_RPT_V

View references STATES table that stores definitions of states defined by applications or context.

Column Name	Datatype	Comment
STT_ID	NUMBER	State ID – sequence number
CONTEXT	VARCHAR2(64)	Context of the state. That is case, incident, user.
NAME	VARCHAR2(64)	Name of the state.
TERMINAL_FLAG	VARCHAR2(1)	Indicates if state of incident is resolved.
DATE_CREATED	DATE	Insert date
DATE_MODIFIED	DATE	Last update date
MODIFIED_BY	NUMBER	Inserting user ID
CREATED_BY	NUMBER	Last updating user ID

UNASSIGNED_INCIDENTS_RPT_V View

View references CASES and INCIDENTS tables to report on unassigned cases.

Name	**Datatype**
INC_ID	NUMBER
NAME	VARCHAR2(255)
SEVERITY	NUMBER
STT_ID	NUMBER
SEVERITY_RATING	VARCHAR2(32)
VULNERABILITY_RATING	VARCHAR2(32)
CRITICALITY_RATING	VARCHAR2(32)
DATE_CREATED	DATE
DATE_MODIFIED	DATE
CREATED_BY	NUMBER
MODIFIED_BY	NUMBER
INC_DESC	VARCHAR2(4000)
INC_PRIORITY	NUMBER
INC_CAT	VARCHAR2(255)
INC_RES	VARCHAR2(4000)

USERS_RPT_V

View references USERS table that lists all users of the application. The users will also be created as database users to accommodate 3rd party reporting tools.

Column Name	**Datatype**	**Comment**
USR_ID	NUMBER	User identifier – Sequence number
NAME	VARCHAR2(64)	Short, unique user name used as a login
CNT_ID	NUMBER	Contact ID – Sequence number
STT_ID	NUMBER	State ID. Status is either active or inactive.
DESCRIPTION	VARCHAR2(512)	Comments
DATE_CREATED	DATE	Insert date
DATE_MODIFIED	DATE	Last update date
CREATED_BY	NUMBER	Inserting user ID
MODIFIED_BY	NUMBER	Last updating user ID
PERMISSIONS	VARCHAR2(4000)	Permissions currently assigned to the Sentinel user
FILTER	VARCHAR2(128)	Current security filter assigned to the Sentinel user
UPPER_NAME	VARCHAR2(64)	User name in upper case
DOMAIN_AUTH_IND	NUMBER	Domain authentication indication

VENDOR_RPT_V

View references VNDR table that stores information about asset product vendors.

Column Name	**Datatype**	**Comment**
VENDOR_ID	number	Vendor identifier

Column Name	Datatype	Comment
VENDOR_NAME	varchar2(255)	Vendor name
DATE_CREATED	date	Insert date
DATE_MODIFIED	date	Last update date
CREATED_BY	number	Inserting user ID
MODIFIED_BY	number	Last updating user ID

VULN_CALC_SEVERITY_RPT_V

View references VULN_RSRC and VULN to calculate eSecurity vulnerability severity rating base on current vulnerabilities.

Column Name	Datatype
RSRC_ID	VARCHAR22(36)
IP	VARCHAR22(32)
HOST_NAME	VARCHAR22(255)
CRITICALITY	NUMBER
ASSIGNED_VULN_SEVERITY	NUMBER
VULN_COUNT	Count of Vulnerabilities for specified Resource
CALC_SEVERITY	Calculated Severity based on ASSIGNED_VULN_SEVERITY and CRITICALITY

VULN_CODE_RPT_V

View references VULN_CODE table that stores industry assigned vulnerability codes such as Mitre's CVEs and CANs.

Column Name	Datatype
VULN_CODE_ID	VARCHAR2(36)
VULN_ID	VARCHAR2(36)
VULN_CODE_TYPE	VARCHAR2(64)
VULN_CODE_VALUE	VARCHAR2(255)
URL	VARCHAR2(512)
DATE_CREATED	DATE
DATE_MODIFIED	DATE
CREATED_BY	NUMBER
MODIFIED_BY	NUMBER

VULN_INFO_RPT_V

View references VULN_INFO table that stores additional information reported during a scan.

Column Name	Datatype
VULN_INFO_ID	VARCHAR2(36)
VULN_ID	VARCHAR2(36)
VULN_INFO_TYPE	VARCHAR2(36)
VULN_INFO_VALUE	VARCHAR2(2000)
DATE_CREATED	DATE
DATE_MODIFIED	DATE

Column Name	Datatype
CREATED_BY	NUMBER
MODIFIED_BY	NUMBER

VULN_RPT_V

View references VULN table that stores information of scanned system. Each scanner will have its own entry for each system.

Column Name	Datatype
VULN_ID	VARCHAR2(36)
RSRC_ID	VARCHAR2(36)
PORT_NAME	VARCHAR2(64)
PORT_NUMBER	NUMBER
NETWORK_PROTOCOL	NUMBER
APPLICATION_PROTOCOL	VARCHAR2(64)
ASSIGNED_VULN_SEVERITY	NUMBER
COMPUTED_VULN_SEVERITY	NUMBER
VULN_DESCRIPTION	CLOB
VULN_SOLUTION	CLOB
VULN_SUMMARY	VARCHAR2(1000)
BEGIN_EFFECTIVE_DATE	DATE
END_EFFECTIVE_DATE	DATE
DETECTED_OS	VARCHAR2(64)
DETECTED_OS_VERSION	VARCHAR2(64)
SCANNED_APP	VARCHAR2(64)
SCANNED_APP_VERSION	VARCHAR2(64)
VULN_USER_NAME	VARCHAR2(64)
VULN_USER_DOMAIN	VARCHAR2(64)
VULN_TAXONOMY	VARCHAR2(1000)
SCANNER_CLASSIFICATION	VARCHAR2(255)
VULN_NAME	VARCHAR2(300)
VULN_MODULE	VARCHAR2(64)
DATE_CREATED	DATE
DATE_MODIFIED	DATE
CREATED_BY	NUMBER
MODIFIED_BY	NUMBER

VULN_RSRC_RPT_V

View references VULN_RSRC table that stores each resource scanned for a particular scan.

Column Name	Datatype
RSRC_ID	VARCHAR2(36)
SCANNER_ID	VARCHAR2(36)
IP	VARCHAR2(32)
HOST_NAME	VARCHAR2(255)
LOCATION	VARCHAR2(128)
DEPARTMENT	VARCHAR2(128)
BUSINESS_SYSTEM	VARCHAR2(128)

Column Name	Datatype
OPERATIONAL_ENVIRONMENT	VARCHAR2(64)
CRITICALITY	NUMBER
REGULATION	VARCHAR2(128)
REGULATION_RATING	VARCHAR2(64)
DATE_CREATED	DATE
DATE_MODIFIED	DATE
CREATED_BY	NUMBER
MODIFIED_BY	NUMBER

VULN_RSRC_SCAN_RPT_V

View references VULN_RSRC_SCAN table that stores each resource scanned for a particular scan.

Column Name	Datatype
RSRC_ID	VARCHAR2(36)
SCAN_ID	VARCHAR2(36)
DATE_CREATED	DATE
DATE_MODIFIED	DATE
CREATED_BY	NUMBER
MODIFIED_BY	NUMBER

VULN_SCAN_RPT_V

View references table that stores information pertaining to scans.

Column Name	Datatype
SCAN_ID	VARCHAR2(36)
SCANNER_ID	VARCHAR2(36)
SCAN_TYPE	VARCHAR2(10)
SCAN_START_DATE	DATE
SCAN_END_DATE	DATE
CONSOLIDATION_SERVER	VARCHAR2(64)
DATE_CREATED	DATE
DATE_MODIFIED	DATE
CREATED_BY	NUMBER
MODIFIED_BY	NUMBER

VULN_SCAN_VULN_RPT_V

View references VULN_SCAN_VULN table that stores vulnerabilities detected during scans.

Column Name	Datatype
SCAN_ID	VARCHAR2(36)
VULN_ID	VARCHAR2(36)
DATE_CREATED	DATE
DATE_MODIFIED	DATE
CREATED_BY	NUMBER
MODIFIED_BY	NUMBER

VULN_SCANNER_RPT_V

View references VULN_SCANNER table that stores information about vulnerability scanners.

Column Name	**Datatype**
SCANNER_ID	VARCHAR2(36)
PRODUCT_NAME	VARCHAR2(100)
PRODUCT_VERSION	VARCHAR2(64)
SCANNER_TYPE	VARCHAR2(64)
VENDOR	VARCHAR2(100)
SCANNER_INSTANCE	VARCHAR2(64)
DATE_CREATED	DATE
DATE_MODIFIED	DATE
CREATED_BY	NUMBER
MODIFIED_BY	NUMBER

Chapter 12 - e-Security Database Views for Microsoft SQL Server

This chapter lists the e-Security Schema Views for Microsoft SQL Server. The views provide information for developing your own reports (Crystal Reports).

Views

ADV_ALERT_CVE_RPT_V

View references ADV_ALERT_CVE table that stores the Advisor alert identification number.

Column Name	Datatype	Comment
ALERT_ID	int	Annotation identfier - sequence number.
CVE	varchar	
DATE_CREATED	datetime	Insert date
DATE_MODIFIED	datetime	Last update date
CREATED_BY	int	By user ID
MODIFIED_BY	int	By user ID

ADV_ALERT_PRODUCT_RPT_V

View references ADV_ALERT_PRODUCT table that stores Advisor product information, such as service pack ID number, version and date created.

Column Name	Datatype	Comment
ALERT_ID	int	Annotation identifier - sequence number.
SERVICE_PACK_ID	int	
VENDOR	varchar	
PRODUCT	varchar	
VERSION	varchar	Contains the version number
SERVICE_PACK	varchar	
PRIMARY_FLAG	int	
DATE_CREATED	datetime	Insert date
DATE_MODIFIED	datetime	Last update date
CREATED_BY	int	By user ID
MODIFIED_BY	int	By user ID

ADV_ALERT_RPT_V

View references ADV_ALERT table that stores Advisor alert information, such as name, threat type and date published.

Column Name	Datatype	Comment
ALERT_ID	int	Annotation identfier - sequence number.
VERSION	int	Contains the version number

Column Name	Datatype	Comment
TEMPLATE_ID	int	
TEMPLATE_NAME	varchar	
THREAT_CATEGORY_NAME	varchar	
THREAT_TYPE_NAME	varchar	
HEADLINE	text	
FIRST_PUBLISHED	datetime	
LAST_PUBLISHED	datetime	
STATUS	varchar	
URGENCY_ID	int	
CREDIBILITY_ID	int	
SEVERITY_ID	int	
SUMMARY	text	
LEGAL_DISCLAIMER	text	
COPYRIGHT	varchar	
BEGIN_EFFECTIVE_DATE	datetime	
END_EFFECTIVE_DATE	datetime	
DATE_CREATED	datetime	Insert date
DATE_MODIFIED	datetime	Last update date
CREATED_BY	int	By user ID
MODIFIED_BY	int	By user ID

ADV_ATTACK_ALERT_RPT_V

View references ADV_ATTACK_ALERT table that stores Advisor attack information, such as name, threat type and date published.

Column Name	Datatype	Comment
ATTACK_ID	int	
ALERT_ID	int	
DATE_CREATED	datetime	Insert date
DATE_MODIFIED	datetime	Last update date
CREATED_BY	int	By user ID
MODIFIED_BY	int	By user ID

ADV_ATTACK_CVE_RPT_V

View references ADV_ATTACK_CVE table that stores Advisor CVE information.

Column Name	Datatype	Comment
ATTACK_ID	int	
CVE	varchar	
DATE_CREATED	datetime	Insert date
DATE_MODIFIED	datetime	Last update date
CREATED_BY	int	By user ID
MODIFIED_BY	int	By user ID

ADV_ATTACK_MAP_RPT_V

View references ADV_ATTACK_MAP table that stores Advisor map information.

Column Name	Datatype	Comment
ATTACK_KEY	int	
ATTACK_ID	int	
SERVICE_PACK_ID	int	
ATTACK_NAME	varchar	
ATTACK_CODE	varchar	
DATE_CREATED	datetime	Insert date
DATE_MODIFIED	datetime	Last update date
CREATED_BY	int	By user ID
MODIFIED_by	int	By user ID

ADV_ATTACK_PLUGIN_RPT_V

View references ADV_ATTACK_PLUGIN table that stores Advisor plug-in information.

Column Name	Datatype	Comment
PLUGIN_KEY	int	
ATTACK_ID	int	
SERVICE_PACK_ID	int	
PLUGIN_ID	varchar	
PLUGIN_NAME	varchar	
DATE_CREATED	datetime	Insert date
DATE_MODIFIED	datetime	Last update date
CREATED_BY	int	By user ID
MODIFIED_BY	int	By user ID

ADV_ATTACK_RPT_V

View references ADV_ATTACK table that stores Advisor attack information.

Column Name	Datatype	Comment
ALERT_ID	int	
TRUSECURE_ATTACK_NAME	int	
FEED_DATE_CREATED	datetime	
FEED_DATE_UPDATED	datetime	
ATTACK_CATEGORY	varchar	
URGENCY_ID	int	
SEVERITY_ID	int	
LOCAL	int	
REMOTE	int	
BEGIN_EFFECTIVE_DATE	datetime	
END_EFFECTIVE_DATE	datetime	
DESCRIPTION	text	
SCENARIO	text	
IMPACT	text	
SAFEGUARDS	text	
PATCHES	text	
FALSE_POSITIVES	text	

Column Name	Datatype	Comment
DATE_CREATED	datetime	Insert date
DATE_MODIFIED	datetime	Last update date
CREATED_BY	int	By user ID
MODIFIED_BY	int	By user ID

ADV_CREDIBILITY_RPT_V

View references ADV_CREDIBILITY table that stores Advisor credibility information.

Column Name	Datatype	Comment
CREDIBILITY_ID	int	
CREDIBILITY_RATING	varchar	
CREDIBILITY_EXPLANATION	varchar	
DATE_CREATED	datetime	Insert date
DATE_MODIFIED	datetime	Last update date
CREATED_BY	int	By user ID
MODIFIED_BY	int	By user ID

ADV_FEED_RPT_V

View references ADV_FEED table that stores Advisor feed information, such as feed name and date.

Column Name	Datatype	Comment
FEED_NAME	varchar	
FEED_FILE	varchar	
BEGIN_DATE	datetime	
END_DATE	datetime	
FEED_INSERT	int	
FEED_UPDATE	int	
FEED_EXPIRE	int	

ADV_PRODUCT_RPT_V

View references ADV_PRODUCT table that stores Advisor product information such as vendor and product ID.

Column Name	Datatype	Comment
PRODUCT_ID	int	
VENDOR_ID	int	
PRODUCT_CATEGORY_ID	int	
PRODUCT_CATEGORY_NAME	varchar	
PRODUCT_TYPE-ID	int	
PRODUCT_TYPE_NAME	varchar	
PRODUCT_NAME	varchar	
PRODUCT_DESCRIPTION	varchar	
FEED_DATE_CREATED	datetime	
FEED_DATE_UPDATED	datetime	
ACTIVE_FLAG	int	
DATE_CREATED	datetime	Insert date

Column Name	Datatype	Comment
DATE_MODIFIED	datetime	Last update date
CREATED_BY	int	By user ID
MODIFIED_BY	int	By user ID

ADV_PRODUCT_SERVICE_PACK_RPT_V

View references ADV_PRODUCT_SERVICE _PACK table that stores Advisor service pack information, such as service pack name, version ID and date.

Column Name	Datatype	Comment
SERVICE_PACK_ID	int	
VERSION_ID	int	Contains the version ID number
SERVICE_PACK_NAME	varchar	
FEED_DATE_CREATED	datetime	
FEED_DATE_UPDATED	datetime	
ACTIVE_FLAG	int	
BEGIN_EFFECTIVE_DATE	datetime	
END_EFFECTIVE_DATE	datetime	
DATE_CREATED	datetime	Insert date
DATE_MODIFIED	datetime	Last update date
CREATED_BY	int	By user ID
MODIFIED_BY	int	By user ID

ADV_PRODUCT_VERSION_RPT_V

View references ADV_PRODUCT_VERSION table that stores Advisor product version information, such as version name, product and version ID.

Column Name	Datatype	Comment
VERSION_ID	int	Contains the version ID number
PRODUCT_ID	int	
VERSION_NAME	varchar	
FEED_DATE_CREATED	datetime	
FEED_DATE_UPDATED	datetime	
ACTIVE_FLAG	int	
DATE_CREATED	datetime	Insert date
DATE_MODIFIED	int	Last update date
CREATED_BY	int	By user ID
MODIFIED_BY	int	By user ID

ADV_SEVERITY_RPT_V

View references ADV_SEVERITY table that stores Advisor severity rating information.

Column Name	Datatype	Comment
SEVERITY_ID	int	
SEVERITY_RATING	varchar	
SEVERITY_EXPLANATION	varchar	
DATE_CREATED	datetime	Insert date
DATE_MODIFIED	datetime	Last update date

Column Name	Datatype	Comment
CREATED_BY	int	By user ID
MODIFIED_BY	int	By user ID

ADV_SUBALERT_RPT_V

View references ADV_SUBALERT table.

Column Name	Datatype	Comment
ALERT_ID	int	
SUBALERT_ID	int	
CHANGED_SECTIONS	varchar	
VARIANTS	text	
VIRUS_NAME	text	
DESCRIPTION	text	
IMPACT	text	
WARNING_INDICATORS	text	
TECHNICAL_INFO	text	
TRUSECURE_COMMENTS	text	
VENDOR_ANNOUNCEMENTS	text	
SAFEGUARDS	text	
PATCHES_SOFTWARE	text	
ALERT_HISTORY	text	
BACKGROUND_INFO	text	
DATE_CREATED	datetime	Insert date
DATE_MODIFIED	datetime	Last update date
CREATED_BY	int	By user ID
MODIFIED_BY	int	By user ID

ADV_URGENCY_RPT_V

View references ADV_URGENCY table.

Column Name	Datatype	Comment
URGENCY_ID	int	
URGENCY_RATING	varchar	
URGENCY_EXPLANATION	varchar	
DATE_CREATED	datetime	Insert date
DATE_MODIFIED	datetime	Last update date
CREATED_BY	int	By user ID
MODIFIED_BY	int	By user ID

ADV_VENDOR_RPT_V

View references ADV_VENDOR table that stores Advisor address information.

Column Name	Datatype	Comment
VENDOR_ID	int	
VENDOR_NAME	varchar	
CONTACT_PERSON	varchar	
ADDRESS_LINE_1	varchar	
ADDRESS_LINE_2	varchar	

Column Name	Datatype	Comment
ADDRESS_LINE_3	varchar	
ADDRESS_LINE_4	varchar	
CITY	varchar	
STATE	varchar	
COUNTRY	varchar	
ZIP_CODE	varchar	
URL	varchar	
PHONE	varchar	
FAX	varchar	
EMAIL	varchar	
PAGER	varchar	
FEED_DATE_CREATED	datetime	
FEED_DATE_UPDATED	datetime	
ACTIVE_FLAG	int	
DATE_CREATED	datetime	Insert date
DATE_MODIFIED	datetime	Last update date
CREATED_BY	int	By user ID
MODIFIED_BY	int	By user ID

ADV_VULN_PRODUCT_RPT_V

View references ADV_VULN_PRODUCT table that stores Advisor vulnerability attack ID and service pack ID.

Column Name	Datatype	Comment
ATTACK_ID	int	
SERVICE_PACK_ID	int	
DATE_CREATED	datetime	Insert date
DATE_MODIFIED	datetime	Last update date
CREATED_BY	int	By user ID
MODIFIED_BY	int	By user ID

ANNOTATIONS_RPT_V

View references ANNOTATIONS table that stores documentation or notes that can be associated with objects in the Sentinel system such as cases and incidents.

Column Name	Datatype	Comment
ANN_ID	INT	Annotation identfier - sequence number.
TEXT	VARCHAR(4000)	Documentation or notes.
DATE_CREATED	DATETIME	Insert date
DATE_MODIFIED	DATETIME	Last update date
MODIFIED_BY	INT	Last updating user ID
CREATED_BY	INT	Inserting user ID
ACTION	Varchar(255)	Action

ASSET_CTGRY_RPT_V

View references ASSET_CTGRY table that stores information about asset categories (e.g. hardware, software, OS, database, etc...).

Column Name	Datatype	Comment
ASSET_CATEGORY_ID	bigint	Asset category identifier
ASSET_CATEGORY_NAME	varchar(100)	Asset category name
DATE_CREATED	datetime	Insert date
DATE_MODIFIED	datetime	Last update date
CREATED_BY	int	Inserting user ID
MODIFIED_BY	int	Last updating user ID

ASSET_HOSTNAME_RPT_V

View references ASSET_HOSTNAME table that stores information about alternate host names for assets.

Column Name	Datatype	Comment
ASSET_HOSTNAME_ID	Uniqueidentifier	Asset alternate hostname identifier
PHYSICAL_ASSET_ID	uniqueidentifier	Physical asset identifier
HOST_NAME	Varchar(255)	Host name
CUSTOMER_ID	bigint	Customer identifier
DATE_CREATED	datetime	Last update date
DATE_MODIFIED	datetime	Last updating user ID
CREATED_BY	int	Inserting user ID
MODIFIED_BY	int	Last updating user ID

ASSET_IP_RPT_V

View references ASSET_IP table that stores information about alternate IP addresses for assets.

Column Name	Datatype	Comment
ASSET_IP_ID	Uniqueidentifier	Asset alternate IP identifier
PHYSICAL_ASSET_ID	uniqueidentifier	Physical asset identifier
IP_ADDRESS	int	Asset IP address
CUSTOMER_ID	bigint	Customer identifier
DATE_CREATED	datetime	Insert date
DATE_MODIFIED	datetime	Last update date
CREATED_BY	int	Inserting user ID
MODIFIED_BY	int	Last updating user ID

ASSET_LOCATION_RPT_V

View references ASSET_LOC table that stores information about asset locations.

Column Name	Datatype	Comment
LOCATION_ID	bigint	Location identifier
CUSTOMER_ID	bigint	Customer identifier
BUILDING_NAME	varchar(255)	Building name
ADDRESS_LINE_1	varchar(255)	Address line 1

Column Name	Datatype	Comment
ADDRESS_LINE_2	varchar(255)	Address line 2
CITY	varchar(100)	City
STATE	varchar(100)	State
COUNTRY	varchar(100)	Country
ZIP_CODE	varchar(50)	Zip code
DATE_CREATED	datetime	Insert date
DATE_MODIFIED	datetime	Last update date
CREATED_BY	int	Inserting user ID
MODIFIED_BY	int	Last updating user ID

ASSET_RPT_V

View references ASSET table that stores information about the physical and soft assets.

Column Name	Datatype	Comment
ASSET_ID	uniqueidentifier	Asset identifier
CUSTOMER_ID	bigint	Customer identifier
ASSET_NAME	varchar(255)	Asset name
PHYSICAL_ASSET_ID	uniqueidentifier	Physical asset identifier
PRODUCT_ID	bigint	Product identifier
ASSET_CATEGORY_ID	bigint	Asset category identifier
ENVIRONMENT_IDENTITY_CD	varchar(5)	Environment identify code
PHYSICAL_ASSET_IND	bit	Physical asset indicator
ASSET_VALUE_CD	varchar(5)	Asset value code
CRITICALITY_CODE	varchar(5)	Asset criticality code
SENSITIVITY_CODE	varchar(5)	Asset sensitivity code
DATE_CREATED	datetime	Insert date
DATE_MODIFIED	datetime	Last update date
CREATED_BY	int	Inserting user ID
MODIFIED_BY	int	Last updating user ID

ASSET_VALUE_RPT_V

View references ASSET_VAL_LKUP table that stores information about the asset value.

Column Name	Datatype	Comment
ASSET_VALUE_CODE	varchar(5)	Asset value code
ASSET_VALUE_NAME	varchar(50)	Asset value name
DATE_CREATED	datetime	Insert date
DATE_MODIFIED	datetime	Last update date
CREATED_BY	int	Inserting user ID
MODIFIED_BY	int	Last updating user ID

ASSET_X_ENTITY_X_ROLE_RPT_V

View references ASSET_X_ENTITY_X_ROLE table that associates a person or an organization to an asset.

Column Name	Datatype	Comment
PERSON_ID	uniqueidentifier	Person identifier
ORGANIZATION_ID	uniqueidentifier	Organization identifier
ROLE_CODE	varchar(5)	Role code
ASSET_ID	uniqueidentifier	Asset identifier
ENTITY_TYPE_CODE	varchar(5)	Entity type code
PERSON_ROLE_SEQUENCE	int	Order of persons under a particular role
DATE_CREATED	datetime	Insert date
DATE_MODIFIED	datetime	Last update date
CREATED_BY	int	Inserting user ID
MODIFIED_BY	int	Last updating user ID

ASSOCIATIONS_RPT_V

View references ASSOCIATIONS table that associates users to incidents, incidents to annotations, etc.

Column Name	Datatype	Comment
TABLE1	VARCHAR(64)	Table name 1. For example, incidents
ID1	VARCHAR(36)	ID1. For example, incident ID.
TABLE2	VARCHAR(64)	Table name 2. For example, users.
ID2	VARCHAR(36)	ID2. For example, user ID.
DATE_CREATED	DATETIME	Insert date.
DATE_MODIFIED	DATETIME	Last update date
CREATED_BY	INT	Inserting user ID
MODIFIED_BY	INT	Last updating user ID

ATTACHMENTS_RPT_V

View references ATTACHMENTS table that stores attachment data.

Column Name	Datatype	Comment
ATTACHMENT_ID	int	Attachment identifier
NAME	varchar(255)	Attachment name
SOURCE_REFERENCE	varchar(64)	Source reference
TYPE	varchar(32)	Attachment type
SUB_TYPE	varchar(32)	Attachment subtype
FILE_EXTENSION	varchar(32)	File extension
ATTACHMENT_DESCRIPTION	varchar(255)	Attachment description
DATA	clob	Attachment data
DATE_CREATED	datetime	Insert date
DATE_MODIFIED	datetime	Last update date
CREATED_BY	int	Inserting user ID
MODIFIED_BY	int	Last updating user ID

CONFIGS_RPT_V

View references CONFIGS table that stores general configuration information of the application.

Column Name	Datatype	Comment
USR_ID	VARCHAR(32)	User name.
APPLICATION	VARCHAR(255)	Application identifier
UNIT	VARCHAR(64)	Application unit
VALUE	VARCHAR(255)	Text value if any
DATA	TEXT	XML data
DATE_CREATED	DATETIME	Insert date.
DATE_MODIFIED	DATETIME	Last update date.
CREATED_BY	INT	Inserting user ID.
MODIFIED_BY	INT	Last updating user ID.

CONTACTS_RPT_V

View references CONTACTS table that stores contact information.

Column Name	Datatype	Comment
CNT_ID	INT	Contact ID - Sequence number
FIRST_NAME	VARCHAR(20)	Contact first name.
LAST_NAME	VARCHAR(30)	Contact last name.
TITLE	VARCHAR(128)	Contact title
DEPARTMENT	VARCHAR(128)	Department
PHONE	VARCHAR(64)	Contact phone
EMAIL	VARCHAR(255)	Contact email
PAGER	VARCHAR(64)	Contact pager
CELL	VARCHAR(64)	Contact cell phone
DATE_CREATED	DATETIME	Insert date
DATE_MODIFIED	DATETIME	Last update date
CREATED_BY	INT	Inserting user ID
MODIFIED_BY	INT	Last updating user ID

CORRELATED_EVENTS_RPT_V

View references CORRELATED_EVENTS_* tables that store correlated event information.

Column Name	Datatype	Comment
PARENT_EVT_ID	uniqueidentifier	Event Universal Unique Identifier (UUID) of parent event
CHILD_EVT_ID	uniqueidentifier	Event Universal Unique Identifier (UUID) of child event
PARENT_EVT_TIME	DATETIME	Parent event created date
CHILD_EVT_TIME	DATETIME	Child event created date
DATE_CREATED	DATE	Insert date generated by DAS
DATE_MODIFIED	DATETIME	Last update date
CREATED_BY	INT	Inserting user ID
MODIFIED_BY	INT	Last updating user ID

CORRELATED_EVENTS_RPT_V1

View contains current and historical correlated events (correlated events imported from archives).

Column Name	Datatype	Comment
PARENT_EVT_ID	uniqueidentifier	Event Universal Unique Identifier (UUID) of parent event
CHILD_EVT_ID	uniqueidentifier	Event Universal Unique Identifier (UUID) of child event
PARENT_EVT_TIME	DATETIME	Parent event time
CHILD_EVT_TIME	DATETIME	Child event time
DATE_CREATED	DATETIME	Insert date generated by DAS
DATE_MODIFIED	DATETIME	Last update date
CREATED_BY	INT	Inserting user ID
MODIFIED_BY	INT	Last updating user ID

CRITICALITY_RPT_V

View references CRIT_LKUP table that contains information about asset criticality.

Column Name	Datatype	Comment
CRITICALITY_CODE	varchar(5)	Asset criticality code
CRITICALITY_NAME	varchar(50)	Asset criticality name
DATE_CREATED	datetime	Insert date
DATE_MODIFIED	datetime	Last update date
CREATED_BY	int	By user ID
MODIFIED_BY	int	By user ID

CUST_RPT_V

View references CUST table that stores customer information for MSSPs.

Column Name	Datatype	Comment
CUSTOMER_ID	bigint	Customer identifier
CUSTOMER_NAME	varchar(255)	Customer name
DATE_CREATED	datetime	Insert date
DATE_MODIFIED	datetime	Last update date
CREATED_BY	int	Inserting user ID
MODIFIED_BY	int	Last updating user ID

ENTITY_TYPE_RPT_V

View references ENTITY_TYP table that stores information about entity types (person, organization).

Column Name	Datatype	Comment
ENTITY_TYPE_CODE	varchar(5)	Entity type code
ENTITY_TYPE_NAME	varchar(50)	Entity type name
DATE_CREATED	datetime	Insert date
DATE_MODIFIED	datetime	Last update date
CREATED_BY	int	Inserting user ID
MODIFIED_BY	int	Last updating user ID

ENV_IDENTITY_RPT_V

View references ENV_IDENTITY_LKUP table that stores information about asset environment identity.

Column Name	Datatype	Comment
ENVIRONMENT_IDENTITY_CODE	varchar(5)	Environment identity code
ENVIRONMENT_IDENTITY_NAME	varchar(255)	Environment identity name
DATE_CREATED	datetime	Insert date
DATE_MODIFIED	datetime	Last update date
CREATED_BY	int	Inserting user ID
MODIFIED_BY	int	Last updating user ID

ESEC_DISPLAY_RPT_V

View references ESEC_DISPLAY table that stores displayable properties of objects. Currently used in renaming meta-tags. Used with Event Configuration (Business Relevance).

Column Name	Datatype	Comment
DISPLAY_OBJECT	VARCHAR(32)	The parent object of the property
TAG	VARCHAR(32)	The native tag name of the property
LABEL	VARCHAR(32)	The display string of tag.
POSITION	INT	Position of tag within display.
WIDTH	INT	The column width
ALIGNMENT	INT	The horizontal alignment
FORMAT	INT	The enumerated formatter for displaying the property
ENABLED	BIT	Indicates if the tag is shown.
TYPE	INT	Indicates datatype of tag. 1 = string 2 = ulong 3 = date 4 = uuid 5 = ipv4
DESCRIPTION	VARCHAR(255)	Textual description of the tag
DATE_CREATED	DATETIME	Insert date.
DATE_MODIFIED	DATETIME	Last update date.
CREATED_BY	INT	Inserting user id.
MODIFIED_BY	INT	Last updating user id.
REF_CONFIG	VARCHAR(4000)	Referential Data configuration

ESEC_PORT_REFERENCE_RPT_V

View references ESEC_PORT_REFERENCE table that stores industry standard assigned port numbers.

Column Name	Datatype	Comment
PORT_NUMBER	INT	Per http://www.iana.org/assignments/port-numbers, the numerical representation of the port. This port number is typically associated with the Transport Protocol level in the TCP/IP stack.
PROTOCOL_NUMBER	INT	Per http://www.iana.org/assignments/protocol-numbers, the numerical identifiers used to represent protocols that are encapsulated in an IP packet.
PORT_KEYWORD	VARCHAR(64)	Per http://www.iana.org/assignments/port-numbers, the keyword representation of the port.
PORT_DESCRIPTION	VARCHAR(512)	Port description.
DATE_CREATED	DATETIME	Insert date.
DATE_MODIFIED	DATETIME	Last updating date.
CREATED_BY	INT	Inserting User ID.
MODIFIED_BY	INT	Last modifying User ID.

ESEC_PROTOCOL_REFERENCE_RPT_V

View references ESEC_PROTOCOL_REFERENCE table that stores industry standard assigned protocol numbers.

Column Name	Datatype	Comment
PROTOCOL_NUMBER	INT	Per http://www.iana.org/assignments/protocol-numbers, the numerical identifiers used to represent protocols that are encapsulated in an IP packet.
PROTOCOL_KEYWORD	VARCHAR(64)	Per http://www.iana.org/assignments/protocol-numbers, the keyword used to represent protocols that are encapsulated in an IP packet.
PROTOCOL_DESCRIPTION	VARCHAR(512)	IP packet protocol description.
DATE_CREATED	DATETIME	Insert date.
DATE_MODIFIED	DATETIME	Last update date.
CREATED_BY	INT	Inserting User ID.

Column Name	Datatype	Comment
MODIFIED_BY	INT	Last updating User ID.

ESEC_SEQUENCE_RPT_V

View references ESEC_SEQUENCE table that's used to generate primary key sequence numbers for e-Security tables.

Column Name	Datatype	Comment
TABLE_NAME	VARCHAR(32)	Name of the table.
COLUMN_NAME	VARCHAR(32)	Name of the column
SEED	INT	Current value of primary key field.
DATE_CREATED	DATETIME	Insert date.
DATE_MODIFIED	DATETIME	Last update date.
CREATED_BY	INT	Inserting user ID.
MODIFIED_BY	INT	Last updating user ID.

EVENTS_ALL_RPT_V (Provided for backward compatibility purpose)

View contains current and historical events (events imported from archives).

Column Name	Datatype	Comment
EVENT_ID	uniqueidentifier	Event identifier
RESOURCE_NAME	varchar(255)	Resource name
SUB_RESOURCE	varchar(255)	Subresource name
SEVERITY	int	Event severity
EVENT_PARSE_TIME	datetime	Event time
EVENT_DATETIME	datetime	Event time
BASE_MESSAGE	varchar(4000)	Base message
EVENT_NAME	varchar(255)	Name of the event as reported by the sensor
EVENT_TIME	varchar(255)	Event time as reported by the sensor
SENSOR_NAME	varchar(255)	Sensor name
SENSOR_TYPE	varchar(5)	Sensor type: H – host-based N – network-based V – virus O – other
PROTOCOL	varchar(255)	Protocol name
SOURCE_IP	int	Source IP address in numeric format
SOURCE_HOST_NAME	varchar(255)	Source host name
SOURCE_PORT	varchar(32)	Source port
DESTINATION_IP	int	Destination IP address in numeric format
DESTINATION_HOST_NAME	varchar(255)	Destination host name
DESTINATION_PORT	varchar(32)	Destination port
SOURCE_USER_NAME	varchar(255)	Source user name
DESTINATION_USER_NAME	varchar(255)	Destination user name
FILE_NAME	varchar(1000)	File name

Column Name	**Datatype**	**Comment**
EXTENDED_INFO	varchar(1000)	Extened information
REPORT_NAME	varchar(255)	Reporter name
PRODUCT_NAME	varchar(255)	Reporting product name
CUSTOM_TAG_1	varchar(255)	Customer Tag 1
CUSTOM_TAG 2	varchar(255)	Customer Tag 2
CUSTOM_TAG 3	int	Customer Tag 3
RESERVED_TAG_1	VARCHAR(255)	Reserved Tag 1 Reserved for future use by e-Security. This field is used for Advisor information concerning attack descriptions.
RESERVED_TAG_2	varchar(255)	Reserved for future use by e-Security. Use of this field for any other purpose may result in data being overwritten by future functionality.
RESERVED_TAG_3	int	Reserved for future use by e-Security. Use of this field for any other purpose may result in data being overwritten by future functionality.
SOURCE_UUID	uniqueidentifier	Source UUID
PORT	varchar(64)	Agent port
AGENT	varchar(64)	Agent name
VULNERABILITY_RATING	int	Vulnerability rating
CRITICALITY_RATING	int	Criticality rating
DATE_CREATED	datetime	Insert date
DATE_MODIFIED	datetime	Last update date
CREATED_BY	int	Inserting User ID.
MODIFIED_BY	int	Last updating User ID.
RV01 - 10	INT	Reserved Value 1 - 10 Reserved for future use by e-Security. Use of this field for any other purpose may result in data being overwritten by future functionality.
RV11 - 20	DATETIME	Reserved Value 11 - 20 Reserved for future use by e-Security. Use of this field for any other purpose may result in data being overwritten by future functionality.

Column Name	Datatype	Comment
RV21 - 25	uniqueidentifier	Reserved Value 21 - 25 Reserved for future use by e-Security to store UUIDs. Use of this field for any other purpose may result in data being overwritten by future functionality.
RV26 - 31	VARCHAR(255)	Reserved Value 26 - 31 Reserved for future use by e-Security. Use of this field for any other purpose may result in data being overwritten by future functionality.
RV32	VARCHAR(255)	Reserved Value 32 Reserved for DeviceCategory Use of this field for any other purpose may result in data being overwritten by future functionality.
RV33	VARCHAR(255)	Reserved Value 33 Reserved for EventContex Use of this field for any other purpose may result in data being overwritten by future functionality.
RV34	VARCHAR(255)	Reserved Value 34 Reserved for SourceThreatLevel Use of this field for any other purpose may result in data being overwritten by future functionality.
RV35	VARCHAR(255)	Reserved Value 35 Reserved for SourceUserContext. Use of this field for any other purpose may result in data being overwritten by future functionality.
RV36	VARCHAR(255)	Reserved Value 36 Reserved for DataContext. Use of this field for any other purpose may result in data being overwritten by future functionality.

Column Name	Datatype	Comment
RV37	VARCHAR(255)	Reserved Value 37 Reserved for SourceFunction. Use of this field for any other purpose may result in data being overwritten by future functionality.
RV38	VARCHAR(255)	Reserved Value 38 Reserved for SourceOperationalContext. Use of this field for any other purpose may result in data being overwritten by future functionality.
RV39	VARCHAR(255)	Reserved Value 39 Reserved for MSSPCustomerName. Use of this field for any other purpose may result in data being overwritten by future functionality.
RV40 - 43	VARCHAR(255)	Reserved Value 40 - 43 Reserved for future use by e-Security. Use of this field for any other purpose may result in data being overwritten by future functionality.
RV44	VARCHAR(255)	Reserved Value 44 Reserved for DestinationThreatLevel. Use of this field for any other purpose may result in data being overwritten by future functionality.
RV45	VARCHAR(255)	Reserved Value 45 Reserved for DestinationUserContext. Use of this field for any other purpose may result in data being overwritten by future functionality.
RV46	VARCHAR(255)	Reserved Value 46 Reserved for VirusStatus. Use of this field for any other purpose may result in data being overwritten by future functionality.

Column Name	Datatype	Comment
RV47	VARCHAR(255)	Reserved Value 47 Reserved for future use by e-Security. Use of this field for any other purpose may result in data being overwritten by future functionality.
RV48	VARCHAR(255)	Reserved Value 48 Reserved for DestinationOperationalContext. Use of this field for any other purpose may result in data being overwritten by future functionality.
RV49	VARCHAR(255)	Reserved Value 49 Reserved for future use by e-Security. Use of this field for any other purpose may result in data being overwritten by future functionality.
RV50	VARCHAR(255)	Taxonomy level 1
RV51	VARCHAR(255)	Taxonomy level 2
RV52	VARCHAR(255)	Taxonomy level 3
RV53	VARCHAR(255)	Taxonomy level 4
CV01 - 10	INT	Custom Value 1 - 10 Reserved for use by Customer, typically for association of Business relevant data
CV11 - 20	DATETIME	Custom Value 11 - 20 Reserved for use by Customer, typically for association of Business relevant data
CV21 - 100	VARCHAR(255)	Custom Value 21 - 100Reserved for use by Customer, typically for association of Business relevant data

EVENTS_ALL_RPT_V1 (Provided for backward compatibility purpose)

View contains current events. It has the same columns as EVENT_ALL_RPT_V.

EVENTS_RPT_V (Provided for backward compatibility purpose)

View contains current and historical events. It has the same columns as EVENT_ALL_RPT_V.

EVENTS_RPT_V1 (Provided for backward compatibility purpose)

View contains current events. It has the same columns as EVENT_ALL_RPT_V.

EVENTS_RPT_V2 (Provided for backward compatibility purpose)

View contains current event and historical events.

Column Name	**Datatype**	**Comment**
EVENT_ID	uniqueidentifier	Event identifier
RESOURCE_NAME	varchar(255)	Resource name
SUB_RESOURCE	varchar(255)	Subresource name
SEVERITY	int	Event severity
EVENT_PARSE_TIME	datetime	Event time
EVENT_DATETIME	datetime	Event time
BASE_MESSAGE	varchar(4000)	Base message
EVENT_NAME	varchar(255)	Name of the event as reported by the sensor
EVENT_TIME	varchar(255)	Event time as reported by the sensor
TAXONOMY_ID	bigint	Taxonomy identifier
PROTOCOL_ID	bigint	Protocol identifier
AGENT_ID	bigint	Agent identifier
SOURCE_IP	int	Source IP address in numeric format
SOURCE_HOST_NAME	varchar(255)	Source host name
SOURCE_PORT	varchar(32)	Source port
DESTINATION_IP	int	Destination IP address in numeric format
DESTINATION_HOST_NAME	varchar(255)	Destination host name
DESTINATION_PORT	varchar(32)	Destination port
SOURCE_USER_NAME	varchar(255)	Source user name
DESTINATION_USER_NAME	varchar(255)	Destination user name
FILE_NAME	varchar(1000)	File name
EXTENDED_INFO	varchar(1000)	Extened information
CUSTOM_TAG_1	varchar(255)	Customer Tag 1
CUSTOM_TAG 2	varchar(255)	Customer Tag 2
CUSTOM_TAG 3	int	Customer Tag 3
RESERVED_TAG_1	VARCHAR(255)	Reserved Tag 1 Reserved for future use by e-Security. This field is used for Advisor information concerning attack descriptions.
RESERVED_TAG_2	varchar(255)	Reserved for future use by e-Security. Use of this field for any other purpose may result in data being overwritten by future functionality.

Column Name	**Datatype**	**Comment**
RESERVED_TAG_3	int	Reserved for future use by e-Security. Use of this field for any other purpose may result in data being overwritten by future functionality.
VULNERABILITY_RATING	int	Vulnerability rating
CRITICALITY_RATING	int	Criticality rating
DATE_CREATED	datetime	Insert date
DATE_MODIFIED	datetime	Last update date
CREATED_BY	int	Inserting user ID.
MODIFIED_BY	int	Last updating user ID.
RV01 - 10	INT	Reserved Value 1 - 10 Reserved for future use by e-Security. Use of this field for any other purpose may result in data being overwritten by future functionality.
RV11 - 20	DATETIME	Reserved Value 1 - 31 Reserved for future use by e-Security. Use of this field for any other purpose may result in data being overwritten by future functionality.
RV21 - 25	uniqueidentifier	Reserved Value 21 - 25 Reserved for future use by e-Security to store UUIDs. Use of this field for any other purpose may result in data being overwritten by future functionality.
RV26 - 31	VARCHAR(255)	Reserved Value 26 - 31 Reserved for future use by e-Security. Use of this field for any other purpose may result in data being overwritten by future functionality.
RV33	VARCHAR(255)	Reserved Value 33 Reserved for EventContex Use of this field for any other purpose may result in data being overwritten by future functionality.
RV34	VARCHAR(255)	Reserved Value 34 Reserved for SourceThreatLevel Use of this field for any other purpose may result in data being overwritten by future functionality.

Column Name	Datatype	Comment
RV35	VARCHAR(255)	Reserved Value 35 Reserved for SourceUserContext. Use of this field for any other purpose may result in data being overwritten by future functionality.
RV36	VARCHAR(255)	Reserved Value 36 Reserved for DataContext. Use of this field for any other purpose may result in data being overwritten by future functionality.
RV37	VARCHAR(255)	Reserved Value 37 Reserved for SourceFunction. Use of this field for any other purpose may result in data being overwritten by future functionality.
RV38	VARCHAR(255)	Reserved Value 38 Reserved for SourceOperationalContext. Use of this field for any other purpose may result in data being overwritten by future functionality.
RV40 - 43	VARCHAR(255)	Reserved Value 40 - 43 Reserved for future use by e-Security. Use of this field for any other purpose may result in data being overwritten by future functionality.
RV44	VARCHAR(255)	Reserved Value 44 Reserved for DestinationThreatLevel. Use of this field for any other purpose may result in data being overwritten by future functionality.
RV45	VARCHAR(255)	Reserved Value 45 Reserved for DestinationUserContext. Use of this field for any other purpose may result in data being overwritten by future functionality.

Column Name	Datatype	Comment
RV46	VARCHAR(255)	Reserved Value 46 Reserved for VirusStatus. Use of this field for any other purpose may result in data being overwritten by future functionality.
RV47	VARCHAR(255)	Reserved Value 47 Reserved for future use by e-Security. Use of this field for any other purpose may result in data being overwritten by future functionality.
RV48	VARCHAR(255)	Reserved Value 48 Reserved for DestinationOperationalContext. Use of this field for any other purpose may result in data being overwritten by future functionality.
RV49	VARCHAR(255)	Reserved Value 49 Reserved for future use by e-Security. Use of this field for any other purpose may result in data being overwritten by future functionality.
REFERENCE_ID 01 - 20	BIGINT	Reserved for future use by e-Security. Use of this field for any other purpose may result in data being overwritten by future functionality.
CV01 - 10	INT	Custom Value 1 - 10 Reserved for use by Customer, typically for association of Business relevant data
CV11 - 20	DATETIME	Custom Value 11 - 20 Reserved for use by Customer, typically for association of Business relevant data
CV21 - 100	VARCHAR(255)	Custom Value 21 - 100Reserved for use by Customer, typically for association of Business relevant data

EVT_AGENT_RPT_V

View references EVT_AGENT table that stores information about agents.

Column Name	Datatype	Comment
AGENT_ID	bigint	Agent identifier
AGENT	varchar(64)	Agent name
PORT	varchar(64)	Agent port
REPORT_NAME	varchar(255)	Reporter name
PRODUCT_NAME	varchar(255)	Product name
SENSOR_NAME	varchar(255)	Sensor name
SENSOR_TYPE	varchar(5)	Sensor type: H - host-based N - network-based V - virus O - other
DEVICE_CTGRY	varchar(255)	Device category
SOURCE_UUID	uniqueidentifier	Source component Universal Unique Identifier (UUID)
DATE_CREATED	datetime	Insert date
DATE_MODIFIED	datetime	Last update date
CREATED_BY	int	Inserting user ID.
MODIFIED_BY	int	Last updating user ID.

EVT_ASSET_RPT_V

View references EVT_ASSET table that stores asset information.

Column Name	Datatype	Comment
EVENT_ASSET_ID	bigint	Event asset identifier
ASSET_NAME	varchar(255)	Asset name
PHYSICAL_ASSET_NAME	varchar(255)	Physical asset name
REFERENCE_ASSET_ID	varchar(100)	Reference asset identifier, links to source asset management system.
MAC_ADDRESS	varchar(100)	MAC address
RACK_NUMBER	varchar(50)	Rack number
ROOM_NAME	varchar(100)	Room name
BUILDING_NAME	varchar(255)	Building name
CITY	varchar(100)	City
STATE	varchar(100)	State
COUNTRY	varchar(100)	Country
ZIP_CODE	varchar(50)	Zip code
ASSET_CATEGORY_NAME	varchar(100)	Asset category name
NETWORK_IDENTITY_NAME	varchar(255)	Asset network identity name
ENVIRONMENT_IDENTITY_NAME	varchar(255)	Environment name
ASSET_VALUE_NAME	varchar(50)	Asset value name
CRITICALITY_NAME	varchar(50)	Asset criticality name
SENSITIVITY_NAME	varchar(50)	Asset sensitivity name

Column Name	Datatype	Comment
CONTACT_NAME_1	varchar(255)	Name of contact person/organization 1
CONTACT_NAME_2	varchar(255)	Name of contact person/organization 2
ORGANIZATION_NAME_1	varchar(100)	Asset owner organization level 1
ORGANIZATION_NAME_2	varchar(100)	Asset owner organization level 2
ORGANIZATION_NAME_3	varchar(100)	Asset owner organization level 3
ORGANIZATION_NAME_4	varchar(100)	Asset owner organization level 4
DATE_CREATED	datetime	Insert date
DATE_MODIFIED	datetime	Last update date
CREATED_BY	int	Inserting user ID.
MODIFIED_BY	int	Last updating user ID.

EVT_DEST_EVT_NAME_SMRY_1_RPT_V

View summarizes event count by destination, taxonomy, event name, severity and event time.

Column Name	Datatype	Comment
DESTINATION_IP	int	Destination IP address
DESTINATION_EVENT_ASSET_ID	bigint	Event asset identifier
TAXONOMY_ID	bigint	Taxonomy identifier
EVENT_NAME_ID	bigint	Event name identifier
SEVERITY	int	Event severity
CUSTOMER_ID	bigint	Customer identifier
EVT_TIME	datetime	Event time
EVT_COUNT	int	Event count
DATE_CREATED	datetime	Insert date
DATE_MODIFIED	datetime	Last update date
CREATED_BY	int	Inserting user ID.
MODIFIED_BY	int	Last updating user ID.

EVT_DEST_SMRY_1_RPT_V

View contains event destination summary information.

Column Name	Datatype	Comment
DESTINATION_IP	int	Destination IP address
DESTINATION_EVENT_ASSET_ID	bigint	Event asset identifier
DESTINATION_PORT	varchar(32)	Destination port
DESTINATION_USR_ID	bigint	Destination user identifier
TAXONOMY_ID	bigint	Taxonomy identifier
EVENT_NAME_ID	bigint	Event name identifier
RESOURCE_ID	bigint	Resource identifier
AGENT_ID	bigint	Agent identifier

Column Name	Datatype	Comment
PROTOCOL_ID	bigint	Protocol identifier
SEVERITY	int	Event severity
CUSTOMER_ID	bigint	Customer identifier
EVENT_TIME	datetime	Event time
EVENT_COUNT	int	Event count
DATE_CREATED	datetime	Insert date
DATE_MODIFIED	datetime	Last update date
CREATED_BY	int	Inserting user ID.
MODIFIED_BY	int	Last updating user ID.

EVT_DEST_TXNMY_SMRY_1_RPT_V

View summarizes event count by destination, taxonomy, severity and event time.

Column Name	Datatype	Comment
DESTINATION_IP	int	Destination IP address
DESTINATION_EVENT_ASSET_ID	bigint	Event asset identifier
TAXONOMY_ID	bigint	Taxonomy identifier
SEVERITY	int	Event severity
CUSTOMER_ID	bigint	Customer identifier
EVENT_TIME	datetime	Event time
EVENT_COUNT	int	Event count
DATE_CREATED	datetime	Insert date
DATE_MODIFIED	datetime	Last update date
CREATED_BY	int	Inserting user ID.
MODIFIED_BY	int	Last updating user ID.

EVT_NAME_RPT_V

View references EVT_NAME table that stores event name information.

Column Name	Datatype	Comment
EVENT_NAME_ID	bigint	Event name identifier
EVENT_NAME	varchar(255)	Event name
DATE_CREATED	datetime	Insert date
DATE_MODIFIED	datetime	Last update date
CREATED_BY	int	Inserting user ID.
MODIFIED_BY	int	Last updating user ID.

EVT_PORT_SMRY_1_RPT_V

View summarizes event count by destination port, severity and event time.

Column Name	Datatype	Comment
DESTINATION_PORT	Varchar(32)	Destination port
SEVERITY	int	Event severity
CUSTOMER_ID	bigint	Customer identifier
EVENT_TIME	datetime	Event time
EVENT_COUNT	int	Event count
DATE_CREATED	datetime	Insert date

Column Name	Datatype	Comment
DATE_MODIFIED	datetime	Last update date
CREATED_BY	int	Inserting user ID.
MODIFIED_BY	int	Last updating user ID.

EVT_PRTCL_RPT_V

View references EVT_PRTCL table that stores event protocol information.

Column Name	Datatype	Comment
PROTOCOL_ID	bigint	Protocol identifier
PROTOCOL_NAME	varchar(255)	Protocol name
DATE_CREATED	datetime	Insert date
DATE_MODIFIED	datetime	Last update date
CREATED_BY	int	Inserting user ID.
MODIFIED_BY	int	Last updating user ID.

EVT_RSRC_RPT_V

View references EVT_RSRC table that stores event resource information.

Column Name	Datatype	Comment
RESOURCE_ID	bigint	Resource identifier
RESOURCE_NAME	varchar(255)	Resource name
SUB_RESOURCE_NAME	varchar(255)	Subresource name
DATE_CREATED	datetime	Insert date
DATE_MODIFIED	datetime	Last update date
CREATED_BY	int	Inserting user ID.
MODIFIED_BY	int	Last updating user ID.

EVT_SEV_SMRY_1_RPT_V

View summarizes event count by severity and event time.

Column Name	Datatype	Comment
SEVERITY	int	Event severity
CUSTOMER_ID	bigint	Customer identifier
EVENT_TIME	datetime	Event time
EVENT_COUNT	int	Event count
DATE_CREATED	datetime	Insert date
DATE_MODIFIED	datetime	Last update date
CREATED_BY	int	Inserting user ID.
MODIFIED_BY	int	Last updating user ID.

EVT_SRC_SMRY_1_RPT_V

View contains event source and destination summary information.

Column Name	Datatype	Comment
SOURCE_IP	int	Source IP address
SOURCE_EVENT_ASSET_ID	bigint	Event asset identifier
SOURCE_PORT	varchar(32)	Source port
SOURCE_USER_ID	bigint	User identifier

Column Name	Datatype	Comment
TAXONOMY _ID	bigint	Taxonomy identifier
EVENT_NAME_ID	bigint	Event name identifier
RESOURCE_ID	bigint	Resource identifier
AGENT_ID	bigint	Agent identifier
PROTOCOL _ID	bigint	Protocol identifier
SEVERITY	int	Event severity
CUSTOMER_ID	bigint	Customer identifier
EVENT_TIME	datetime	Event time
EVENT_COUNT	int	Event count
DATE_CREATED	datetime	Insert date
DATE_MODIFIED	datetime	Last update date
CREATED_BY	int	Inserting user ID.
MODIFIED_BY	int	Last updating user ID.

EVT_TXNMY_RPT_V

View references EVT_TXNMY table that stores event taxonomy information.

Column Name	Datatype	Comment
TAXONOMY _ID	bigint	Taxonomy identifier
TAXONOMY _ LEVEL _1	varchar(100)	Taxonomy level 1
TAXONOMY _ LEVEL _2	varchar(100)	Taxonomy level 2
TAXONOMY _ LEVEL _3	varchar(100)	Taxonomy level 3
TAXONOMY _ LEVEL _4	varchar(100)	Taxonomy level 4
DATE_CREATED	datetime	Insert date
DATE_MODIFIED	datetime	Last update date
CREATED_BY	int	Inserting user ID.
MODIFIED_BY	int	Last updating user ID.
TAXONOMY _ID	bigint	Taxonomy identifier

EVT_USR_RPT_V

View references EVT_USR table that stores event user information.

Column Name	Datatype	Comment
USER_ID	bigint	User identifier
USER_NAME	varchar(255)	User name
DATE_CREATED	datetime	Insert date
DATE_MODIFIED	datetime	Last update date
CREATED_BY	int	Inserting user ID.
MODIFIED_BY	int	Last updating user ID.
USER_ID	bigint	User identifier

EXTERNAL_DATA_RPT_V

View references EXTERNAL_DATA table that stores external data.

Column Name	Datatype	Comment
EXTERNAL_DATA_ID	int	External data identifier
SOURCE_NAME	varchar(50)	Source name
SOURCE_DATA_ID	varchar(255)	Source data identifier

Column Name	Datatype	Comment
EXTERNAL_DATA	text	External data
EXTERNAL_DATA_TYPE	varchar(10)	External data type
DATE_CREATED	datetime	Insert date
DATE_MODIFIED	datetime	Last update date
CREATED_BY	int	Inserting user ID.
MODIFIED_BY	int	Last updating user ID.

HIST_EVENTS_RPT_V

View historical events (events restored from archives).

HIST_INCIDENTS_RPT_V

View historical incidents (incidents restored from archives).

IMAGES_RPT_V

View references IMAGES table that stores system overview image information.

Column Name	Datatype	Comment
NAME	VARCHAR(128)	Image name
TYPE	VARCHAR(64)	Image type
DATA	TEXT	Image data
DATE_CREATED	DATETIME	Insert date
DATE_MODIFIED	DATETIME	Last update date
CREATED_BY	INT	Inserting user ID
MODIFIED_BY	INT	Last updating user ID

INCIDENTS_ASSETS_RPT_V

View references INCIDENTS_ASSETS table that stores information about the assets that makeup incidents created in the Sentinel Console.

Column Name	Datatype	Comment
INC_ID	INT	Incident identifier – sequence number
ASSET_ID	uniqueidentifier	Asset Universal Unique Identifier (UUID)
DATE_CREATED	DATETIME	Insert date
DATE_MODIFIED	DATETIME	Last update date
CREATED_BY	INT	Inserting user ID
MODIFIED_BY	INT	Last updating user ID

INCIDENTS_EVENTS_RPT_V

View references INCIDENTS_EVENTS table that stores information about the events that makeup incidents created in the Sentinel Console.

Column Name	Datatype	Comment
INC_ID	INT	Incident identifier – sequence number
EVT_ID	uniqueidentifier	Event Universal Unique Identifier (UUID)

Column Name	Datatype	Comment
EVT_TIME	DATETIME	Event time
DATE_CREATED	DATETIME	Insert date
DATE_MODIFIED	DATETIME	Last update date
CREATED_BY	INT	Inserting user ID
MODIFIED_BY	INT	Last updating user ID

INCIDENTS_RPT_V

View references INCIDENTS table that stores information describing the details of incidents created in the Sentinel Console.

Column Name	Datatype	Comment
INC_ID	INT	Incident identifier – sequence number
NAME	VARCHAR(255)	Incident name
SEVERITY	INT	Incident severity
STT_ID	INT	Incident State ID
SEVERITY_RATING	VARCHAR(32)	Average of all the event severities that comprise an incident.
VULNERABILITY_RATING	VARCHAR(32)	Reserved for future use by e-Security. Use of this field for any other purpose may result in data being overwritten by future functionality.
CRITICALITY_RATING	VARCHAR(32)	Reserved for future use by e-Security. Use of this field for any other purpose may result in data being overwritten by future functionality.
DATE_CREATED	DATETIME	Insert date
DATE_MODIFIED	DATETIME	Last update date
CREATED_BY	INT	Inserting user ID
MODIFIED_BY	INT	Last updating user ID
INC_DESC	varchar(4000)	Incident description
INC_PRIORITY	int	Incident priority
INC_CAT	varchar(255)	Incident category
INC_RES	varchar(4000)	Incident resolution

INCIDENTS_VULN_RPT_V

View references INCIDENTS_VULN table that stores information about the vulnerabilities that makeup incidents created in the Sentinel Console.

Column Name	Datatype	Comment
INC_ID	INT	Incident identifier – sequence number
VULN_ID	uniqueidentifier	Vulnerability Universal Unique Identifier (UUID)
DATE_CREATED	DATETIME	Insert date

Column Name	Datatype	Comment
DATE_MODIFIED	DATETIME	Last update date
CREATED_BY	INT	Inserting user ID
MODIFIED_BY	INT	Last updating user ID

L_STAT_RPT_V

View references L_STAT table that stores statistical information.

Column Name	Datatype	Comment
RES_NAME	VARCHAR(32)	Resource name
STATS_NAME	VARCHAR(32)	Statistic name
STATS_VALUE	VARCHAR(32)	Value of the statistic
OPEN_TOT_SECS	NUMERIC	Number of seconds since 1970.

LOGS_RPT_V

View references LOGS_RPT table that stores logging information.

LOGS Table		
Column Name	**Datatype**	**Comment**
LOG_ID	NUMBER	Sequence number
TIME	DATE	Date of Log
MODULE	VARCHAR(64)	Module log is for
TEXT	VARCHAR(4000)	Log text

NETWORK_IDENTITY_RPT_V

View references NETWORK_IDENTITY_LKUP table that stores asset network identity information.

Column Name	Datatype	Comment
NETWORK_IDENTITY_CD	varchar(5)	Network identity code
NETWORK_IDENTITY_NAME	varchar(255)	Network identify name
DATE_CREATED	datetime	Insert date
DATE_MODIFIED	datetime	Last update date
CREATED_BY	int	Inserting user ID
MODIFIED_BY	int	Last updating user ID

ORGANIZATION_RPT_V

View references ORGANIZATION table that stores organization (asset) information.

Column Name	Datatype	Comment
ORGANIZATION_ID	uniqueidentifier	Organization identifier
ORGANIZATION_NAME	varchar(100)	Organization name
CUSTOMER_ID	bigint	Customer identifier
DATE_CREATED	datetime	Insert date
DATE_MODIFIED	datetime	Last update date
CREATED_BY	int	Inserting user ID
MODIFIED_BY	int	Last updating user ID

PERSON_RPT_V

View references PERSION table that stores personal (asset) information.

Column Name	Datatype	Comment
PERSON_ID	uniqueidentifier	Person identifier
FIRST_NAME	varchar(255)	First name
LAST_NAME	varchar(255)	Last name
CUSTOMER_ID	bigint	Customer identifier
PHONE_NUMBER	varchar(50)	Phone number
EMAIL_ADDRESS	varchar(255)	Email address
DATE_CREATED	datetime	Insert date
DATE_MODIFIED	datetime	Last update date
CREATED_BY	int	Inserting user ID
MODIFIED_BY	int	Last updating user ID

PHYSICAL_ASSET_RPT_V

View references PHYSICAL_ASSET table that stores physical asset information.

Column Name	Datatype	Comment
PHYSICAL_ASSET_ID	uniqueidentifier	Physical asset identifier
CUSTOMER_ID	int	Customer identifier
LOCATION_ID	bigint	Location identifier
HOST_NAME	varchar(255)	Host name
IP_ADDRESS	int	IP address
NETWORK_IDENTITY_CD	varchar(5)	Network identity code
MAC_ADDRESS	varchar(100)	MAC address
RACK_NUMBER	varchar(50)	Rack number
ROOM_NAME	varchar(100)	Room name
DATE_CREATED	datetime	Insert date
DATE_MODIFIED	datetime	Last update date
CREATED_BY	int	Inserting user ID
MODIFIED_BY	int	Last updating user ID

PRODUCT_RPT_V

View references PRDT table that stores asset product information.

Column Name	Datatype	Comment
PRODUCT _ID	bigint	Product identifier
PRODUCT _NAME	varchar(255)	Product name
PRODUCT _VERSION	varchar(100)	Product version
VENDOR _ID	bigint	Vendor identifier
DATE_CREATED	datetime	Insert date
DATE_MODIFIED	datetime	Last update date
CREATED_BY	int	Inserting user ID
MODIFIED_BY	int	Last updating user ID

ROLE_RPT_V

View references ROLE_LKUP table that stores user role (asset) information.

Column Name	**Datatype**	**Comment**
ROLE_CODE	varchar(5)	Role code
ROLE_NAME	varchar(255)	Role name
DATE_CREATED	datetime	Insert date
DATE_MODIFIED	datetime	Last update date
CREATED_BY	int	Inserting user ID
MODIFIED_BY	int	Last updating user ID

SENSITIVITY_RPT_V

View references SENSITIVITY_LKUP table that stores asset sensitivity information.

Column Name	**Datatype**	**Comment**
SENSITIVITY_CODE	varchar(5)	Asset sensitivity code
SENSITIVITY_NAME	varchar(50)	Asset sensitivity name
DATE_CREATED	datetime	Insert date
DATE_MODIFIED	datetime	Last update date
CREATED_BY	int	By user ID
MODIFIED_BY	int	By user ID

STATES_RPT_V

View references STATES table that stores definitions of states defined by applications or context.

Column Name	**Datatype**	**Comment**
STT_ID	INT	State ID – sequence number
CONTEXT	VARCHAR(64)	Context of the state. That is case, incident, user.
NAME	VARCHAR(64)	Name of the state.
TERMINAL_FLAG	VARCHAR(1)	Indicates if state of incident is resolved.
DATE_CREATED	DATETIME	Insert date
DATE_MODIFIED	DATETIME	Last update date
MODIFIED_BY	INT	Inserting user ID
CREATED_BY	INT	Last updating user ID

UNASSIGNED_INCIDENTS_RPT_V View

View references CASES and INCIDENTS tables to report on unassigned cases.

Name	**Datatype**
INC_ID	INT
NAME	VARCHAR(255)
SEVERITY	INT
STT_ID	INT
SEVERITY_RATING	VARCHAR(32)
VULNERABILITY_RATING	VARCHAR(32)
CRITICALITY_RATING	VARCHAR(32)
DATE_CREATED	DATETIME
DATE_MODIFIED	DATETIME

Name	Datatype
CREATED_BY	INT
MODIFIED_BY	INT
INC_DESC	VARCHAR(4000)
INC_PRIORITY	INT
INC_CAT	VARCHAR(255)
INC_RES	VARCHAR(4000)

USERS_RPT_V

View references USERS table that lists all users of the application. The users will also be created as database users to accommodate 3rd party reporting tools.

Column Name	Datatype	Comment
USR_ID	INT	User identifier – Sequence number
NAME	VARCHAR(64)	Short, unique user name used as a login
CNT_ID	INT	Contact ID – Sequence number
STT_ID	INT	State ID. Status is either active or inactive.
DESCRIPTION	VARCHAR(512)	Comments
DATE_CREATED	DATETIME	Insert date
DATE_MODIFIED	DATETIME	Last update date
CREATED_BY	INT	Inserting user ID
MODIFIED_BY	INT	Last updating user ID
PERMISSIONS	VARCHAR(4000)	Permissions currently assigned to the Sentinel user
FILTER	VARCHAR(128)	Current security filter assigned to the Sentinel user
UPPER_NAME	VARCHAR(64)	User name in upper case
DOMAIN_AUTH_IND	Bit	Domain authentication indication

VENDOR_RPT_V

View references VNDR table that stores information about asset product vendors.

Column Name	Datatype	Comment
VENDOR_ID	bigint	Vendor identifier
VENDOR_NAME	varchar(255)	Vendor name
DATE_CREATED	datetime	Insert date
DATE_MODIFIED	datetime	Last update date
CREATED_BY	int	Inserting user ID
MODIFIED_BY	int	Last updating user ID

VULN_CALC_SEVERITY_RPT_V

View references VULN_RSRC and VULN to calculate eSecurity vulnerability severity rating base on current vulnerabilities.

Column Name	Datatype
RSRC_ID	uniqueidentifier

Column Name	Datatype
IP	VARCHAR(32)
HOST_NAME	VARCHAR(255)
CRITICALITY	int
ASSIGNED_VULN_SEVERITY	int
VULN_COUNT	Count of Vulnerabilities for specified Resource
CALC_SEVERITY	Calculated Severity based on ASSIGNED_VULN_SEVERITY and CRITICALITY

VULN_CODE_RPT_V

View references VULN_CODE table that stores industry assigned vulnerability codes such as Mitre's CVEs and CANs.

Column Name	Datatype
VULN_CODE_ID	VARCHAR(36)
VULN_ID	VARCHAR(36)
VULN_CODE_TYPE	VARCHAR(64)
VULN_CODE_VALUE	VARCHAR(255)
URL	VARCHAR(512)
DATE_CREATED	DATETIME
DATE_MODIFIED	DATETIME
CREATED_BY	INT
MODIFIED_BY	INT

VULN_INFO_RPT_V

View references VULN_INFO table that stores additional information reported during a scan.

Column Name	Datatype
VULN_INFO_ID	VARCHAR(36)
VULN_ID	VARCHAR(36)
VULN_INFO_TYPE	VARCHAR(36)
VULN_INFO_VALUE	VARCHAR(2000)
DATE_CREATED	DATETIME
DATE_MODIFIED	DATETIME
CREATED_BY	INT
MODIFIED_BY	INT

VULN_RPT_V

View references VULN table that stores information of scanned system. Each scanner will have its own entry for each system.

Column Name	Datatype
VULN_ID	VARCHAR(36)
RSRC_ID	VARCHAR(36)
PORT_NAME	VARCHAR(64)
PORT_NUMBER	INT

Column Name	Datatype
NETWORK_PROTOCOL	INT
APPLICATION_PROTOCOL	VARCHAR(64)
ASSIGNED_VULN_SEVERITY	INT
COMPUTED_VULN_SEVERITY	INT
VULN_DESCRIPTION	CLOB
VULN_SOLUTION	CLOB
VULN_SUMMARY	VARCHAR(1000)
BEGIN_EFFECTIVE_DATE	DATETIME
END_EFFECTIVE_DATE	DATETIME
DETECTED_OS	VARCHAR(64)
DETECTED_OS_VERSION	VARCHAR(64)
SCANNED_APP	VARCHAR(64)
SCANNED_APP_VERSION	VARCHAR(64)
VULN_USER_NAME	VARCHAR(64)
VULN_USER_DOMAIN	VARCHAR(64)
VULN_TAXONOMY	VARCHAR(1000)
SCANNER_CLASSIFICATION	VARCHAR(255)
VULN_NAME	VARCHAR(300)
VULN_MODULE	VARCHAR(64)
DATE_CREATED	DATETIME
DATE_MODIFIED	DATETIME
CREATED_BY	INT
MODIFIED_BY	INT

VULN_RSRC_RPT_V

View references VULN_RSRC table that stores each resource scanned for a particular scan.

Column Name	Datatype
RSRC_ID	VARCHAR(36)
SCANNER_ID	VARCHAR(36)
IP	VARCHAR(32)
HOST_NAME	VARCHAR(255)
LOCATION	VARCHAR(128)
DEPARTMENT	VARCHAR(128)
BUSINESS_SYSTEM	VARCHAR(128)
OPERATIONAL_ENVIRONMENT	VARCHAR(64)
CRITICALITY	INT
REGULATION	VARCHAR(128)
REGULATION_RATING	VARCHAR(64)
DATE_CREATED	DATETIME
DATE_MODIFIED	DATETIME
CREATED_BY	INT
MODIFIED_BY	INT

VULN_RSRC_SCAN_RPT_V

View references VULN_RSRC_SCAN table that stores each resource scanned for a particular scan.

Column Name	Datatype
RSRC_ID	VARCHAR(36)
SCAN_ID	VARCHAR(36)
DATE_CREATED	DATETIME
DATE_MODIFIED	DATETIME
CREATED_BY	INT
MODIFIED_BY	INT

VULN_SCAN_RPT_V

View references table that stores information pertaining to scans.

Column Name	Datatype
SCAN_ID	VARCHAR(36)
SCANNER_ID	VARCHAR(36)
SCAN_TYPE	VARCHAR(10)
SCAN_START_DATE	DATETIME
SCAN_END_DATE	DATETIME
CONSOLIDATION_SERVER	VARCHAR(64)
DATE_CREATED	DATETIME
DATE_MODIFIED	DATETIME
CREATED_BY	INT
MODIFIED_BY	INT

VULN_SCAN_VULN_RPT_V

View references VULN_SCAN_VULN table that stores vulnerabilities detected during scans.

Column Name	Datatype
SCAN_ID	VARCHAR(36)
VULN_ID	VARCHAR(36)
DATE_CREATED	DATETIME
DATE_MODIFIED	DATETIME
CREATED_BY	INT
MODIFIED_BY	INT

VULN_SCANNER_RPT_V

View references VULN_SCANNER table that stores information about vulnerability scanners.

Column Name	Datatype
SCANNER_ID	VARCHAR(36)
PRODUCT_NAME	VARCHAR(100)
PRODUCT_VERSION	VARCHAR(64)
SCANNER_TYPE	VARCHAR(64)
VENDOR	VARCHAR(100)

Column Name	**Datatype**
SCANNER_INSTANCE	VARCHAR(64)
DATE_CREATED	DATETIME
DATE_MODIFIED	DATETIME
CREATED_BY	INT
MODIFIED_BY	INT

Appendix A – Sentinel Copyright Information

e-Security Sentinel™ 5

Sentinel 5 may contain the following third-party technologies:

- **Apache Axis** and **Apache Tomcat**, Copyright © 1999 to 2005, Apache Software Foundation. For more information, disclaimers and restrictions, see http://www.apache.org/licenses/
- **ANTLR.** For more information, disclaimers and restrictions, see http://www.antlr.org
- **Boost**, Copyright © 1999, Boost.org.
- **Bouncy Castle**, Copyright © 2000-2004, the Legion of Bouncy Castle. For more information, disclaimers and restrictions see http://www.bouncycastle.org.
- **Checkpoint.** Copyright © Check Point Software Technologies Ltd.
- **Concurrent**, utility package. Copyright © Doug Lea. Used without CopyOnWriteArrayList and ConcurrentReaderHashMap classes.
- **Crypto++ Compilation**. Copyright © 1995-2003, Wei Dai, incorporating the following copyrighted work: **mars.cpp** by Brian Gladman and Sean Woods. For more information, disclaimers and restrictions see http://www.eskimo.com/~weidai/License.txt.
- **Crystal Reports Developer and Crystal Reports Server**. Copyright © 2004 Business Objects Software Limited.
- **DataDirect Technologies Corp**. Copyright © 1991-2003.
- **edpFTPj**, licensed under the Lesser GNU Public License. For more information, disclaimers and restrictions see http://www.enterprisedt.com/products/edtftpj/purchase.html.
- **Enhydra Shark**, licensed under the Lesser General Public License available at: http://shark.objectweb.org/license.html.
- **ICEsoft ICEbrowser**. ICEsoft Technologies, Inc. Copyright © 2003-2004.
- **ILOG, Inc**. Copyright © 1999-2004.
- **InstallShield Universal**. Copyright © 1996–2005, Macrovision Corporation and/or Macrovision Europe Ltd.
- **Java 2 Platform, Standard Edition**. Copyright © Sun Microsystems, Inc. For more information, disclaimers and restrictions see http://java.sun.com/j2se/1.4.2/j2re-1_4_2_10-license.txt.

  The Java 2 Platform may also contain the following third-party products:

  - CoolServlets © 1999
  - DES and 3xDES © 2000 by Jef Poskanzer
  - Crimson © 1999-2000 The Apache Software Foundation
  - Xalan J2 © 1999-2000 The Apache Software Foundation
  - NSIS 1.0j © 1999-2000 Nullsoft, Inc.
  - Eastman Kodak Company © 1992
  - Lucinda, a registered trademark or trademark of Bigelow and Holmes

  - Taligent, Inc.
  - IBM, some portions available at: http://oss.software.ibm.com/icu4j/

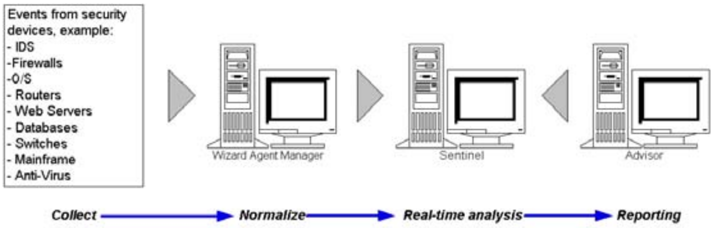
  For more information regarding these third-party technologies and their associated disclaimers and restrictions, see: http://java.sun.com/j2se/1.4.2/j2se-1_4_2-thirdpartylicensereadme.txt.

- **JavaBeans Activation Framework (JAF)**. Copyright © Sun Microsystems, Inc. For more information, disclaimers and restrictions see http://www.java.sun.com/products/javabeans/glasgow/jaf.html and click download > license.
- **JavaMail**. Copyright © Sun Microsystems, Inc. For more information, disclaimers and restrictions see http://www.java.sun.com/products/javamail/downloads/index.html and click download > license.
- **Java Ace**, by Douglas C. Schmidt and his research group at Washington University and **Tao (with ACE wrappers)** by Douglas C. Schmidt and his research group at Washington University, University of California, Irvine and Vanderbilt University. Copyright © 1993-2005. For more information, disclaimers and restrictions see http://www.cs.wustl.edu/~schmidt/ACE-copying.html and http://www.cs.wustl.edu/~pjain/java/ace/JACE-copying.html
- **Java Authentication and Authorization Service Modules**, licensed under the Lesser General Public License. For more information, disclaimers and restrictions see http://free.tagish.net/jaas/index.jsp.
- **Java Network Launching Protocol (JNLP).** Copyright © Sun Microsystems, Inc. For more information, disclaimers and restrictions, please see http://www.java.sun.com/products/javawebstart/download-jnlp.html and click download > license.
- **Java Service Wrapper**. Portions copyrighted as follows: Copyright © 1999, 2004 Tanuki Software and Copyright © 2001 Silver Egg Technology. For more information, disclaimers and restrictions, see http://wrapper.tanukisoftware.org/doc/english/license.html.
- **JIDE**. Copyright © 2002 to 2005, JIDE Software, Inc.
- **jTDS** is licensed under the Lesser GNU Public License. For more information, disclaimers and restrictions see http://jtds.sourceforge.net/.
- **MDateSelector**. Copyright © 2005, Martin Newstead, licensed under the Lesser General Public License. For more information, disclaimers and restrictions see http://web.ukonline.co.uk/mseries.
- **Monarch Charts**. Copyright © 2005, Singleton Labs.
- **Net-SNMP**. Portions of the code are copyrighted by various entities, which reserve all rights. Copyright © 1989, 1991, 1992 by Carnegie Mellon University; Copyright © 1996, 1998 to 2000, the Regents of the University of California; Copyright © 2001 to 2003 Networks Associates Technology, Inc.; Copyright © 2001 to 2003, Cambridge Broadband, Ltd.; Copyright © 2003 Sun Microsystems, Inc. and Copyright © 2003 to 2004, Sparta, Inc. For more information, disclaimers and restrictions, see http://net-snmp.sourceforge.net.
- **The OpenSSL Project**. Copyright © 1998-2004. the Open SSL Project. For more information, disclaimers and restrictions, see http://www.openssl.org.
- **Oracle Help for Java**. Copyright © 1994-2006, Oracle Corporation.
- **RoboHELP Office**. Copyright © Adobe Systems Incorporated, formerly Macromedia.

- **Skin Look and Feel (SkinLF)**. Copyright © 2000-2006 L2FProd.com. Licensed under the Apache Software License. For more information, disclaimers and restrictions see https://skinlf.dev.java.net/.
- **Sonic Software Corporation**. Copyright © 2003-2004. The SSC software contains security software licensed from RSA Security, Inc.
- **Tinyxml**. For more information, disclaimers and restrictions see http://grinninglizard.com/tinyxmldocs/index.html.
- **SecurityNexus**. Copyright © 2003 to 2006. SecurityNexus, LLC. All rights reserved.
- **Xalan** and **Xerces**, both of which are licensed by the Apache Software Foundation Copyright © 1999-2004. For more information, disclaimers and restrictions see http://xml.apache.org/dist/LICENSE.txt.
- **yWorks.** Copyright © 2003 to 2006, yWorks.

NOTE: As of the publication of this documentation, the above links were active. In the event you find that any of the above links are broken or the linked webpages are inactive, please contact e-Security's Office of the Counsel at 1921 Gallows Road, Vienna, VA 22182. 703-852-8000.

Glossary

activity_container.xml	Specifies the execution and configuration for the activity service
Advanced Correlation Rule	Allows you to create a correlation rule that incorporates all of the features of the simple correlation rule, as well as send an event when a set of events has meta-tag values that are different, such as a sensor being inside or outside the firewall. For example, an Advanced Correlation rule can look for events from the same source IP address to the same destination IP address with the same event name that occur both inside and outside a firewall (meaning the attack may have made it through the firewall).
Advisor	An integrated system with SecurityNexus database of vulnerabilities to provide a cross-reference between real-time events and known vulnerabilities.
Agent	An agent is the receptor that collects and normalizes raw events from security devices and programs and outputs normalized events that can be correlated, reported and used for incident response.  Events from security devices, example: - IDS -Firewalls -O/S - Routers - Web Servers - Databases - Switches - Mainframe - Anti-Virus Wizard Agent Manager Sentinel Advisor Collect Normalize Real-time analysis Reporting There are three levels of Agents, they are: ▪ Supported Agents (T1) ▪ Documented Agents (T2) ▪ Sample Agents (T3) Agents are made of: ▪ template files ▪ parameter files ▪ lookup files ▪ mapping files

Agent Builder	A GUI that allows you to create rules based agents to collect, filter and normalize data from many different sources and securely communicate relevant information to the e-Security Sentinel Server that can be used to monitor traffic.
Agent Engine	Processes the template logic for each port. An agent engine runs a corresponding port.
Agent Manager	The wizard back-end that manages agents and system status messages.
Aggregation and Event Normalization	Aggregation is the process of taking individual low-relevant data items and combining them, resulting in a data item that could possibly be high importance. The individual parts of event, such as the event name, event date, Source IP, Destination IP, UUID, Sensor Type and so on, by themselves may not have much meaning. However, put them together and an event is created that could be an event of interest that could possibly be an attack on the network resulting in a possible exploit of an asset. Saving an entire event causes the storage of duplicate information. For example, in a non-aggregated system for ten events that are identical, except for event date will store each event resulting in identical data items (event name, Sensor Type, etc...) being saved ten times. Aggregation will store the identical data items just one time and then keep a running account for an hour. Event data is transformed, summarized and stored into summary tables. Summary reports can then run against the pre-computed summaries, which makes queries less contention on the real-time event tables. The event aggregation engine captures binary event data, transforms it into a normalized event structure and summarizes it based on a set of pre-defined set of summary definitions. The event aggregation engine processes events in a near real-time fashion with minimum overhead to the real-time e-Security system.
Analysis	In Sentinel Control Center, allows for historical reporting. Historical and vulnerability reports are published on a Crystal$^{®}$ web server, these run directly against the database and they appear on the Analysis and Advisor tabs on the Navigator bar in the Sentinel Control Center.

Asset Management	Purpose behind Asset Management to is to link an event or events to assets and vulnerability information to perform a method to protect the organization’s assets efficiently. There are two types of assets, physical and soft. Physical Assets is hardware and Soft Assets is services and applications.
Backout Sequences	See Sequences.
Basic Correlation Rule	Allows you to select any of the meta-tags to create a correlation rule that enables you to count the number of times certain conditions are met within a specific timeframe. For example, a Basic Correlation rule can look for the same source IP address reported five times in five minutes, even if the events are reported from different products, such as an intrusion detection system (IDS) and a firewall.
Business Relevance	See Mapping Service.
CorrelatedEventUUID	The event identifier of the correlated event generated by the rule that has triggered.
Correlation	The process of analyzing security events to identify potential relationships between two or more events. Correlation allows quick association of priority attacks based on common elements of event data. Trends or patterns among lower level events that are designed to operate below security thresholds can be more effectively identified using correlation. e-Security provides you with five types of correlation rules. They are: ▪ Watchlist ▪ Basic Correlation ▪ Advance Correlation ▪ Free Form RuleLg
Correlation Engine	The Correlation Engine performs analysis of incoming events to find patterns of interest and drill-down on correlation events to determine the details that triggered a rule.

Correlation Engine Process (correlation_engine)	The Correlation Engine (correlation_engine) process receives events from the Wizard Agent Manager and publishes correlated events based on user-defined correlation rules.
Data Access Service Process (DAS)	The Data Access Service (DAS) process is Sentinel Server's persistence service and provides a message bus (iSCALE) interface to the database. It provides data driven access to the backend database. It receives XML request from the different Sentinel processes, converts them to a query against the database, processes the result from the database and converts it that back to an XML reply. It supports requests to retrieve events for Quick Query and Event Drill Down, to retrieve vulnerability information and advisor information and to manipulate configuration information. DAS also handles logging of all events being received from the Wizard Agent Manager and requests to retrieve and store configuration information.
das_binary.xml	Specifies configuration parameters for the Data Access Service (DAS), a component of e-Security Database.
das_query.xml	Specifies configuration parameters for the Data Access Service (DAS), a component of e-Security Database.
das_rt.xml	Specifies the configuration for the Active Views function within the Sentinel Control Console
Data Controller	See Data Synchronizer Process.

Data Synchronizer Process (Data Controller)

The Data Synchronizer (data_synchronizer) process manages the modification of configuration data by multiple users. When a user requests to modify data through the Sentinel Control Center, the data record is locked by the data_synchronizer. The details of who locked the data are published to the other active Sentinel Control Centers, and no other users may modify that data. If a Sentinel Control Center is closed before it unlocks any data that it has locked, the locks will timeout.

event

An event is an action or occurrence detected by a security device (external event) or process (internal). Events can be security-related, performance-related or information related. For example, an external event could be an attack detected by an Intrusion Detection System (IDS), a successful login detected by an operating system or a customer-defined situation such as a user accessing a file. Information related events are internal events. Internal events indicate a change in state of a process. For example the stopping of a port.

Event Configuration

Event configuration (Part of Mapping Service) allows you to:

- Enable Regulatory Compliance monitoring
- Enable Policy compliance
- Enable response prioritization
- Enable security data to be analyzed related business operations
- Enhance accountability

Event configuration is the assigning of names to existing labels. For example, renaming Ct2 to City. Changes propagate to filters and correlation rules.

event ID number

A number assigned to an event.

Event Normalization

See Aggregation

Event Router

Event Router performs the event mapping transformation and filtering.

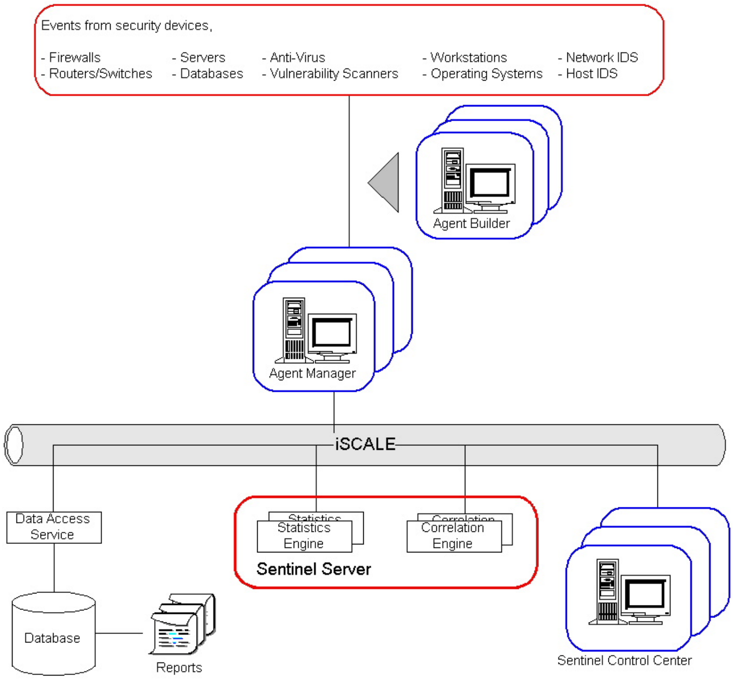
Event Real Time	Ability to monitor events as they are happening and perform queries on these events. You can monitor them in a table form or though a 3-D graphical representation.
Exploit Detection	See Mapping Service.
Filter Engine	See Event Performance Process.
Filters	e-Security filters allows processing of data based on a specific criteria for both events coming into the system and users of the system. There are multiple levels of filtering: ▪ agent - done through the script using the agent builder ▪ global filter - Applied equally to all events generated by all Wizards in the system. Only events that go past the Global Filters are sent to all Sentinel processes. ▪ security filter - Applied to active Users. These filters restrict the events that an active user can observe. These filters are assigned by the Administrator. ▪ display filter - Applied to interface views. These filters let the user define their event windows for real time analysis. These filters are applied by each user. There are two types of filters: ▪ public - Public filters are system-owned. Public filters can be used as security filters or display filters. Security filters are based on user permissions. Display filters determine which events are depicted in the real time event tables, charts and graphs. ▪ private - Private filters are user-owned. Private filters are display filters and are shareable if you have the View Private Filters permission.
Filter Engine	See Event Performance Process.
Incidents	A grouping a set of events together as a whole representing something of interest (group of similar events or set of different events that indicate a pattern of interest such an attack).

Internal Events

See System Events.

iSCALE™

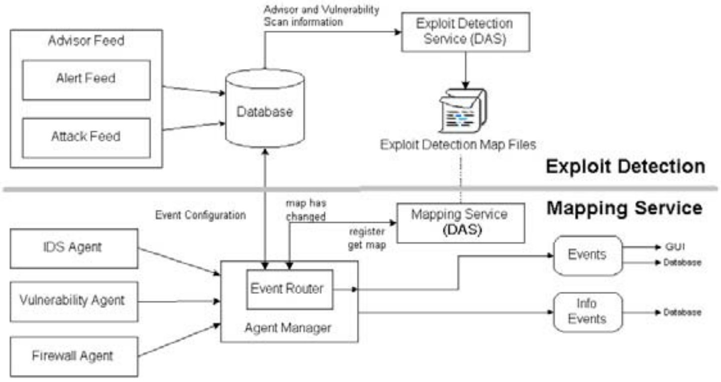
The Message Bus provides a Java Message Service (JMS) framework for inter-process communication. Processes communicate through a broker, which is responsible for routing and buffering messages. Multiple brokers can communicate with each other in order to traverse firewalls and for load balancing.



The following processes communicate with each other through the Message Bus.

- Watchdog
- Event Performance (Filter Engine)
- Event Counts Over Time (Statistics Engine)
- Data Synchronizer (Data Controller)
- Correlation Engine
- RuleLg Checker (Correlation Rule Checker)
- Data Access Service (DAS)
- Query Manager

iTRAC™	iTRAC involves the automation of procedures, the ability to respond to incidents. e-Security provides a workflow management system that provides procedural automation of the SANS Incident Handling process. The main parts iTRAC are: ▪ Worklist Handler – application used to move from one activity to another. ▪ Activity Builder – application used to create your own custom iTRAC ▪ Process Monitor – Monitors the activities (steps) taken to complete a process.
Lookup Files	For Agents, Lookup files are optional tables (.lkp files) against which received values are compared to determine what actions, if any, to take in response to security events. Lookup files contain match clauses, which are used to compare individual strings. Based on the match clauses in a specific lookup file and the data received from sensors, the LOOKUP Command will determine whether the search string is found or is not found. Optionally, parsing commands may be associated with the match string. The parsing commands are executed if a match is found.
Mapping Files	For Agents, Mapping files are optional files (.csv) that allow for fast lookup of key entries. The csv file is a relative path from an agent’s script directory. The editing of these files is currently not within Agent Builder, but the files can be edited using Excel.

Mapping Service	e-Security's Mapping Service enables immediate, actionable notification of attacks on vulnerable systems. It provides a real-time link between events and vulnerability scan results, so that users are notified automatically and immediately when an attack is attempting to exploit a vulnerable system. This enhances the efficiency and effectiveness of incident response, resulting in increased availability of critical systems and highly cost-effective security.  Advisor and Vulnerability Scan information Exploit Detection Service (DAS) Advisor Feed Alert Feed Attack Feed Database Exploit Detection Map Files Exploit Detection Mapping Service Event Configuration map has changed register get map Mapping Service (DAS) IDS Agent Vulnerability Agent Firewall Agent Event Router Agent Manager Events GUI Database Info Events Database
Message Oriented Middleware	See iSCALE™.
MOM	See iSCALE™.
meta data	Meta-data is information about data, pre-defined variable names for meta data. For example, the source IP of an attack is stored in the SourceIP meta-tag. Product names are stored in the ProductName meta-tag. Data used to populate meta-tags is either extracted from event data or is set as part of the agent processing.
meta-tag	Meta-tags store meta data.

Parameter Files	For Agents, Parameter files (.par files) are tables used to define parameter names on the associated run script files. They are used when referenced in the parsing code. Parameters are the equivalent of variables. Parameters are stored as strings. Any numeric value needs to be converted into a string for manipulation. When new values for parameters are entered, they take effect after you build your script. They are merged with the template file when creating a script. Run script file names are displayed in the first row of the table and the parameter names or labels are displayed in the first column of the table. The second row of the table is used to define the icons that appear in the Agent’s tree. The remaining row defines the variables or parameter values to be used for parameter as it relates to the particular script. Values within a parameter file are: ▪ Meta-tags, information and comments – there are over 200 available meta-tags, 100 are user configurable and the rest are reserved. ▪ Rule – set file names appear in the header row of the table, while parameters themselves appear in the first column in the table ▪ Bitmap – second row of the table, defines the bitmap used for that file. The bitmap will appear in the Agents list.
parsing command	In Wizard, a high-level scripting interface that allows manipulation of data. Parsing is the process of breaking an event down into its components.
Port	In Wizard, ports enable an Agent to locate the security event data on the network by providing the IP address and other information about the source (security device [router, IDS, switch, etc…]). Each row in the Port Configuration table runs one agent script to one event source.

Query Manager Process (query_manager)	The query manager (query_manager) receives quick query and drill down requests from Sentinel Control Center and forwards them to the database through DAS. The requests from Sentinel Control Center define the events needed via a criteria or a filter. If a filter is used, the Query Manager retrieves the filter definition and converts the filter to an xml criterion. Query Manager then sends the request to the database. Not all filters can be completely converted to xml. If the filter is fully converted, the Query Manager instructs DAS to send the reply directly to the Sentinel Control Center. If the filter contains regular expressions that cannot be converted to xml the query manager converts what it can and generates a conservative xml criterion that returns a superset of the required events. In that case, Query Manager instructs DAS to return the result to the Query Manager. When the reply comes back to the Query Manager it filters it in memory and sends those events that pass the filter to the Sentinel Control Center.
Quick Query	See Query Manager.
Rx Buffer	Part of Agent Manager, default size is 50,000 events. The receive buffer is an editable parameter. Minimum size is 5,000.
Rx Buffer Pointer	The Receive Buffer pointer points to data bytes in the Receive Buffer. Prior to each evaluated decide string, the Receive Buffer pointer is reset to its held value (normally zero).
RuleLg Checker Process (rulelg_checker)	The RuleLg Checker (rulelg_checker) process validates filter and correlation rule expressions. The Sentinel Control Center uses these results to determine if a filter or a correlation rule can be saved.
script file	In Wizard, a compiled file (*.asd) that is made up of the agent template file, parameter file, lookup file and mapping file.

Sentinel Control Center	Sentinel Control Center is the central management console to view summary displays, historical reports, filter real-time events and create incidents. Sentinel Control Center provides real-time display of events, system overview of changes in activity triggered by settings set in agents, administration of filters, reporting, correlated rules and global filters and security event management through incidents.
Sentinel Server	Sentinel Server receives normalized event information collected by Agents from Wizard Agent Manager. The Sentinel Server correlates these events to find patterns and identify threats and reports on real-time data and historical information that can be viewed on the Sentinel Control Center.
Sequences (startup and backout)	Startup and backout sequences are assigned to a port, which executes the series of scripts that it contains when it is started or stopped. A script must be included in a startup or backout sequence in order to be used by a port. Ports enable an agent to locate Wizard hosts on the network by providing the IP address or file name about the host. They also provide Sentinel with information on the location of sensors and the agent that is used to manage data for those sensors. The following options are configurable for ports: ▪ Connection type ▪ Process name ▪ Socket information ▪ SNMP information ▪ Input/output file names ▪ Agent name
Startup Sequences	See Sequences.
Statistics Engine	See Event Counts Over Time Process.

System Events	Internal or System Events is a means to report on the status and status change of the system. There are two types of events generated by the system, they are: ▪ Internal events ▪ Performance events Internal events are informational and describe a single state or change of state in the system. They report when a user logs in or fails to authenticate, when a process is started or a correlated rule is activated. Performance events are generated on a periodic basis and describe average resources used by different parts of the system.
Template Files	For Agents, you can create, add states to, edit and delete templates. Templates determine how records will be processed. Most of the decisions about templates revolve around what types of records you are working with and their format. There is an equivalent template file with a .tem extension. Template files are based on states. A state is a decision point within the logical flow or path of a template. Each point (state) contains information indicating the next processing to perform. States include parameters when the template is merged with a parameter file, specific values replace the parameters. When the parameters are replaced by specific values, one or more script files are created. As a state is inserted into a template, it is assigned a number that remains with it no matter where it is moved in the template.
Vulnerability Visualization	A graphical representation of real-time event data against vulnerable systems and is available on an event for current and event time vulnerability.
Watchdog Process	Watchdog is a Sentinel Process that manages all other Sentinel Processes. If a process other than Watchdog stops, Watchdog will restart that process.
Watchlist Correlation Rule	Allows you to specify a text string that the Correlation Engine will watch for in every meta-tag for every incoming event. For example, a Watchlist rule can look for a specific source IP address of a hacker and notify you anytime that IP address is seen in any event message.

Wizard	The Agent Builder and Agent Manager.
Wizard host	Any machine that has the Agent Manager software installed.
workflow	See iTRAC™.
workflow_container.xml	Specifies the configuration for the workflow (iTRAC) service.

activity_container.xml........................ 9-1
advanced correlation
  definition....................................... 7-6
advanced correlation rule
  creating 7-13
agent builder 4-1
agent engine 4-1
Agent Manager................................. 4-1
ALERT.. 3-3
APPEND 3-4
basic correlation
  definition....................................... 7-5
basic correlation rule
  creating 7-9
  which events should be excluded
    from the pattern match 7-4
  which events should be included in
    the pattern match 7-3
BITFIELD 3-7
BREAKPOINT 3-9
BYTEFIELD...................................... 3-9
CLEAR 3-11
CLEARTAGS 3-13
COMMENT..................................... 3-13
COMPARE 3-14
ConnectionManager.......................... 9-2
CONSTANTTAGS........................... 3-15
CONVERT...................................... 3-16
COPY 3-17
COPY-FROM-RX-BUFF 3-17
COPY-FROM-RX-BUFF-UNTIL-
  SEARCH 3-17
COPY-FROM-STRING-TO-STRING-
  UNTIL-SEARCH......................... 3-17
COPY-STRING-TO-STRING 3-17
correlation
  output 7-43
  output structure 7-43
  script parameters 7-43
correlation command line.................. 8-1
  affinityOneProcessor..................... 8-2
  configurationFile............................ 8-2
  dbRetries....................................... 8-2
  dbTimeout 8-2
  debug 8-1
  help 8-2
  inputChannel 8-1
  logFile... 8-2
  logPeriod 8-2
  mgmtInputChannel 8-2
  mgmtOutputChannel......................8-2
  mgmtService.................................8-2
  name...8-2
  noStartupRules 8-2
  outputChannel 8-1
  outputExecuteChannel 8-1
  outputUpdateChannel....................8-1
  ruleFile..8-1
  service 8-2
  useEventTime................................8-2
  useNullOutput 8-2
  version 8-2
  xmlruleFile 8-1
correlation rule example
  brute force – same source and target
   7-28
  buffer overflow - same source to
    same target.............................7-28
  buffer overflow – service stoppage.....
   7-24
  denial of service...........................7-24
  login failures - any source to any
    destination 7-27
  login failures - same source to same
    destination 7-27
  Microsoft - anonymous login........7-30
  Microsoft - general windows
    authentication 7-31
  Microsoft - IE...............................7-31
  Microsoft – IIS.............................7-29
  Microsoft - LAN Manager
    Authentication..........................7-30
  Microsoft - MDAC 7-29
  Microsoft - NETBIOS 7-30
  Microsoft - remote registry access......
   7-31
  Microsoft - Windows scripting 7-32
  Microsoft– SQL Server 7-29
  multiple backdoor – different sources
   7-26
  multiple backdoor – single source7-26
  trojan horse..................................7-26
  UNIX - Apache Web server 7-32
  UNIX - BIND/DNS........................7-35
  UNIX - FTP 7-33
  UNIX - general UNIX 7-35
  UNIX - line printer daemon 7-34
  UNIX - remote procedure call 7-32

  UNIX - remote services 7-34
  UNIX - secure shell 7-33
  UNIX - sendmail 7-34
  UNIX – SNMP 7-33
  virus outbreak............................. 7-25
  worm outbreak 7-25
CRC 3-19
das_binary.xml 9-1
  reconfiguring 9-2
das_query.xml 9-1
  reconfiguring 9-2
das_rt.xml... 9-1
data type
  array (variable arrays) 2-6
  derived aggregate 2-6
  fvar (float variable) 2-6
  ivar (integer variable) 2-5
  number 2-5
  quoted data 2-6
  svar (string variable)...................... 2-6
DATE... 3-20
DATETIME 3-21
DBCLOSE 3-22
dbconfig.. 9-3
DBDELETE 3-22
DBGETROW 3-23
DBINSERT 3-23
DBOPEN 3-24
DBSELECT 3-25
DEC.. 3-26
decide strings
  format 2-1
  hierarchy 2-2
  parameter names 2-2
  receive buffer 2-1
  receive buffer pointer rules............ 2-2
DECODE.. 3-27
default user
  ESEC_CORR 6-1
  esecadm....................................... 6-1
  esecapp.. 6-1
  esecdba.. 6-1
  esecrpt 6-1
default user password
  esecadm..................................... 10-1
  esecapp...................................... 10-1
  esecdba...................................... 10-2
  esecrpt 10-2
  e-Security Administrator 10-3
  e-Security Application DB
    Administrator 10-4
  e-Security DB Administrator 10-3
  e-Security Report user.................10-5
DELETE...3-27
DispatchManager 9-2
DISPLAY 3-28
ELSE 3-29
ENCODE 3-30
ENDFOR 3-30
ENDIF..3-31
ENDWHILE....................................3-31
esecadm
  changing password......................10-1
esecapp
  changing password......................10-1
esecdba
  changing password......................10-2
esecrpt
  changing password......................10-2
e-Security
  information....................................1-2
  technical support............................1-2
  website...1-2
e-Security Administrator
  changing password......................10-3
e-Security Application DB Administrator
  changing password......................10-4
e-Security DB Administrator
  changing password......................10-3
e-Security Report user
  changing password......................10-5
ESNMPGET 3-31
ESNMPSET....................................3-32
EVENT...3-33
Event Reserved Variable
  i_Severity....................................3-34
  s_BM...3-34
  s_CRIT..3-34
  s_CT1 3-34
  s_CT2 3-34
  s_CT3 3-34
  s_CV1 – s_CV100 3-34
  s_DHN 3-34
  s_DIP 3-34
  s_DP 3-34
  s_DUN 3-34
  s_EI...3-34
  s_ET 3-34
  s_EVT 3-34
  s_FN 3-34
  s_P...3-34
  s_PN 3-34

  s_Res 3-34
  s_RN 3-34
  s_RT1.. 3-34
  s_RT2.. 3-34
  s_RT3.. 3-34
  s_RV1 – s_RV100...................... 3-35
  s_SHN 3-34
  s_SIP... 3-34
  s_SN 3-34
  s_SP.. 3-34
  s_ST 3-34
  s_SubRes.................................. 3-34
  s_SUN 3-34
  s_VULN 3-34
EXPORTVAR 3-36
FILEA 3-37
FILEL.. 3-38
FILER 3-38
FILEW 3-39
filter operation operator
  flow 7-22
  intersection 7-22
  union 7-22
FOR... 3-40
free form RuleLg correlation
  definition 7-6
Free Form RuleLg Correlation Rule
  creating 7-17
GETCONFIG 3-41
GETENV 3-42
HEXTONUM.................................. 3-42
IF 3-43
INC 3-45
INDICATOR 3-45
INFO_CLEARTAGS 3-46
INFO_CLOSE 3-46
INFO_CONSTANTTAGS 3-47
INFO_CREATE 3-47
INFO_DUMP 3-48
INFO_PUSH................................. 3-48
INFO_SEND................................. 3-48
INFO_SETTAG 3-49
IPTONUM..................................... 3-53
LENGTH....................................... 3-54
LENGTH-OPTION2....................... 3-54
LOOKUP 3-54
meta-tag
  CorrelatedEventUuids 5-1
  Criticality....................................... 5-1
  Ct* 5-2
  CustomerVar* 5-2
  DataContect.........................3-35, 5-4
  DateTime 5-2
  DestinationAssetCategory 5-6
  DestinationAssetId 5-7
  DestinationAssetMaintainer 5-7
  DestinationAssetName 5-6
  DestinationAssetOwner 5-7
  DestinationAssetValue..................5-6
  DestinationBuilding 5-6
  DestinationBusinessUnit 5-7
  DestinationCity..............................5-6
  DestinationCountry 5-7
  DestinationCriticality 5-6
  DestinationDepartment 5-7
  DestinationDivision 5-7
  DestinationEnvironmentIdentity 5-6
  DestinationFunction 3-35, 5-5
  DestinationHostName 5-3
  DestinationIP 5-3
  DestinationLineOfBusiness...........5-7
  DestinationMacAddress 5-6
  DestinationNetworkIdentity 5-6
  DestinationOperationalContext.......... ....3-35, 5-5
  DestinationPort 5-3
  DestinationRackNumber 5-6
  DestinationRoom 5-6
  DestinationSensitivity....................5-6
  DestinationState 5-7
  DestinationThreatLevel.........3-35, 5-5
  DestinationUserContext........3-35, 5-5
  DestinationUserName...................5-3
  DestinationZipCode 5-7
  DeviceCategory 3-35, 5-4
  DeviceName 5-4
  eSecTaxonomyLevel1 ....3-35, 5-5
  eSecTaxonomyLevel2 ....3-35, 5-5
  eSecTaxonomyLevel3 ....3-35, 5-5
  eSecTaxonomyLevel4 ....3-35, 5-5
  EventContext 3-35, 5-4
  EventID 5-3
  EventName 5-3
  EventTime.....................................5-3
  ExtendedInformation......................5-3
  File Name (FN) 5-3
  Message 5-3
  MSSPCustomerName 3-35, 5-4
  NormalizedAttackName 5-4
  ProductName................................5-4

Protocol (Prot) 5-3
ReporterName.............................. 5-4
ReservedVar1-10.......................... 5-4
ReservedVar11-20........................ 5-4
ReservedVar21-25........................ 5-4
ReservedVar26-29........................ 5-4
ReservedVar40-43........................ 5-5
ReservedVar49 .... 3-35, 5-5
ReservedVar54-100...................... 5-7
ReservedVar54-55........................ 5-5
Resource...................................... 5-7
Rt1.. 5-7
Rt2.. 5-7
Rt3.. 5-7
SensorName 5-8
SensorType.................................. 5-8
Severity 5-8
SourceAssetCategory 5-5
SourceAssetID 5-6
SourceAssetMaintainer 5-6
SourceAssetName 5-5
SourceAssetOwner 5-6
SourceAssetValue........................ 5-5
SourceBuilding 5-5
SourceBusinessUnit..................... 5-6
SourceCity.................................... 5-5
SourceCountry 5-6
SourceCriticality 5-5
SourceDepartment........................ 5-6
SourceDivision 5-6
SourceEnvironmentIdentity........... 5-5
SourceFunction 3-35, 5-4
SourceHostName.......................... 5-7
SourceID 5-7
SourceIP 5-7
SourceLineOfBusiness................. 5-6
SourceMacAddress....................... 5-5
SourceNetworkIdentity.................. 5-5
SourceOperationalContext... 3-35, 5-4
SourcePort 5-8
SourceRackNumber...................... 5-5
SourceRoom 5-5
SourceSensitivity.......................... 5-5
SourceState 5-6
SourceThreatLevel............... 3-35, 5-4
SourceUserContext.............. 3-35, 5-4
SourceUserName.......................... 5-8
SourceZipCode 5-6
SubResource 5-8
VirusStatus........................... 3-35, 5-5
Vulnerability.................................. 5-8
  WizardAgent 5-8
  WizardPort 5-8
NEGSEARCH 3-56
NUMTOHEX 3-57
NUMTOIP 3-57
PARSER_ATTACHVARIABLE........3-58
PARSER_CREATEBASIC 3-59
PARSER_NEXT 3-60
PARSER_PARSESTRING 3-61
parsing
  command format...........................3-3
parsing command
  ALERT 3-3
  APPEND 3-4
  BITFIELD 3-7
  BREAKPOINT................................3-9
  BYTEFIELD 3-9
  CLEAR 3-11
  CLEARTAGS 3-13
  COMMENT 3-13
  COMPARE...................................3-14
  CONSTANTTAGS 3-15
  CONVERT 3-16
  COPY...3-17
  COPY-FROM-RX-BUFF 3-17
  COPY-FROM-RX-BUFF-UNTIL-SEARCH 3-17
  COPY-FROM-STRING-TO-STRING-UNTIL-SEARCH 3-17
  COPY-STRING-TO-STRING.......3-17
  CRC 3-19
  database interaction function.........3-1
  DATE 3-20
  DATETIME...................................3-21
  DBCLOSE....................................3-22
  DBDELETE..................................3-22
  DBGETROW................................3-23
  DBINSERT...................................3-23
  DBOPEN......................................3-24
  DBSELECT..................................3-25
  debugging function 3-1
  DEC 3-26
  DECODE 3-27
  DELETE.......................................3-27
  DISPLAY.....................................3-28
  ELSE..3-29
  ENCODE 3-30
  ENDFOR......................................3-30
  ENDIF 3-31
  ENDWHILE..................................3-31
  ESNMPGET.................................3-31

ESNMPSET 3-32
EVENT 3-33
EXPORTVAR 3-36
file interaction function 3-1
FILEA 3-37
FILEL.. 3-38
FILER 3-38
FILEW 3-39
FOR.. 3-40
GETCONFIG 3-41
GETENV 3-42
HEXTONUM................................ 3-42
IF 3-43
INC 3-45
INDICATOR 3-45
INFO_CLEARTAGS.................... 3-46
INFO_CLOSE 3-46
INFO_CONSTANTTAGS 3-47
INFO_CREATE 3-47
INFO_DUMP 3-48
INFO_PUSH................................ 3-48
INFO_SEND................................ 3-48
INFO_SETTAG 3-49
IPTONUM................................... 3-53
LENGTH..................................... 3-54
LENGTH-OPTION2..................... 3-54
logical operation function 3-1
LOOKUP 3-54
NEGSEARCH 3-56
network interaction function 3-1
notification function 3-1
NUMTOHEX............................... 3-57
NUMTOIP................................... 3-57
PARSER_ATTACHVARIABLE ... 3-58
PARSER_CREATEBASIC 3-59
PARSER_NEXT.......................... 3-60
PARSER_PARSESTRING.......... 3-61
PAUSE 3-61
POPUP....................................... 3-62
PRINTF 3-62
raw data manipulation function 3-1
REGEXPREPALCE 3-65
REGEXPSEARCH 3-66
REGEXPSEARCH_EXPLICIT 3-66
REGEXPSEARCH_STRING....... 3-66
REPLACE 3-69
RESET 3-70
RXBUFF..................................... 3-70
SEARCH 3-71
SET 3-72
  SETBYTES 3-73
  SETCONFIG................................3-73
  SHELL 3-74
  SKIP..3-75
  SKIPWORD 3-77
  SNMPGET...................................3-78
  SNMPSET 3-79
  SOCKETW...................................3-80
  STONUM 3-81
  string manipulation function 3-2
  STRIP 3-82
  STRIP-ASCII-RANGE..................3-82
  TBOSSETCOMMAND 3-83
  TBOSSETREQUEST...................3-85
  TIME 3-87
  TOKENSIZE 3-87
  TOLOWER...................................3-89
  TOUPPER 3-89
  TRANSLATE................................3-90
  TRIM 3-92
  utility function 3-2
  variable handling function 3-2
  Vulnerability Scanning 3-2
  WHILE 3-92
parsing command formats 3-3
parsing commands 2-5
  format..3-3
  using arrays 3-3
PAUSE 3-61
POPUP 3-62
popup.cfg 4-2
popup.exe 4-1
PRINTF..3-62
receive buffer 2-1
REGEXPREPLACE 3-65
REGEXPSEARCH...........................3-66
REGEXPSEARCH_EXPLICIT 3-66
REGEXPSEARCH_STRING 3-66
regular expressions 2-3
  special characters 2-4
REPLACE 3-69
RESET...3-70
RuleLg operator
  and..7-18
  not...7-18
  or 7-18
RXBUFF 3-70
script parameters 7-43
  %agent% 7-44
  %all%...7-46

  %CorrelatedEventID% 7-44
  %crt% 7-44
  %ct1% 7-45
  %ct2% 7-45
  %ct3% 7-45
  %cv1% - %cv100% 7-45
  %dhn% 7-45
  %dip% 7-44
  %dp% 7-45
  %dt% 7-44
  %dun% 7-45
  %ei% 7-45
  %et% 7-45
  %evt% 7-44
  %fn% 7-45
  %id% 7-44
  %msg% 7-45
  %pn% 7-45
  %port% 7-44
  %prot% 7-45
  %res% 7-44
  %rn% 7-45
  %rt1% 7-45
  %rt2% 7-45
  %rt3% 7-45
  %RuleCount% 7-44
  %RuleDescription% 7-43
  %RuleDuration% 7-44
  %RuleLg% 7-44
  %RuleName% 7-43
  %RulePattern% 7-44
  %RuleResource% 7-43
  %RuleSeverity% 7-43
  %RuleSubResource% 7-44
  %RuleType% 7-43
  %rv1% - %rv100% 7-45
  %sev% 7-44
  %shn% 7-45
  %sip% 7-44
  %sn% 7-44
  %sp% 7-45
  %src% 7-44
  %sres% 7-44
  %st% 7-44
  %sun% 7-45
  %vul% 7-44
SEARCH 3-71
SET 3-72
SETBYTES 3-73
SETCONFIG 3-73
SHELL 3-74
SKIP 3-75
SKIPWORD 3-77
SNMPGET 3-78
SNMPSET 3-79
SOCKETW 3-80
STONUM 3-81
STRIP 3-82
STRIP-ASCII-RANGE 3-82
TBOSSETCOMMAND 3-83
TBOSSETREQUEST 3-85
technical support 1-2
TIME 3-87
TOKENSIZE 3-87
TOLOWER 3-89
TOUPPER 3-89
TRANSLATE 3-90
trigger operation operator
  flow 7-22
  intersection 7-22
  union 7-22
TRIM 3-92
user permission
  Active Views 6-2
  administration 6-5
  advisor 6-5
  agent management........................6-4
  analysis...6-5
  correlation 6-5
  DAS statistics................................6-6
  event file information......................6-6
  general..6-1
  global filters...................................6-6
  incidents..6-4
  integration actions..........................6-2
  iTRAC 6-4
  iTRAC role management 6-7
  menu configuration 6-6
  menu items 6-2
  private filter 6-2
  process management 6-4
  public filter......................................6-2
  summary display............................6-3
  template management 6-4
  user management..........................6-6
  user session management............6-7
users
  default .... *See* default user
variables
  special rules..................................2-7
watchlist
  definition 7-5

watchlist rule
  creating 7-6
WHILE.. 3-92
window operation operator
  flow... 7-22
  intersection.................................. 7-22
  union 7-22
Wizard
  directory structure 4-2
Wizard directory structure..................4-2
Wizard utilities
  agent builder..................................4-1
  agent engine..................................4-1
  agentmanager................................4-1
  popup.cfg.......................................4-2
  popup.exe......................................4-1
workflow_container.xml 9-1