Novell® Sentinel™

6.0.1

www.novell.com

Volume IV – SENTINEL REFERENCE GUIDE

October 5, 2007

Novell®

Legal Notices

Novell, Inc. makes no representations or warranties with respect to the contents or use of this documentation, and specifically disclaims any express or implied warranties of merchantability or fitness for any particular purpose. Further, Novell, Inc. reserves the right to any and all parts of Novell software, to revise this publication and to make changes to its content, at any time, without obligation to notify any person or entity of such revisions or changes.

Any products or technical information provided under this Agreement may be subject to U.S. export controls and the trade laws of other countries. You agree to comply with all export control regulations and to obtain any required licenses or classification to export, re-export, or import deliverables. You agree not to export or re-export to entities on the current U.S. export exclusion lists or to any embargoed or terrorist countries as specified in the U.S. export laws. You agree to not use deliverables for prohibited nuclear, missile, or chemical biological weaponry end uses. Please refer to www.novell.com/info/exports/ for more information on exporting Novell software. Novell assumes no responsibility for your failure to obtain any necessary export approvals.

Novell, Inc. has intellectual property rights relating to technology embodied in the product that is described in this document. In particular, and without limitation, these intellectual property rights may include one or more of the U.S. patents listed at http://www.novell.com/company/legal/patents/ and one or more additional patents or pending patent applications in the U.S. and in other countries.

Novell, Inc.
404 Wyman Street, Suite 500
Waltham, MA 02451
U.S.A.
www.novell.com

Online Documentation: To access the online documentation for this and other Novell products and to get updates, see www.novell.com/documentation.

Novell Trademarks

For Novell trademarks, see the Novell Trademark and Service Mark list (http://www.novell.com/company/legal/trademarks/tmlist.html).

Third-Party Materials

All third-party trademarks are the property of their respective owners.

Third Party Legal Notices

This product may include the following open source programs that are available under the LGPL license. The text for this license can be found in the Licenses directory.

- edtFTPj-1.2.3 is licensed under the Lesser GNU Public License. For more information, disclaimers and restrictions see http://www.enterprisedt.com/products/edtftpj/purchase.html.
- Esper. Copyright © 2005-2006, Codehaus.
- jTDS-1.2.jar 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.
- Enhydra Shark, licensed under the Lesser General Public License available at: http://shark.objectweb.org/license.html.
- Tagish 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.

This product may include software developed by The Apache Software Foundation (http://www.apache.org/) and licensed under the Apache License, Version 2.0 (the "License"); the text for this license can be found in the Licenses directory or at http://www.apache.org/licenses/LICENSE-2.0. Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

The applicable open source programs are listed below.

- Apache Axis and Apache Tomcat, Copyright © 1999 to 2005, Apache Software Foundation. For more information, disclaimers and restrictions, see http://www.apache.org/licenses/.
- Apache Lucene, Copyright © 1999 to 2005, Apache Software Foundation. For more information, disclaimers and restrictions, see http://www.apache.org/licenses/.
- Bean Scripting Framework (BSF), licensed by the Apache Software Foundation Copyright © 1999-2004. For more information, disclaimers and restrictions see http://xml.apache.org/dist/LICENSE.txt.
- 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/.
- 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.

This product may include the following open source programs that are available under the Java license.

- 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.
- Java 2 Platform, Standard Edition. Copyright © Sun Microsystems, Inc. For more information, disclaimers and restrictions see http://java.sun.com/j2se/1.5.0/docs/relnotes/SMICopyright.html.
- 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.

This product may also include the following open source programs.

- ANTLR. For more information, disclaimers and restrictions, see http://www.antlr.org.
- Boost. Copyright © 1999, Boost.org.
- Concurrent, utility package. Copyright © Doug Lea. Used without CopyOnWriteArrayList and ConcurrentReaderHashMap classes.
- Java Ace, by Douglas C. Schmidt and his research group at Washington 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 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.
- JLDAP. Copyright 1998-2005 The OpenLDAP Foundation. All rights reserved. Portions Copyright © 1999 - 2003 Novell, Inc. All Rights Reserved.
- OpenSSL, by the OpenSSL Project. Copyright © 1998-2004. For more information, disclaimers and restrictions, see http://www.openssl.org.
- Rhino. Usage is subject to Mozilla Public License 1.1. For more information, see http://www.mozilla.org/rhino/.
- 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.
- Tinyxml. For more information, disclaimers and restrictions see http://grinninglizard.com/tinyxmldocs/index.html.

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 web pages are inactive, please contact Novell, Inc., 404 Wyman Street, Suite 500, Waltham, MA 02451 U.S.A.

Preface

The Sentinel 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 Sentinel's Enterprise Security Management System. There is additional documentation available on the Novell web portal (http://www.novell.com/documentation/).

Sentinel Technical documentation is broken down into six different volumes. They are:

- Volume I – Sentinel Install Guide
- Volume II – Sentinel User Guide
- Volume III – Sentinel Collector Builder User Guide
- Volume IV – Sentinel User Reference Guide
- Volume V – Sentinel 3rd Party Integration
- Volume VI – Sentinel Patch Installation Guide

Volume I – Sentinel Install Guide

This guide explains how to install:

- Sentinel Server
- Sentinel Console
- Sentinel Correlation Engine
- Sentinel Crystal Reports
- Collector Builder
- Collector Manager
- Advisor

Volume II – Sentinel User 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
- Collector Host Management
- Incidents
- Cases
- User management
- Workflow

Volume III – Collector Builder User Guide

This guide discusses:

- Collector Builder Operation
- Collector Manager
- Collectors
- Collector Host Management
- Building and maintaining Collectors

Volume IV - Sentinel User Reference Guide

This guide discusses:

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

Volume V - Sentinel 3rd Party Integration Guide

- Remedy
- HP OpenView Operations
- HP Service Desk

Volume VI - Sentinel Patch Installation Guide

- Patching from Sentinel 4.x to 6.0
- Patching from Sentinel 5.1.3 to 6.0

Feedback

We want to hear your comments and suggestions about this manual and the other documentation included with this product. Please use the User Comments feature at the bottom of each page of the online documentation and enter your comments there.

Additional Documentation

The other manuals on this product are available at http://www.novell.com/documentation. The additional documentation available on Sentinel:

- Sentinel 6.0 Installation Guide
- Sentinel 6.0 Patch Installation Guide
- Sentinel 6.0 User Guide

Documentation Conventions

Notes and Cautions

NOTE: Notes provide additional information that may be useful.

WARNING:

Warning provides additional information that may keep you away from performing actions that may cause 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
```

References:

- For more information, see "Section Name" (if in the same Chapter).
- For more information, see Chapter number, "Chapter Name" (if in the same Guide).
  - For more information, see Section Name in Chapter Name, *Guide Name* (if in a different Guide).

Other References

The following manuals are available with the Sentinel install CDs.

- Sentinel User Guide
- Sentinel Collector Builder User Guide
- Sentinel User Reference Guide
- Sentinel 3rd Party Integration Guide
- Release Notes

Contacting Novell

- Website: http://www.novell.com
- Novell Technical Support: http://support.novell.com/phone.html?sourceidint=suplnav4_phonesup
- Self Support: http://support.novell.com/support_options.html?sourceidint=suplnav_supportprog
- Patch Download Site: http://download.novell.com/index.jsp
- 24x7 support: http://www.novell.com/company/contact.html.
- For Collectors/Connectors/Reports/Correlation/Hotfixes/TIDS: http://support.novell.com/products/sentinel.

Contents

1 Sentinel™ User Reference Introduction

The Sentinel User Reference Guide is your reference for:

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

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

This guide discusses about the

- Scripting Language used to develop Collectors
- Parsing Commands
- Sentinel Meta tags
- Sentinel User Permissions
- Correlation Engine RuleLG Language
- Sentinel Data Access Service
- Sentinel Accounts and Password Changes
- Sentinel Database Views for Oracle
- Sentinel Database Views for Microsoft SQL Server

2 Collector Scripting Language

This chapter and the following chapter discuss how to use the Collector scripting language to build scripts. The operators in the various strings and parsing commands that are used in Collector building are covered. "Decide Strings", "Regular Expressions" and "Parsing Commands" are discussed in this chapter.

NOTE: Collectors and Collector Managers will only run on English Operating Systems. For Collectors to operate on non-English Operating Systems, they will need to be modified. Novell cannot guarantee proper operation of a Collector or Collector Manager on a non-English Operating System.

Decide Strings

Strings are case-sensitive.

As Collectors 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 String 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 deployed Collector 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 (such as end-of-line characters) to search for in the receive buffer. The search is performed byte-for-byte from the Receive Buffer pointer position forward.

NOTE: Because the Decide String box is cut off at the last printable character, the hex equivalent of a space must be used. The ":" Logical 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 (1).

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
\0A\&\02\	1 & 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 escape 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)
<form-feed>	<U000C>	FORM FEED (FF)

Symbolic Name	UCS	Description
<space>	<U0020>	SPACE

Parsing Commands

The Collector parsing language is function-oriented. Most of the parsing functions enable you to manipulate Collector variables and their contents. The Collector 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 deployed Collector and are not shared globally across all deployed Collector. 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 event source through its Connector (such as 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 Collector parsing language also enables you to copy or append data from string variables into other string variables.

Simple Data Types

number

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

```
SKIP(-1)
SKIPWORD(+3)
SET(f_total=f_total+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:

```
SET(i_bits[5]=1)
COPY(s_resources[3]:”FinanceServer”)
```

Quoted Data

Quoted data is scanned and parsed as follows:

- /=Escape 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 commands
COPY(s: "test")
SET(i_counter=i_counter+1)
*/
```

3 Collector Parsing Commands

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

Function	Parsing Command
Database Interaction **NOTE:** The database interaction commands are they are supported only for backward compatibility. Most database connections are now made through the JDBC connector.	"DBCLOSE" "DBDELETE" "DBGETROW" "DBINSERT" "DBOPEN" "DBSELECT"
Debugging	"BREAKPOINT"
File Interaction	"FILEA" "FILEL" "FILER" "FILEW"
Logical Operations	"COMPARE" "ELSE" "ENDFOR" "ENDIF" "ENDWHILE" "FOR" "IF" "LOOKUP" "WHILE"
Network Interaction	"SOCKETW"
Notification	"ALERT" "CLEARTAGS" "CONSTANTTAGS" "EVENT" "PAUSE"

Function	Parsing Command
Raw Data Manipulation	“BITFIELD” “BYTEFIELD” “CONVERT” “CRC” “DECODE” “DECODEMIME” “ENCODE” “ENCODEMIME” “HASH” “HEXTONUM” “NUMTOHEX” “SETBYTES” “STRIP” “STRIP-ASCII-RANGE” “UUID”
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” “DATETIMETOSECONDS” “PAUSE” “SHELL” “TBOSSETCOMMAND” “TBOSSETREQUEST” “TIME”

Function	Parsing Command
Variable Handling	“CLEAR” “DELETE” “GETCONFIG” “GETENV” “INC” “RESET” “RXBUFF” “SET” “SETCONFIG”
Vulnerability Scanning	“INFO_CLEARTAGS” “INFO_CLOSE” “INFO_CONSTANTTAGS” “INFO_CREATE” “INFO_DUMP” “INFO_PUSH” “INFO_SEND” “INFO_SETTAG”
Commands no longer viable in Sentinel 6.0	“DISPLAY” “INDICATOR” “POPUP” “DBCLOSE” “DBDELETE” “DBGETROW” “DBINSERT” “DBOPEN” “DBSELECT”

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, but this command has been replaced by the EVENT command. ALERT is included for backward compatibility only. Please see the documentation for Sentinel 5.1.3 for more information about this command.

The ALERT command does not get populated with several important new fields available in Sentinel 6. Collector scripts that still use the ALERT command should be updated to send these new fields:

- ConnectorID (RV23)
- EventSourceID (RV24)
- Trust Event Source Time (i_TrustDeviceTime)

Here is some sample code that could be added before calling the ALERT command. The exact code that should be used may vary from Collector to Collector. This code sample makes the following assumptions:

- s_MetaData is a string compiled in the Collector Script that includes the Base Message from the Event Source, the Source IP, and the Destination IP.
- RV23, RV24, and i_TrustDeviceTime have been populated by the Collector Script and simply need to be added to the Alert Message (s_AlertMsg)

```
PRINTF(s_NewFor60,"RV23='%s' RV24='%s'
i_TrustDeviceTime='%d'
",s_RV23,s_RV24,i_TrustDeviceTime)
APPEND(s_MetaData:s_NewFor60)
PRINTF(s_AlertMsg, "%s [%s]", s_BM, s_MetaData)
ALERT(s_ResSubRes, s_AlertMsg, i_Severity)
```

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 datax

- 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 Collector 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.

WARNING:

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 (For example, `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 Collector script debugger is running, the breakpoint command stops the parser pending user intervention. For example, from Collector Builder 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

WARNING:

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 event reserved and date/time reserved variables.

> **NOTE:** The CLEARTAGS command does not clear tags RV21-RV25 or variables that are protected by the CONSTANTTAGS command.

This command should be used at the beginning of every loop before parsing the device data and mapping it 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 Collector template file. This allows you to enter comments from the visual editor without switching to the text editor.

Format

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

For example:

```
/* COLLECTOR INFORMATION
; ------------------------------------------------
----
Collector_Name:                Standard Template
Collector_Description:         Template to base
new Collectors on
Collector_Manufacturer:        N/A
Collector_Product/Version:     N/A
Collector_Version:             release 4.1
Collector_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 Collector is processing. The s_PN variable should be declared constant and set once in the Collector setup state.

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

Argument	Type	Description
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 Collector 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 Collector 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"
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").

Argument	Type	Description
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: 02-19-1996).
svar_time	svar (OUTPUT)	The string variable in which the time will be stored (for example: 15:14:33).
day_of_week	svar	(Optional) The string variable in which the

Argument	Type	Description
	(OUTPUT) [OPTIONAL] ivar (OUTPUT) [OPTIONAL] Or fvar (OUTPUT) [OPTIONAL]	day of the week will be stored; written as the full Day name (for example: svar = Saturday) (Optional) 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)
```

The resulting date and time string variables have the time at the UTC timezone (timezone offset +0000).

Current Output Variables' Contents:

```
sdatevar = "01-01-70"
stimevar = "00:00:00"
sdate = "01-01-2000"
stime = "12:00:00"
sday = "Saturday"
```

DATETIMETOSECONDS

The DATETIMETOSECONDS command converts a date string and a time string to an integer representation of the number of seconds since January 1, 1970.

IMPORTANT:

The supported Date time format is MM-DD-YYYY HH:MM:SS. If the input does not follow this format, value "0" will be returned.

The valid date range is "January 1, 1970 00:00:00" to "January 18, 2038 11:59:59` including these values."

The input date and time string values are assumed to be the time at the UTC timezone (i.e., timezone offset +0000)

In the following example, the DATETIMETOSECONDS command gives you the number of seconds since January 1, 1970.

```
DATETIMETOSECONDS (i_timesecs, “01-01-2000”,
“12:00:00”)
```

Current Output Variables’ Contents:

```
i_timesecs = “946728000”
```

Format

```
DATETIMETOSECONDS(itime_secs, s_date, s_time)
```

Data Types

Argument	Type	Description
itime_secs	numvar (OUTPUT)	The integer number that will contain the number of seconds since 1970.
s_date	sring (INPUT)	The string variable of the date (for example: 02-19-1996).
s_time	string (INPUT)	The string variable of the time (for example: 15:14:33).

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 Collector only, DBOPEN will not work when the database name is configured to point to a "mapped drive". Since the Collector 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 Collector must be to a completely local database.

There are five required parameters.

- The first required parameter is the database type. This can be selected through 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"
```

DECODEMIME

The DECODEMIME command allows the user to decode a base-64 encoded string or string variable using base-64 decoding and store the resulting decoded string into a string variable. If there is an error the resulting data string would be

zero length and the optional number variable success is set to 0. If decoding is successful then the number variable success is set to 1.

Format

```
DECODEMIME(encoded_data, data, success)
```

Data Types

Argument	Type	Description
encoded_data	String/String Variable(INPUT)	Base-64 encoded string that needs to be decoded.
data	String Variable(OUTPUT)	Resultant decoded data.
success	Integer variable/Float Variable(OUTPUT) [OPTIONAL]	Set to one if decoding is successful, in case of an error it is set to zero.

For example:

```
DECODEMIME("VGVzdGluZyBEYXRhIEVuY29kaW5n",
s_data, i_success)
```

In the above example, DECODEMIME command decodes the string in double quotes using base-64 decoding and stores the resulting decoded string in s_data. S_data gets populated with following:

```
test encode64 command
```

Since decoding is successful, 1 gets assigned to the integer variable i_success.

Also see to the "ENCODEMIME" command.

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).

Argument	Type	Description
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 was deprecated in Sentinel 6.0. The debugger in the Sentinel Control Center provides similar functionality.

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 '!!'"
```

ENCODEMIME

The ENCODEMIME command allows the user to encode a string or string variable using base-64 encoding and store the resulting encoded string into a string variable.

Format

```
ENCODEMIME(data, encoded_data)
```

Data Types

Argument	Type	Description
data	String/string variable (INPUT)	Data string that needs to be encoded.
encoded_data	String variable (OUTPUT)	Resultant encoded data.

For Example:

```
COPY(s_data:"test encode64 command")
ENCODEMIME(s_data, s_encd_data)
```

In the above example ENCODEMIME command, encodes the string in s_data variable using base-64 encoding and stores the resulting encoded string in s_encd_data. S_encd_data gets populated with following:

```
VGVzdGluZyBEYXRhIEVuY29kaW5n
```

Also see to the "DECODEMIME" command.

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()
```

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 Collector Builder template. Only those variables that are used in the script and are not set to "" are sent. Any of the Standard Sentinel variable, Reserved variable or Custom variable can be sent. Variables like i_Severity and s_Res are required for an alert message to be processed by the Collector 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)

Variable	Short Description	Maps to meta-tag (label)
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	Device Attack Name (Reserved Sentinel 1)	Rt1 (rt1)
s_RT2	Reserved Sentinel 2	Rt2 (rt2)
s_RT3	Reserved Sentinel 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)
s_RV1 to s_RV29	Reserved Variable 1 to 29 **NOTE**: Reserved for Novell's use.	Rv1 to Rv29 (rv1 to rv29)
s_RV30	AttackId	Rv30
s_RV31	DeviceName	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)

Variable	Short Description	Maps to meta-tag (label)
s_RV40 to s_RV43	Reserved Value 40 to 43 **NOTE:** Reserved for Novell'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 Novell'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 Novell'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_Min
s_Month	s_Sec
s_Day	s_TZ
s_Hour	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()
```

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
- By default, the working directory is %ESEC_HOME%\data or $ESEC_HOME/data.
  - 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 a Collector, 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
- By default, the working directory is %ESEC_HOME%\data or $ESEC_HOME/data.
  - 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 a Collector, 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.
- By default, the working directory is %ESEC_HOME%\data or $ESEC_HOME/data.
  - 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 a Collector, 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
- By default, the working directory is %ESEC_HOME%\data or $ESEC_HOME/data.
  - 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 a Collector, 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

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()	Any valid parameter that can be passed to

Argument	Type	Description
	parameter	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	Description (Example)
System.OS.Family	Operating system family (Solaris, Windows)
System.OS.Name	Operating system name (Windows 2000)
System.OS.Version.Major	Operating system major version (5)
System.OS.Version.Minor	Operating system minor version (0)
System.Net.Hostname	Collector Manager server name (CollectorManager_LON1)
System.Net.IP_List	Collector Manager IP addresses, separated by a semicolon (172.163.3.45;172.45.2.1)
System.Agent_Dir	Path to parent directory holding Collector directories for all running collectors ($ESEC_HOME/data/collector_mgr.cache/collector_instances)
System.PortScript	Collector instance name and UUID (WMI_6_0_Collector_68714633-A987-1029-A520-000C29F2D765)
System.Local_Dir	Path to directory of the running Collector This is equivalent to the combination of System.Agent_Dir and System.PortScript
System.Data_Dir	Path to a directory that is protected during uninstallation. %ESEC_HOME%\data
FileConnector.InputFile	This option has been deprecated in Sentinel 6.0.
FileConnector.OutputFile	This option has been deprecated in Sentinel 6.0.

See also "SETCONFIG" command.

Format

```
GETCONFIG("Config_Option", Variable)
```

- Config_Option is the system property that you want to retrieve (FileConnector.InputFile) or FileConnector.OutputFile).
- Variable is the name of a string variable that will hold the retrieved value.

Data Types

Argument	Type	Description
Config Option	String (INPUT)	Name of the system property to retrieve (FileConnector.InputFile)
Variable	String (OUTPUT)	Variable to hold the retrieved value.

For example:

```
GETCONFIG(“System.OSFamily”, s_osfamilyname)
```

Current Output Variable’s Contents

```
S_osfamilyname = “Windows”
```

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)
```

HASH

The HASH command allows the user to perform a hash on a string or string variable. The user can specify what kind of hash (dss1, sha1, md2, md4, md5, ripemd) needs to be performed. In case an incorrect hash name is specified then Unsupported Algorithm is returned. The resulting hash value is stored in a string variable. An error message will be stored in the output string variable for unsupported algorithms.

Format

```
HASH(hash_algorithm, data, hash_data)
```

Data Types

Argument	Type	Description
hash_algorithm	String/String Variable(INPUT)	Type of hash that needs to be performed.
data	String/String Variable(INPUT)	Data on which hash needs to be performed.
hash_data	String	Resultant hash string.

Argument	Type	Description
	Variable(OUTPUT)	

For example:

```
COPY(s_data: "test hash data")
HASH("ripemd", s_data, s_ripemd_data)
```

In the above example, HASH command performs a ripemd hash on s_data and stores the resulting ripemd hashed data in s_ripemd_data. s_ripemd_data contains following hash value, as viewed in the Sentinel debugger:

```
"\d6a0d5e2d0a09dfba5\MH\10b7\V\fc\#\b9\f6\ff\"
```

Although the Sentinel debugger shows this string, the actual value is binary. To prevent storage problems, Novell recommends that the hash_data be converted to HEX using the CONVERT command before the data is inserted into the database.

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 (INPUT) [OPTIONAL]	The data to compare to data1. This must be the same type is data1.
…	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 was deprecated in Sentinel 6.0. The command is supported in Sentinel 6.0 for backward compatibility. The EVENT command provides similar functionality.

INFO_CLEARTAGS

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

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

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

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

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

Argument	Type	Description
OUT string-variable	string (OUTPUT)	string variable to refer to IN handle

INFO_PUSH

This will tag the current values of all tag names (through 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 Collectormanager. For INFO_CLOSE, the info blocks are sent to Collectormanager 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

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

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 Collector parameters.	String	X	
ProductName	Name of the scanner.	String	X	
ProductVersion	Version of the scanner	String	X	
ScannerType	The type of scanner.	String	X	
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		

Tag Name	Explanation	Type	Constant	Required
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		
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		

Tag Name	Explanation	Type	Constant	Required
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		

Asset Info Block Tags

The following are valid asset 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 an asset record in the database (* The info block needs to have at least either a HostName or an IpAddress or both to be stored as an asset record in the database, if neither HostName nor IpAddres exist, the recorded will be ignored.). If a tag is set that is not in the following list, the asset back end will ignore the tag.

Tag Name	Explanation	Type	Constant	Required
ScanStartDate		String		
CustomerId	The Id of the customer in CUST table	Number		X
AssetEntryType	Type of asset	String		X

Tag Name	Explanation	Type	Constant	Required
AssetCategory	Asset category	String		
EnvironmentIdentity	Environment Identity	String		
AssetValue	Asset value	String		
Criticality	Criticality	String		
Sensitivity	Sensitivity	String		
AssetName	Asset name	String		X Note:Required for soft asset
ProductName	Product name	String		
ProductVersion	Product version	String		
Vendor	Vendor name	String		
OwnerFirstName	Asset owner first name	String		
OwnerLastName	Asset owner last name	String		
OwnerPhoneNumber	Asset owner phone number	String		
OwnerEmail	Asset owner email address	String		
MaintainerFirstName	Asset maintainer first name	String		
MaintainerLastName	Asset maintainer last name	String		
MaintainerPhoneNumber	Asset maintainer phone number	String		
MaintainerEmail	Asset maintainer email address	String		
BusinessUnit	The Business Unit name the asset belongs to	String		
LineOfBusiness	The line of business name the asset belongs to	String		

Tag Name	Explanation	Type	Constant	Required
Division	The division name the asset belongs to	String		
Department	The department name the asset belongs to	String		
PersonnelSeq	The sequence number for the personnel	Number		
IpAddress	IP address	String		X* - either an IP address or a host name is required
HostName	Hostname	String		X* - either an IP address or a host name is required
MacAddress	Mac address	String		
RackNumber	Rack number	String		
Building	Building	String		
Room	Room	String		
AddressLine1	Address line 1	String		
AddressLine2	Address line 2	String		
City	City	String		
State	State	String		
Country	Country	String		
ZipCode	Zip Code	String		
NetworkIdentity	Network Identity	String		

INFO_* COMMAND EXAMPLES

Sentinel 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".

Example 1 (for vulnerability):

```
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 1:

```
<?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>
```

Example 2 (for assets):

```
INFO_CREATE(handle,"asset")
INFO_SETTAG(handle, "ScanStartDate", s_date)
INFO_SETTAG(handle, "CustomerId", i_customerid)
INFO_SETTAG(handle, "AssetEntryType",
s_entrytype)
INFO_SETTAG(handle, "IpAddress", s_ip)
INFO_SETTAG(handle, "AssetCategory", s_category)
COPY(s_date:"2004|Aug|03|09|08|03|-0500")
SET(i_customerid=1)
COPY(s_entrytype:"physical")
COPY(s_ip:"192.168.0.2")
COPY(s_category:"DESKTOP")
INFO_PUSH(handle)
INFO_DUMP(handle, s_assetinfo    )
INFO_SEND(handle)
INFO_CLOSE(handle)
```

Results 2:

```
<?xml version="1.0" encoding="UTF-8"?>
<infos id="3A4A1CD0B56E10299966000D56C732D7"
type="asset" segnum="0" version="4.2.0.0"
EOD="false">
<info AssetCategory="DESKTOP"
AssetEntryType="physical" CustomerId="1"
IpAddress="192.168.0.2"
ScanStartDate="2004|Aug|03|09|08|03|-
0500"/></info>
</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 Collector 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 <Collector>.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: Collector.par
column name	string (INPUT)	The column within the parameter file to use. Default: `Collector 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.

```
COLLECTOR 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.

Argument	Type	Description
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 was deprecated in Sentinel 6.0. The debugger in the Sentinel Control Center provides similar functionality.

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.

Argument	Type	Description
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 “-“.
%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 apply 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 Collector 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.
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.

Argument	Type	Description
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 198 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.

Argument	Type	Description
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	Description (Example)
System.OS.Family	Operating system family (Solaris, Windows)
System.OS.Name	Operating system name (Windows 2000)
System.OS.Version.Major	Operating system major version (5)
System.OS.Version.Minor	Operating system minor version (0)
System.Net.Hostname	Collector Manager server name (CollectorManager_LON1)
System.Net.IP_List	Collector Manager IP addresses, separated by a semicolon (172.163.3.45;172.45.2.1)
System.Agent_Dir	Path to parent directory holdng collector directories for all running collectors ($ESEC_HOME/data/collector_mgr.cache/collector_instances)
System.PortScript	Collector instance name and UUID (WMI_6_0_Collector_68714633-A987-1029-A520-000C29F2D765)
System.Local_Dir	Path to directory of the running collector This is equivalent to the combination of System.Agent_Dir and System.PortScript
System.Data_Dir	Path to a directory that is protected during uninstallation. %ESEC_HOME%\data
FileConnector.InputFile	This option has been deprecated in Sentinel 6.0.
FileConnector.OutputFile	This option has been deprecated in Sentinel 6.0.

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(“System.Net.Hostname”, s_CMhostname)
SETCONFIG(“System.Net.Hostname”,
“CollectorManager_LON1”)
```

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(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(4)
aaaaaa bbbbb c d ee
                ^
SKIPWORD(5)
aaaaaa bbbbb c d ee
                 ^
```

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).

WARNING:

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 Collector 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 7 6 5 4 3 2 1	1 1 0 0 1 1 0 0	Echo of Character: The remote command response is the echo of this byte back to the port.

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 7	0 0	Operation Code: 00 = Character Scan Request
6 5 4	MSB LSB	Display No.: 000 = No. 1 001 = No. 2 ... 111 = No. 3
3 2 1	MSB LSB	Type: 000 = No. 1 001 = No. 2 ... 111 = No. 8

Character 1 – Character Scan Response		
Bit Numbers(s)	**Value**	**Meaning**
8 7 6 5 4 3 2 1	MSB LSB	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).

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) numvar (OUTPUT)	The string representation of the time is placed in this string variable (for example: “23-11-55”). 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"
" "
"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.

Argument	Type	Description
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 a Collector's script directory. Collector Builder does not support edting of these files; therefore, Novell 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 "
```

UUID

The UUID command allows the user to assign UUIDs to a list of one or more string variables. Up to fifty variable names can be assigned UUID's in one UUID command.

Format

```
UUID(uuid_var1, uuid_var2, uuid_var3,
…uuid_var50)
```

Data Types

Argument	Type	Description
Uuid_var1	String variable(OUTPUT)	String variable that would be assigned a uuid.
Uuid_var2	String variable(OUTPUT) [OPTIONAL]	String variable that would be assigned a uuid.
Uuid_var3	String variable(OUTPUT) [OPTIONAL]	String variable that would be assigned a uuid.

For example:

```
UUID(s_uuid1, s_uuid2)
```

In the above example, UUID command assigns uuid's to following variables: s_uuid1, s_uuid2.

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 ^.

WARNING:

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).

Argument	Type	Description
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")
```

4 Sentinel Meta-tags

Meta-tags store meta-data. Meta-data is information about data and pre-defined variable names. For Example, the Source IP of an attack is mapped to SIP meta-tag and Product names are mapped to PN meta-tag. Data into meta-tags can be populated either from device log data or is set as part of the Collector processing.

For information on the Event Configuration and mapping feature in the Sentinel Control Center, see Admin tab documentation.

The value in the Collector Variable column is the name of the Collector variable to set in order to populate the corresponding Meta-tag. For more information about parsing commands, see Collector Parsing Commands and the documentation for specific Collectors.

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 (For example, - 6A5349DA-7CBF-1028-9795-000BCDFFF482)
- **date:** Collector 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 (that is – xxx.xxx.xxx.xxx)

NOTE: In the table below, Labels and Meta-tags are used in the Sentinel Control Center. Collector Variables are used in the Collector parsing language. Not all meta-tags have a corresponding Collector Variable.

Label	Meta-tag	Type	Description	Collector Variable
Severity	sev	integer	The normalized severity of the event (0-5).	i_Severity
Vulnerability	vul	integer	The vulnerability of the asset identified in this event.	s_VULN
Criticality	crt	integer	The criticality of the asset identified in this event.	s_CRIT
EventTime	dt	date	The normalized date and time of the event, as given by the collector.	
SourceIP	sip	IPv4	The source IP address from which the event originated.	s_SIP
DestinationIP	dip	IPv4	The destination IP address to which the event was targeted.	s_DIP
EventID	id	UUID	Unique identifier for this event.	

Label	Meta-tag	Type	Description	Collector Variable
SourceID	src	UUID	Unique identifier for the Sentinel service which generated this event.	
Collector	port	string	Name of the Collector that generated this event.	Not Applicable
CollectorScript	agent	string	The name of the Collector Script used by the Collector to generate this event.	Not Applicable
Resource	res	string	Compliance monitoring hierarchy level 1	s_RES
SubResource	sres	string	Compliance monitoring hierarchy level 2	s_SubRes
EventName	evt	string	The descriptive name of the event as reported (or given) by the sensor. Example Port Scan.	s_EVT
SensorName	sn	string	The name of the ultimate detector of the event when received in raw data. Example FW1 for a firewall.	s_SN
SensorType	st	string	The single character designator for the sensor type (N, H, O, V, C, A, I).	s_ST
DeviceEventTime	det	date	The normalized date and time of the event, as reported by the sensor.	
Protocol	prot	string	The network protocol of the event.	s_P
SourceHostName	shn	string	The source host name from which the event originated.	s_SHN
SourcePort	spint	integer	The source port from which the event originated.	s_SPINT
DestinationHostName	dhn	string	The destination host name to which the event was targeted.	s_DHN
DestinationPort	dpint	integer	The destination port to which the event was targeted.	s_DPINT
SourceUserName	sun	string	The source user name used to initiate an event. Example jdoe during an attempt to su.	s_SUN
DestinationUserName	dun	string	The destination user name on which an action was attempted. Example root during a password reset.	s_DUN
FileName	fn	string	The name of the program executed or the file accessed, modified or affected.	s_FN
ExtendedInformation	ei	string	Stores additional collector processed information. Values within this variable are separated by semi-colons ().	s_EI

Label	Meta-tag	Type	Description	Collector Variable
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
ProductName	pn	string	Indicates the type, vendor and product code name of the sensor from which the event was generated.	s_PN
Message	msg	string	Free-form message text for the event.	s_BM
DeviceAttackName	rt1	string	Device specific attack name that matches attack name known by Advisor. (String)	s_RT1
Rt2	rt2	string	Reserved by Novell for expansion. (String)	s_RT2
Ct1 thru Ct2	ct1 thru ct2	string	Reserved for use by customers for customer-specific data. (String)	s_CT1 and s_CT2
Rt3	rt3	integer	Reserved by Novell for expansion. (Number)	
Ct3	ct3	integer	Reserved for use by customers for customer-specific data. (Number)	s_CT3
CorrelatedEventUuids	ceu	string	List of event UUIDs associated with this correlated event. Only relevant for correlated events.	s_RT3
CustomerHierarchyId	rv1	integer	Customer Hierarchy Id	s_RV1
ReservedVar2 thru ReservedVar10	rv2 thru rv10	integer	Reserved by Novell for expansion. (Number)	s_RV2 thru s_RV10
ReservedVar11 thru ReservedVar20	rv11 thru rv20	date	Reserved by Novell for expansion. (Date)	s_RV11 thru s_RV20
CollectorManagerId	rv21	UUID	Unique identifier for the Collector Manager which generated this event.	s_RV21
CollectorId	rv22	UUID	Unique identifier for the Collector which generated this event.	s_RV22
ConnectorId	rv23	UUID	Unique identifier for the Connector which generated this event.	s_RV23
EventSourceId	rv24	UUID	Unique identifier for the Event Source which generated this event.	s_RV24
RawDataRecordId	rv25	UUID	Unique identifier for the Raw Data Record associated with this event.	s_RV25
ControlPack	rv26	string	Not currently in use	s_RV26

Label	Meta-tag	Type	Description	Collector Variable
ControlMonitor	rv27	string	Not currently in use	s_RV27
ReservedVar28	rv28	string	Reserved by Novell for expansion. (String)	s_RV28
SourceIPCountry	rv29	string	Country of source IP address.	s_RV29
AttackId	rv30	string	Normalized Attack Id. This is taken from Advisor data. (String)	s_RV30
DeviceName	rv31	string	The name of the device generating the event. If this device is supported by Advisor, the name should match the name known by Advisor. (String)	s_RV31
DeviceCategory	rv32	string	Device category (FW, IDS, AV, OS, DB).	s_RV32
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
VendorEventCode	rv40	string	Event code reported by device vendor. (String)	s_RV40
DestinationDomain	rv41	string	Destination Domain. (String)	s_RV41
SourceDomain	rv42	string	Source Domain. (String)	s_RV42
ReservedVar43	rv43	string	Reserved by Novell for expansion. (String)	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
CustomerHierarchyLevel1	rv49	string	Customer Hierarchy Level 1 (used by MSSPs)	s_RV49
eSecTaxonomyLevel1	rv50	string	Sentinel event code categorization - level 1.	s_RV50

Label	Meta-tag	Type	Description	Collector Variable
eSecTaxonomyLevel2	rv51	string	Sentinel event code categorization - level 2.	s_RV51
eSecTaxonomyLevel3	rv52	string	Sentinel event code categorization - level 3.	s_RV52
eSecTaxonomyLevel4	rv53	string	Sentinel event code categorization - level 4.	s_RV53
CustomerHierarchyLevel2	rv54	string	Customer Hierarchy Level 2 (used by MSSPs)	s_RV54
CustomerHierarchyLevel3	rv55	string	Customer Hierarchy Level 3 (used by MSSPs)	s_RV55
SourceAssetName	rv56	string	Source Asset Name. Part of source host asset data. (String)	s_RV56
SourceMacAddress	rv57	string	Source Mac Address. Part of source host asset data. (String)	s_RV57
SourceNetworkIdentity	rv58	string	Source Network Identity. Part of source host asset data. (String)	s_RV58
SourceAssetCategory	rv59	string	Source Asset Category. Part of source host asset data. (String)	s_RV59
SourceEnvironmentIdentity	rv60	string	Source Environment Identity. Part of source host asset data. (String)	s_RV60
SourceAssetValue	rv61	string	Source Asset Value. Part of source host asset data. (String)	s_RV61
SourceCriticality	rv62	string	Source Criticality. Part of source host asset data. (String)	s_RV62
SourceSensitivity	rv63	string	Source Sensitivity. Part of source host asset data. (String)	s_RV63
SourceBuilding	rv64	string	Source Building. Part of source host asset data. (String)	s_RV64
SourceRoom	rv65	string	Source Room. Part of source host asset data. (String)	s_RV65
SourceRackNumber	rv66	string	Source Rack Number. Part of source host asset data. (String)	s_RV66
SourceCity	rv67	string	Source City. Part of source host asset data. (String)	s_RV67
SourceState	rv68	string	Source State. Part of source host asset data. (String)	s_RV68
SourceCountry	rv69	string	Source Country. Part of source host asset data. (String)	s_RV69
SourceZipCode	rv70	string	Source Zip Code. Part of source host asset data. (String)	s_RV70
SourceAssetOwner	rv71	string	Source Asset Owner. Part of source host asset data. (String)	s_RV71
SourceAssetMaintainer	rv72	string	Source Asset Maintainer. Part of source host asset data. (String)	s_RV72
SourceBusinessUnit	rv73	string	Source Business Unit. Part of source host asset data. (String)	s_RV73

Label	Meta-tag	Type	Description	Collector Variable
SourceLineOfBusiness	rv74	string	Source Line Of Business. Part of source host asset data. (String)	s_RV74
SourceDivision	rv75	string	Source Division. Part of source host asset data. (String)	s_RV75
SourceDepartment	rv76	string	Source Department. Part of source host asset data. (String)	s_RV76
SourceAssetId	rv77	string	Source Asset Id. Part of source host asset data. (String)	s_RV77
DestinationAssetName	rv78	string	Destination Asset Name. Part of destination host asset data. (String)	s_RV78
DestinationMacAddress	rv79	string	Destination Mac Address. Part of destination host asset data. (String)	s_RV79
DestinationNetworkIdentity	rv80	string	Destination Network Identity. Part of destination host asset data. (String)	s_RV80
DestinationAssetCategory	rv81	string	Destination Asset Category. Part of destination host asset data. (String)	s_RV81
DestinationEnvironmentIdentity	rv82	string	Destination Environment Identity. Part of destination host asset data. (String)	s_RV82
DestinationAssetValue	rv83	string	Destination Asset Value. Part of destination host asset data. (String)	s_RV83
DestinationCriticality	rv84	string	Destination Criticality. Part of destination host asset data. (String)	s_RV84
DestinationSensitivity	rv85	string	Destination Sensitivity. Part of destination host asset data. (String)	s_RV85
DestinationBuilding	rv86	string	Destination Building. Part of destination host asset data. (String)	s_RV86
DestinationRoom	rv87	string	Destination Room. Part of destination host asset data. (String)	s_RV87
DestinationRackNumber	rv88	string	Destination Rack Number. Part of destination host asset data. (String)	s_RV88
DestinationCity	rv89	string	Destination City. Part of destination host asset data. (String)	s_RV89
DestinationState	rv90	string	Destination State. Part of destination host asset data. (String)	s_RV90

Label	Meta-tag	Type	Description	Collector Variable
DestinationCountry	rv91	string	Destination Country. Part of destination host asset data. (String)	s_RV91
DestinationZipCode	rv92	string	Destination Zip Code. Part of destination host asset data. (String)	s_RV92
DestinationAssetOwner	rv93	string	Destination Asset Owner. Part of destination host asset data. (String)	s_RV93
DestinationAssetMaintainer	rv94	string	Destination Asset Maintainer. Part of destination host asset data. (String)	s_RV94
DestinationBusinessUnit	rv95	string	Destination Business Unit. Part of destination host asset data. (String)	s_RV95
DestinationLineOfBusiness	rv96	string	Destination Line Of Business. Part of destination host asset data. (String)	s_RV96
DestinationDivision	rv97	string	Destination Division. Part of destination host asset data. (String)	s_RV97
DestinationDepartment	rv98	string	Destination Department. Part of destination host asset data. (String)	s_RV98
DestinationAssetId	rv99	string	Destination Asset Id. Part of destination host asset data. (String)	s_RV99
CustomerHierarchyLevel4	rv100	string	Customer Hierarchy Level 4 (used by MSSPs)	s_RV100
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_CV29
SARBOX	cv90	string	Set to 1 if the asset is governed by Sarbanes-Oxley through an asset map. (String)	s_CV90
HIPAA	cv91	string	Set to 1 if the asset is governed by the Health Insurance Portability and Accountability Act regulation through an asset map. (String)	s_CV91
GLBA	cv92	string	Set to 1 if the asset is governed by the Gramm-Leach Bliley Act regulation through an asset map. (String)	s_CV92

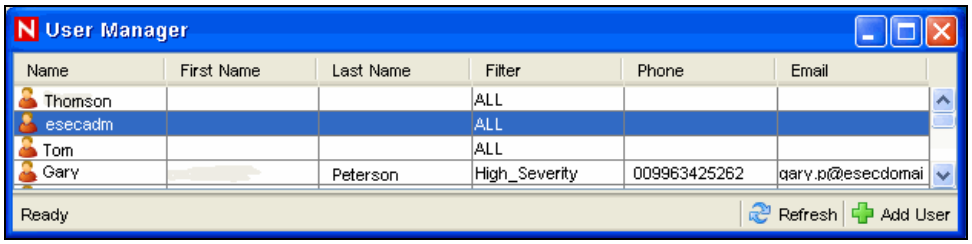
Label	Meta-tag	Type	Description	Collector Variable
FISMA	cv93	string	Set to 1 if the asset is governed by the Federal Information Security Management Act (FISMA) regulation through an asset map. (String)	s_CV93
NISPOM	cv94	string	Set to 1 if the asset is governed by National Industrial Security Program Operating Manual (NISPOM) regulation through an asset map. (String)	s_CV94
SIPCountry	cv95	string	Source Country based on Source Ip. (String)	s_CV95
DIPCountry	cv96	string	Destination Country based on Destination Ip. (String)	s_CV96
CustomerVar97 thru CustomerVar100	cv97 thru cv100	string	Reserved for use by customers for customer-specific data. (String)	s_CV97 thru s_CV100
DeviceEventTimeString	et	string	The normalized date and time of the event, as reported by the sensor.	s_ET
SentinelProcessTime	spt	date	The date and time Sentinel received the event.	Not Applicable
BeginTime	bgnt	date	The date and time the event started occurring.	s_BGNT
EndTime	endt	date	The date and time the event stopped occurring.	s_ENDT
RepeatCount	rc	integer	The number of times the same event occurred if multiple occurrences were consolidated.	s_RC
SourcePortName	sp	string	The source port from which the event originated.	s_SP
DestinationPortName	dp	string	The destination port to which the event was targeted.	s_DP

5 Sentinel Control Center User Permissions

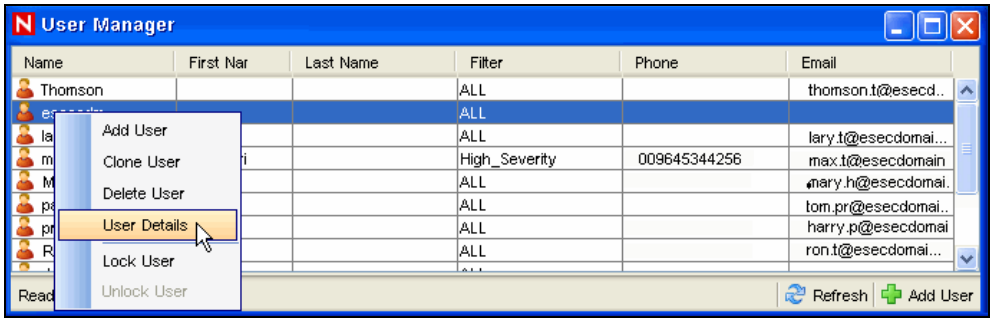
Sentinel allows administrators to set user permissions in the Sentinel Control Center at a granular level. The only user created by default is the esecadm, or Sentinel Administrator. All other users are created by the Sentinel Administrator, or someone with similar permissions.

To change user permissions:

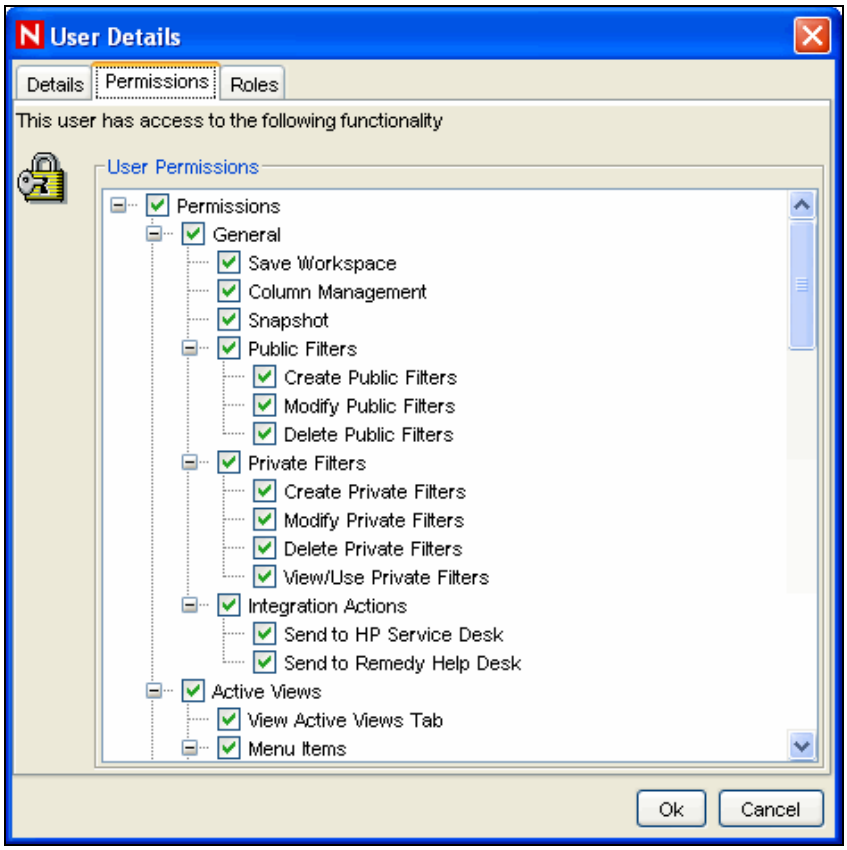
1. Log into the Sentinel Control Center as a user with "User Management" permissions.
2. Click the Admin tab.
3. Select *User Configuration* from *Admin* tab. Alternatively, Select *User Manager* from *User Configuration* in the Navigator.



4. Right click user and select *User Details.*



5. Select the *Permissions* tab.



6. Uncheck the checkboxes for which you want to restrict user.
7. Click *OK*.

The permissions in the User Manager are grouped into several major categories:

- "General"
- "Active Views"
- "Correlation"
- "iTRAC"
- "Incidents"
- "Event Source Management"
- "Analysis"
- "Advisor"
- "Administration"

Each of these groups of setting is described in more detail below.

General

Permission Name	Description
Save Workspace	Allows user to save preferences. If this permission is unavailable, user will never be prompted to save changes to preferences when logging out or exiting the Sentinel Control Center.
Column Management	Allows user to manage the columns in the Active View tables.
Snapshot	Allows user to take a snapshot of Active View tables.

General – Public Filters

Permission Name	Description
Create Public Filters	Allows user to create a filter with an owner ID of PUBLIC. If user does not have this permission, then the value PUBLIC will not be listed as one of the owner IDs that user can create a filter for.
Modify Public Filters	Allows user to modify a public filter.
Delete Public Filters	Allows user to delete a public filter.

General – Private Filters

Permission Name	Description
Create Private Filters	Allows user to create private filters for themselves or for other users.
Modify Private Filters	Allows user to modify their own private filters and private filters created by other users.
Delete Private Filters	Allows user to delete their own private filters and private filters created by other users.
View/Use Private Filters	Allows user to view/use their own private filters and private filters crated by other users.

General – Integration Actions

Permission Name	Description
Send to HP Service Desk	Allows user to send events, incident and associated objects to HP Service Desk. (requires the optional HP integration component)
Send to Remedy Help Desk	Allows user to send events, incident and associated objects to Remedy. (requires the optional Remedy integration component)

Active Views

Permission Name	Description
View Active Views Tab	Allows user to see and use the Active Views tab, menu and other related functions associated with the Active Views tab.

Active Views – Menu Items

Permission Name	Description
Use Assigned Menu Items	Allows user to use assigned menu items in the Active Views Events table (the right-click menu).
Add to Existing Incident	Allows user to add events to existing incidents using the Active Views Events table (the right-click menu).
Remove from Incident	Allows user to remove events from an existing incident using the Events tab Events table (the right-click menu).
Email Events	Allows user to e-mail events using the Active Views Events table (the right-click menu).
View Advisor Attack Data	Allows user to view the Advisor Attack Data stream.
View Vulnerability	Allows user to view the vulnerabilities present in the Sentinel database

Active Views – Active Views

Permission Name	Description
Use/View Active Views	Allows user to access the Active Views charts.

iTRAC

Permission Name	Description
View iTRAC Tab	Allows user to see and use the iTRAC tab, menu and other related functions associated with the iTRAC tab.
Activity Management	Allows user to access the Activity Manager.
WorkItem Management	Allows user to use the steps in a process.

iTRAC - Template Management

Permission Name	Description
View/Use Template Manager	Allows user to access the Template Manager.
Create/Modify Templates	Allows user to create and modify templates.

iTRAC - Process Management

Permission Name	Description
View/Use Process Manager	Allows user to access the Process View Manager.
Control Processes	Allows user to use the Process View Manager.

Correlation

Permission Name	Description
View Correlation Tab	Allows user to use the Correlation functions.
View/Use Correlation Rule Manager	Allows user to start or stop the Correlation Rules.
View/Use Correlation Engine Manager	Allows user to deploy/undeploy the Correlation Rules.
View/Use Correlation Action Manager	Allows user to create/associate Actions to the Correlation Rules.
View/Use Dynamic Lists	Allows user to Create, use, view, modify the Dynamic Lists.

Incidents

Permission Name	Description
View Incidents Tab	Allows user to see and use the Incidents tab, menu and other related functions associated with the View Incidents tab.
Incident Administration	Allows user to modify an incident.

Permission Name	Description
View Incident(s)	Allows user to view/modify the details of an incident. If the user does not have this permission, then the Incident Details window will not be displayed when the user either double-clicks an Incident in the "Incident View" window or right-clicks the incident or selects the "Modify" option.
Create Incident(s)	Allows user to create Incidents in the in the "Incident View" window or by right clicking on the incident and select "Modify" option. Alternatively you may select "Create Incident" menu item in the "Incidents" menu bar and clicking "Create Incident" option in the tool bar.
Modify Incident(s)	Allows user to modify an incident in the Incident Details window.
Delete Incident(s)	Allows user to delete incidents.
Assign Incident(s)	Allows user to assign an incident in the Modify and Create Incident window.
Email Incidents	Allows user to e-mail Incidents of interest.
Incident Actions	Allows user to enable/disable the Incident Action Configuration/Execution.
Add Notes	Allows user to add any number notes to an incident.

Event Source Management

Permission Name	Description
View Status	Allows user to view the status of ESM components.
View Scratchpad	Allows user to design and configure ESM components.
Configure ESM Components	Allows you to configure ESM components.
Control ESM Components	Allows you to control and manage ESM components.
Manage Plugins	Allows you to manage Collector and Connector Plugins.
View Raw Data	Allows you to view/parse raw data.
Debug Collector	Allows you to debug Collector.

Command and Control consists of:

- start/stop individual ports
- start/stop all ports
- restart hosts
- rename hosts

Analysis Tab

Permission Name	Description
View Analysis Tab	Allows user to see and use the View Analysis tab, menu and other related functions associated with the View System Overview tab.

Advisor Tab

Permission Name	Description
View Advisor Tab	Allows 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 user to see and use the View Administration tab, menu and other related functions associated with the View Administration tab.
Archive Configuration	Allows user to set database archive parameters.
DAS Statistics	Allows user to view DAS activity (DAS binary and query).
Event Configuration	Allows user to rename columns, set mappings from mapping files. This function is associated with Mapping Configuration.
Mapping Configuration	Allows user to add, edit and delete mapping files.
Menu Configuration	Allows user to access the Menu Configuration window and add new options that display on the Event menu when you right-click an event.
Reporting Data Configuration	Allows user to enable or disable summary tables used in aggregation.
User Management	Allows user to add, modify and delete user details
User Session Management	Allows user to view, lock and terminate active users (logins to Sentinel Control Center).
iTRAC Role Management	Allows user to view and use the role manager in the Admin Tab.

Administration – Global Filters

Permission Name	Description
View/Use Global Filters	Allows user to access the Global Filter Configuration window.
Modify Global Filters	Allows user to modify the global filters configuration. **NOTE:** To access this function, View Global Filters permission must also be assigned.

Administration – Server Views

Permission Name	Description
View Servers	Allows user to monitor the status of all processes.
Control Servers	Allows user to start, restart and stop processes.

6 Sentinel Correlation Engine RuleLG Language

Correlation RuleLG Language Overview

The Sentinel Correlation Engine runs rules that are written in the Correlation RuleLg language. Rules are created in the Sentinel Control Center. Users can create rules using a wizard for the following rule types:

- Simple Rule
- Composite Rule
- Aggregate Rule
- Sequence Rule

These rules are converted to the Correlation RuleLg language when the rules are saved. The same rule types, plus even more complex rules, can be created in the Sentinel Control Center using the Custom/Freeform option. To use the Custom/Freeform option, the user must have a good understanding of the Correlation RuleLg language.

RuleLg uses several operations, operators, and event field metatags to define a rule. The Correlation Engine loads the rule definition and uses the rules to evaluate, filter, and store in memory events that meet the criteria specified by the rule.. Depending on the rule definition, a correlation rule may fire based on

- the value of one field or multiple fields
- the comparison of an incoming event to past events
- the number of occurrences of similar events within a defined time period
- one or more subrules firing
- one or more subrules firing in a particular order

Each of these constructs is represented by an operation in RuleLg.

Event Fields

All operations function on event fields, which may be referred to by their labels or by their metatags within the correlation rule language. For a full list of labels and metatags, see Sentinel Metatags in *Sentinel 6.0 User Reference Guide*. The label or metatag must also be combined with a prefix to designate whether the event field is part of the incoming event or a past event that is stored in memory.

Examples:

```
e.DestinationIP (Destination IP for the current
event)
e.dip (Destination IP for the current event)
w.dip (Destination IP for any stored event)
```

WARNING:

If you rename the label of a metatag, do not use the original label name when creating a correlation rule.

Event Operations

Event operations evaluate, compare, and count events. They include the following operations:

- **Filter:** Evaluates the current to determine whether they could potentially trigger a rule to fire
- **Window:** Compares the current event to past events that have been stored in memory
- **Trigger:** Counts events to determine whether enough events have occurred to trigger a rule

Each operation works on a set of events, receiving a set of events as input and returning a set of events as output. 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.

Filter Operation

Filter consists of a Boolean expression that evaluates the current event from the real-time event stream. It compares event attributes to user-specified values using a wide set of operators

The Boolean expression is a composite of comparison and match instructions.

The syntax for *filter* is:

```
Filter <Boolean expression 1> [NOT|AND|OR
<Boolean expression 2] […] [NOT|AND|OR <Boolean
expression n>]
```

Where

```
<Boolean expressions 1…n> are expressions using
one or more event field names and filter
operators
```

For example, this rule detects whether the current event has a severity of 4 and the resource event field contains either "FW" or "Comm."

```
filter(e.sev = 4 and (e.res match regex ("FW") or
e.res match regex ("Comm")))
```

Boolean Operators

Filter expressions can be combined using the Boolean operators AND, OR and NOT. The filter boolean operator precedence (from highest [top] to lowest [bottom] precedence) is:

Operator	Meaning	Operator Type	Associativity
Not	logical not	unary	None
And	logical and	binary	left to right
Or	logical or	binary	left to right

In addition to Boolean operators, *filter* supports the following operators.

Standard Arithmetic Operators

Standard arithmetic operators can be used to build a condition that compares the value of a Sentinel metatag and a user-specified value (either a numeric value or a string field). The standard arithmetic operators in Sentinel are =, <, >, *!*=, <=, and >=.

Examples:

```
filter(e.Severity > 3)
filter(e.BeginTime < 1179217665)
filter(e.SourceUserName != “Administrator”)
```

Match Regex Operators

The *match regex* operator can be used to build a condition where the value of a metatag matches a user-specified regular expression value specified in the rule. This operator is used only for string metatags, and the user-specified values for this operator are case-sensitive.

Examples:

```
filter(e.Collector match regex ("IBM"))
filter(e.EventName match regex ("Attack"))
```

Match Subnet Operators

The *match subnet* operator can be used to build a condition where the value of a metatag maches a user-specified subnet specified in the rule in CIDR notation. This operator is used only for IP address metatags.

Example:

```
filter(e.DestinationIP match subnet
(208.130.28.0/22))
```

Inlist Operator

The *inlist* operator is used to perform a lookup on an existing dynamic list of string values, returning true if the value is present in the list. For more information on Dynamic Lists, see Correlation Tab in *Sentinel 6.0 User Guide*..

For example, this filter expression is used to evaluate whether the Source IP of the current event is present on a dynamic list called MailServerList. If the Source IP is present in this list, the expression evaluates to TRUE.

```
filter(e.sip inlist MailServerList)
```

As another example, this filter expression combines the NOT and the INLIST operator. This expression evaluates to TRUE if the Source IP is not present in the dynamic list called MailServerList.

```
filter(not (e.sip inlist MailServerList))
```

This filter expression is used to evaluate whether the event name of the current event equals “File Access” and the Source User Name is also not present on a dynamic list called AuthorizedUsers. If both conditions are true for the current event, the expression evaluates to TRUE.

```
filter(e.evt="File Access" and not(e.sun inlist
AuthorizedUsers))
```

ISNULL Operator

The *isnull* operator returns true if the metatag value is equal to NULL.

Example:

```
Filter(isnull(e.SIP))
```

Output Sets

- The output of a filter is either the empty set (if the Boolean expression evaluates to false) or a set containing the current event and all of the other events from the incoming set (if the Boolean expression evaluates to true).
- If *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 trigger events are the filter operation output set of events with the current event first.
- If *filter* is not the last operation of a correlation rule (that is, *filter* is followed by a *flow* operatior), then the output set of a filter is used as the input set to other operations (through the flow operator).

Additional Information

- The *filter* operator can be used to compare metatag values with other metatag values, for example:
  `e.SourceIP=e.DestinationIP`

Window Operation

Window compares the current event to a set of past events that are stored in a "window." The events in the window may be all past events for a certain time period, or they may be filtered.

The Boolean expression is a composite of comparison instructions and match instructions with the Boolean operators AND, OR and NOT.

The syntax for *window* is:

```
Window (<Boolean expression>[, <filter
expression>, <evaluation period>)
```

Where

```
<Boolean expression> is an expression comparing a
metatag value from the current event to a metatag
value from a past event (or a user-specified
constant)
<filter expression> is optional and specifies
filter criteria for the past events
<evaluation period> specifies the duration for
which past events matching the filter expression
are maintained, specified in seconds (s), minutes
(m), or hours (h). If no letter is specified,
seconds are assumed.
```

For example, this rule detects whether the current event has a source IP address in the specified subnet (208.130.28.0/22) and matches an event(s) that happened within the past 60 seconds.

```
window(e.sip = w.sip, filter(e.sip match subnet
(208.130.28.0/22),60)
```

As another example, this rule is a domino type of rule. An attacker exploits a vulnerable system and uses it as an attack platform.

```
window((e.sip = w.dip AND e.dp = w.dp AND e.evt =
w.evt), 1h)
```

This rule identifies a potential security breach after a denial of service attack. The rule fires if the destination of a denial of service attack has a service stopped within 60 seconds of the attack.

```
filter(e.rv51="Service" and e.rv52="Stop" and
e.st = "H") flow window (e.sip = w.dip,
filter(e.rv52="Dos"), 60s) flow trigger(1,0))
```

Output Sets

- 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 matching past events.
- If no events in the window match the current event for the simple boolean expression, the output set is empty.
- 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).

Additional Information

- You must prepend a metatag name with "e." to specify the current event or with "w." to specify the past events
- All *window* simple Boolean expressions must include a metatag in the form w.[metatag].
- For more information about valid *filter* expressions, see Filter Operation.
- 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. With extremely high event rates or long durations, this may require a large amount of memory.
- The current event is not placed into the window until after the current event window evaluation is complete
- To minimize memory usage, only the relevant parts of the past events, not all metatag values, are maintained in memory.

Trigger Operation

Trigger is used to specify a number of events for a user-specified duration.

The syntax for *trigger* is:

```
Trigger (<number of events>, <evaluation
period>[, discriminator (<list of metatags>))
```

Where

```
<number of events> is an integer value specifying
the number of matching events that are necessary
for the rule to fire
<evaluation period> specifies the duration for
which past events matching the filter expression
are maintained, specified in seconds (s), minutes
(m), or hours (h).  If no letter is specified,
seconds are assumed.
discriminator is a field to group by
```

For example, this rule detects if 5 events with the same source IP address happen within 10 seconds.

```
trigger(5,10,discriminator(e.sip))
```

Output Sets

- 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.
- 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, it is followed by a *flow* operator), then the output set of a trigger is used as the input set to other operations (through the flow operator).
- 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.

Rule Operations

Rule operations work on subrules that have been combined into a compound rule. They include:

- Gate
- Sequence

Gate Operation

The *gate* operation is used to create a composite rule which is used in identifying complex situations from the occurrence of simple situations.

The composite rule is made up of one or more nested subrules and can be configured to fire if some, any or all of the subrules fire within a specified time window. The subrules may be a simple rule or another composite rule. For more information on Composite Rule, see Correlation Tab in *Sentinel 6.0 User's Guide*.

The syntax for *gate* is:

```
Gate(<subrule 1 rulelg>, <subrule 2
rulelg>…<subrule n ruleLg>,  <mode>, <evaluation
period>, discriminator(<list of metatags>))
```

Where

```
Subrule Rulelgs are the rulelg definitions for 1
to n subrules
mode = all | any | 1 | 2 | … | n, which is the
number of subrules that must be triggered in
order for the gate rule to trigger
<evaluation period> specifies the duration for
which past events matching the filter expression
are maintained, specified in seconds (s), minutes
(m), or hours (h).  If no letter is specified,
seconds are assumed.
discriminator is a field to group by
```

For example, this rule is a typical perimeter security IDS inside/outside rule

```
filter(e.sev > 3) flow gate(filter(e.sn = "in"),
filter(e.sn = "out"), all, 60s,
discriminator(e.dip, e.evt))
```

Sequence Operation

Sequence rules are similar to *gate* rules, except that all child rules must fire in time order for the sequenced rule to evaluate to true.

The subrules may be a simple rule or another composite rule.

The syntax for *sequence* is:

```
Sequence(<subrule 1 rulelg>, <subrule 2
rulelg>…<subrule n ruleLg>, <evaluation period>,
discriminator(<list of metatags>))
```

Where

```
Subrule Rulelgs are the rulelg definitions for 1
to n subrules
<evaluation period> is a time period expressed in
seconds (s), minutes (m), or hours (h)
discriminator is a field to group by
```

For example, this rule detects three failed logins by a particular user in 10 minutes followed by a successful login by same user.

```
sequence (filter(e.evt="failed logins") flow
trigger(3, 600, discriminator(e.sun,e.dip)),
```

```
filter(e.evt="goodlogin"), 600,
discriminator(e.sun, e.dip))
```

Operators

Operators are used to transition between operations or expressions. The fundamental operators used between operations are:

- Flow operator
- Union operator
- Intersection operator

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. Flow is typically used to transition from one correlation operation to the next.

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)
```

Discriminator Operator

The *discriminator* operator allows users to group by event fields within other event operations. *Discriminator* can be used within the *trigger*, *gate*, or *sequence* operations This is the last operation while executing a condition. The input for this operator would generally be the output of other operations, if any.

For example, this filter expression is used to identify five severity 5 events within 60s that all have the same Source IP. Note that the attribute (SIP in this example) can be any value, even a NULL, but it must be the same for all five events in order for the rule to fire.

```
filter(e.sev=5 ) flow trigger(5, 60s,
discriminator(e.sip)
```

Order of Operators

The 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

Differences between Correlation in 5.x and 6.x

There are several new functionalities updated / included in 6.0 to widen the usage of Correlation to meet user's requirements and for the ease-of-use.

- Gate Operation: This is new in 6.0.
- Sequence Operation: This is new in 6.0.
- Inlist Operator and Dynamic Lists: These are new in 6.0.
- Isnull Operator: This is new in 6.0. For metatag values equal to null, Sentinel 5.x supported the following syntax which is replaced by the ISNull operator in Sentinel 6.0

  ```
  e.SIP= " "
  ```

- Update Window: This is new in 6.0
- Sentinel 6.0 merges the "C" (Correlated Events) and "W" (watchlist events) SensorTypes. All events generated by the Correlation Engine are now labeled "C" in the SensorType field.
- Correlation Actions and Correlation Rules: Correlation Actions and Correlation Rules are decoupled in Sentinel 6.0
- Although the *filter* operation supported AND and OR Boolean expressions in Sentinel 5.x, the *window* operation supports Boolean expressions for the first time in Sentinel 6.0. For example:

  ```
  OR: window(e.dip=w.dip OR e.sip=w.sip,
     filter(e.sev>2),60)

  AND: window(e.evt=w.evt AND e.sun=w.sun,
     filter(e.sev>2),60)
  ```

- Sentinel 6.0 no longer has the GUI option to create a rule from a PUBLIC filter. The filter criteria must be defined in the correlation wizard or language.
- The update functionality for a rule that is triggered more than once is configurable in Sentinel 6.0. In Sentinel 5.1.3, updates to a rule were based on a sliding window based on the trigger time period. In Sentinel 6.0, the update functionality can be set when the rule is deployed; the rule actions may happen every time the rule is triggered, or they may be set to occur once and then wait for some period of time before the action occurs again. This prevents multiple notifications on a single, ongoing event.
- The *in* and *not in* operators are deprecated in Sentinel 6.0. Correlation rules using these operators must be modified before running them in Sentinel 6.0.
- The *e.all* metatag has been deprecated. Correlation rules using this operator should be updated to use specific metatags before running them in Sentinel 6.0.

7 Sentinel Data Access Service

The Data Access Service (DAS) process is Sentinel Server's persistence service and provides a message bus interface to the database. Some of the services it provides are event storage, Historical Query, event drill down, vulnerability and Advisor data retrieval, and configuration manipulation.

DAS Container Files

DAS is a collection of services provided by five different processes. Each process is a container responsible for different types of database operations. These processes are:

- **DAS Query:** Performs general Sentinel Service operations including Login and Historical Query.
- **DAS Binary:** Performs event database insertion.
- **DAS RT:** Provides the server-side functionality for Active Views.
- **DAS Aggregation:** Calculates event data summaries that are used in reports.
- **DAS iTRAC:** Provides the server-side functionality for the Sentinel iTRAC functionality.
- **DAS CMD:** Provides a command line interface to certain DAS services. Used primarily for third-party integration.
- **DAS Proxy:** Provides the server-side of the SSL proxy connection to Sentinel Server.

DAS Proxy is not directly part of the DAS collection of services. It is part of the Communication Server and does not directly connect to the database.

Reconfiguring Database Connection Properties

The primary settings in these configuration files that can be configured using the dbconfig utility are related to the database connection, including:

- username
- password
- hostname
- port number
- database (database name)
- server (oracle, oracle10g, or mssql)

If any of these database connection settings need to be changed, they must be changed in every das_*.xml file using the dbconfig utility. Using the *–a* argument, this utility can update all files at once (For example, update all files in the %ESEC_HOME%\config or $ESEC_HOME/config directory). Alternately, using the *–n* argument, this utility can update a single file's contents if only one file needs to be updated. Typically, all files should be updated at once.

WARNING:

Do not manually edit the database connection properties. Use the dbconfig utility to change any database connection values within these files.

To Reconfigure Database Connection Properties:

1. Login to the machine where DAS is installed as the esecadm user on UNIX or a user with administrative rights on Windows.
2. Go to:

   For Windows:

   ```
   %ESEC_HOME%\bin
   ```

   For UNIX:

   ```
   $ESEC_HOME/bin
   ```

3. Enter the following command:

   For Windows:

   ```
   dbconfig –a %ESEC_HOME%\config [[-u username]
      [-p password] | [-winAuth]] [-h hostname]
      [-t portnum] [-d database] [-s server] [-
      help] [-version]
   ```

   For UNIX:

   ```
   dbconfig –a $ESEC_HOME/config [-u username] [-
      p password] [-h hostname] [-t portnum] [-d
      database] [-s server] [-help] [-version]
   ```

NOTE: The *-winAuth* argument is available only on Windows and should be used instead of the *–u* and *–p* arguments if the Sentinel Application User is a Windows Authentication user.

Other settings in the files may be adjusted manually (without using dbconfig):

- maxConnections
- batchSize
- loadSize

Changing these settings may affect database performance and should be done with caution

DAS Logging Properties Configuration Files

The following files are used to configure logging of the DAS process. These files are typically changed when troubleshooting the DAS process.

- das_query_log.prop
- das_binary_log.prop
- das_rt_log.prop
- das_itrac_log.prop
- das_aggregation_log.prop
- das_cmd_log.prop

- das_proxy_log.prop

They are located in the following locations:

For Windows:

```
%ESEC_HOME%\config
```

For UNIX:

```
$ESEC_HOME/config
```

These files contain the configuration that determines how the DAS processes will log messages. The most important part of the configuration is the logging levels, which indicate how verbose the log messages should be. The section of the file to configure these settings is:

```
###### Configure the logging levels
# Logging level rules are read from the top down.
# Start with the most general, then get more
specific.
#
# Defaults all loggers to INFO (enabled by
default)
.level=INFO
#
# < Set level of specific loggers here >
#
# Turns off all logging (disabled by default)
#.level=OFF
######
```

NOTE: The logger *.level* is a wildcard logger name that refers to all loggers. Setting this logger's level will affect all loggers.

The available logging 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- for debugging
- 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.

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. Therefore, it is not necessary to restart the processes to begin using the updated logging levels.

Log messages are written to ESEC_HOME%\log (for Windows) or $ESEC_HOME/log (for UNIX), in the following files:

```
das_query_0.*.log
das_binary_0.*.log
das_itrac_0.*.log
das_aggregation0.*.log
das_rt0.*.log
das_cmd0.*.log
das_proxy0.*.log
```

The 0 indicates the unique number to resolve conflicts and the * indicates a generation number to distinguish rotated logs. For example, das_query0.0.log is the log with index 0 (latest) file in a rotated set of log files for the DAS Query process.

Log messages are also written to the process's console (standard output). However, since the processes are running as services, users do not have access to the console output. It is possible, however, to capture the console output in the sentinel0.*.log file. This is useful, for example, if the process is producing an error that is not printed to the process's own log file. This can be enabled by adding the following line to the *sentinel_log.prop* file:

```
esecurity.base.process.MonitorableProcess.level=F
INEST
```

8 Sentinel Accounts and Password Changes

This chapter discusses users that are created or used during Sentinel installation and normal Sentinel operations. Unless you create domain users in advance in order to use Windows Authentication, these users are created by the Sentinel installer. These user accounts are used for Sentinel's normal operations, such as event inserts into the Sentinel database.

The administrator may choose to occasionally change the passwords for these accounts. To ensure continued normal Sentinel operations, there are special procedures necessary to update the passwords in all necessary locations.

Sentinel Default Users

Native Database Authentication

Installer creates several users during installation if you use native database authentication (Oracle or Microsoft SQL Server). These users are all created as database users in the Oracle or SQL Server database, and the passwords are configurable at install time. The installer will create the users with the following default names:

- **esecdba**: Schema owner
- **esecadm:** Sentinel administrator
- **esecrpt:** Reporter user, same password as the admin user
- **esecapp:** Sentinel application user. Used by Sentinel Server to connect to the database

In addition to creating a database user for the Sentinel administrator, the installer also creates a Sentinel user with the same username and password for the Sentinel Control Center. For UNIX only, the installer creates an operating system user with no password set. To log in as this user, the UNIX administrator must set a password or *su* to the user as root.

Windows Authentication

If you use Windows authentication, the Windows administrator must create several domain accounts before the installation is started. The credentials for these accounts must be given during the Sentinel installation:

- **Sentinel DB Administrator:** Schema owner
- **Sentinel Administrator:** Sentinel administrator
- **Sentinel Report User:** Reporter user, same password as the admin user.
- **Sentinel Application User:** Sentinel application username for connecting to the database.

Windows Authentication users are supported only when SQL Server is being used and DAS is running on Windows.

Password Changes

Corporate policy may require that passwords be changed on a regular schedule. Sentinel user passwords can be changed using database utilities. After changing a password, some Sentinel components need to be updated to use the new password.

Changing Password

SQL Server Accounts

On Windows, this procedure can be used to change the password for the Sentinel Application User, the Sentinel Database User, or the Sentinel Report User.

To change password in MS SQL Server Management Studio:

1. Open the MS SQL Enterprise Manager/ MS SQL and select *Security* > *Logins*.
2. Right-click a username from the right pane and select properties.
3. Change the password. Click *OK*.

   Follow the procedures in Sentinel updates after a password change.

Oracle Accounts

This procedure can be used to change the password for the Sentinel Application User, the Sentinel Database User, or the Sentinel Report User.

To change password in Oracle:

1. Connect to Oracle Enterprise Manager with user having sysdba privilege.
2. Select your specific database from the left pane.
3. In *Database* > *Security* > *Users*, select a user for which you want to change the password.
4. Enter new password and confirm the password. Click *Apply*.

   Follow the procedures in Sentinel updates after a password Change.

Windows Domain Accounts

If the Sentinel system uses domain user accounts and Windows Authentication, use the following password change procedures. These procedures can be used for the Sentinel Administrator, the Sentinel Database User, the Sentinel Report User, and the Sentinel Application User. It can also be used for any Sentinel Control Center account that uses Windows Authentication.

To change the password for Windows domain accounts:

1. Log into a machine using the account and use standard Windows password change procedures

   OR

   Request a password change from a Windows administrator.
2. Follow the procedures in Sentinel updates after a password change.

Sentinel Control Center Accounts (Native DB Authentication)

This procedure can be used to change the password for the Sentinel Administrator account or any other Sentinel Control Center user.

To change the Sentinel Administrator password:

1. Login to the Sentinel Control Center as the Sentinel Administrator or another user with User Management permissions.
2. Click *Admin* > *User Configuration*. The *User Manager* window displays.
3. Double-click esecadm user account or right-click *User Details*.
4. Modify the account password and confirm password. Click *OK*.

   No additional updates are needed in the Sentinel system.

Sentinel Control Center Accounts (Windows Authentication)

Use standard procedures for changing the password for Windows domain accounts.

Sentinel Updates After a Password Change

The passwords for certain Sentinel users, such as the Sentinel Database User and the Sentinel Application User, are encrypted and stored in configuration files and used in normal Sentinel operations. These configuration files must be updated after the passwords are changed.

Updating Sentinel Application User Password

The Sentinel Application User credentials are stored encrypted in the container xml files. After a password change, these files must be updated for Sentinel to continue working.

The procedures are different depending on whether the Sentinel Application User uses Native Database Authentication or Windows Authentication.

To update the Sentinel Application User password (Native DB Authentication):

1. Change the password for the Sentinel Application User (esecapp by default) using database utilities as described in “Changing Password”.
2. 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.

   The container xml files are located in the following locations:

   For Windows:

   ```
   %ESEC_HOME%\config
   ```

   For Oracle:

   ```
   $ESEC_HOME/config
   ```

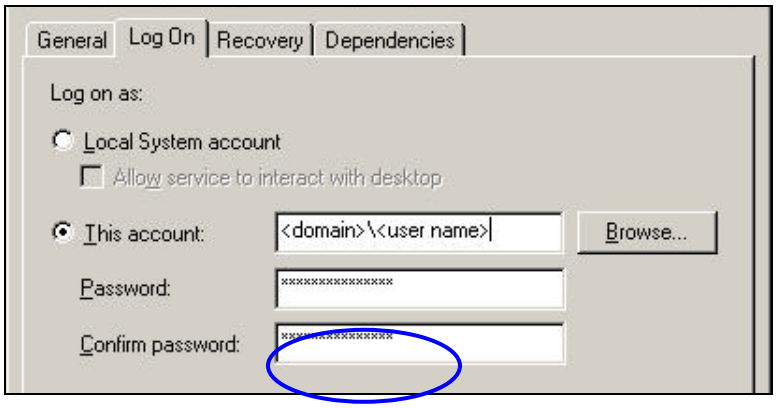
   For more information on usage of the dbconfig utility, go to Sentinel Reference Guide, Chapter 9 - Sentinel Data Access Service.

   ```
   dbconfig –a {$ESEC_HOME/config |
      %ESEC_HOME%\config} -p <password>
   ```

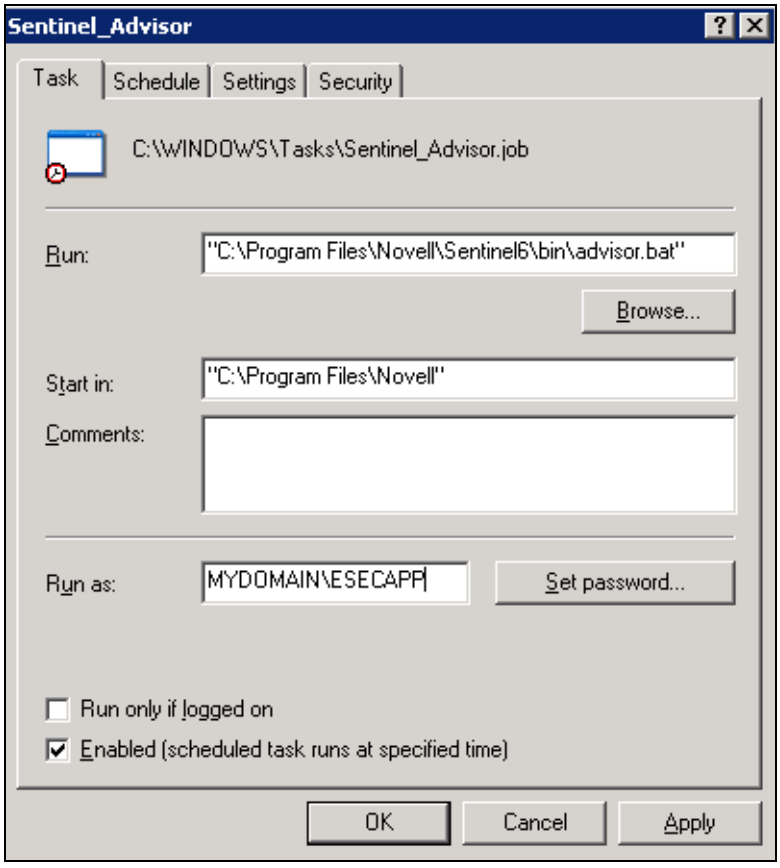
To update the Sentinel Application User password (Windows Authentication):

1. Change the password for the Sentinel Application User domain account as described in “Changing Password”.

2. On your DAS machine, open Windows Services (*Control Panel* > *Administrative Tools* > *Services*).
3. Right-click *Sentinel* > *Properties*. Click the *Log On* tab and update *Log on as* password. Click *Apply* and click *OK*.



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*.

Updating Sentinel Database User Password

These password change procedures are only necessary if extra Sentinel Data Manager jobs have been created and scheduled or the Sentinel Data Manager command line interface is being used.

To change Sentinel DB Administrator password (Windows Authentication):

1. Use the Windows Operating System to change the password as described in "Changing Password".
2. If you are running any SDM command line scheduled tasks in your environment, you will need to update the *Run as* property (*Control Panel* > *Scheduled Tasks* > right-click *Properties*).
3. Click *Set password*. Enter the new password twice and click *OK*. Click *Apply* and click *OK*.

To update the Sentinel DB Administrator password (Native DB Authentication):

1. Change the password for the Sentinel DB Administrator User (esec by default) using database utilities password as described in "Changing Password".
2. In order for automated SDM command line tasks to continue to work (if applicable in your environment), update the dbPass in the sdm.connect file with the new esecdba password using the SDM GUI or command line. For more information, see Chapter 10 Sentinel Data Manager in *Sentinel 6 User Guide.*

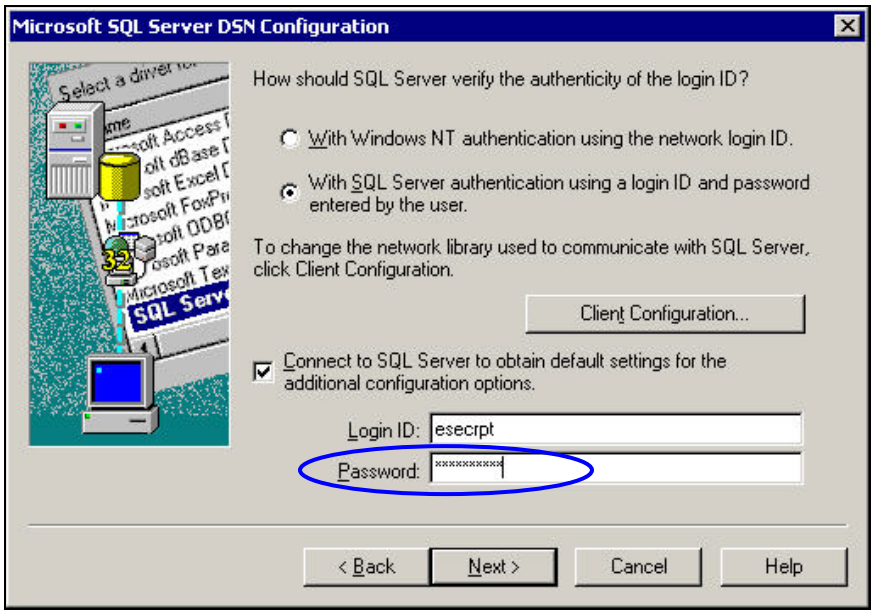
```
sdm -action saveConnection -server
   <oracle/mssql> -host <hostIp/hostName> -
   port <portnum> -database <databaseName/SID>
   [-driverProps <propertiesFile>] {-user
   <dbUser> -password <dbPass>} -connectFile
   <filenameToSaveConnection>
```

Updating Sentinel Report User Password

This procedure is only necessary for Crystal on Windows. For Crystal on Linux, no changes are necessary.

To update the Sentinel Report User password for Crystal on Windows:

1. Change the password for the Sentinel Report User (esecrpt by default) using database utilities as described in "Changing Password".
2. Log into the Crystal Server machine.
3. Go to *Control Panel>Administrative Tools>Data Sources (ODBC)* to update the ODBC Data Source Name (DSN).
4. Under the System DSN tab, highlight sentineldb and click *Configure*.
5. Click *Next*. Update the password.



6. Click *Next* until you get a Finish button. Click *Finish*.

9 Sentinel Database Views for Oracle

This chapter lists the Sentinel 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	integer	Annotation identifier - sequence number.
CVE	varchar2 (13)	
DATE_CREATED	date	Insert date
DATE_MODIFIED	date	Last update date
CREATED_BY	integer	By user ID
MODIFIED_BY	integer	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	integer	Annotation identifier - sequence number.
SERVICE_PACK_ID	integer	
VENDOR	varchar2(128)	
PRODUCT	varchar2(128)	
VERSION	varchar2(128)	Contains the version number
SERVICE_PACK	varchar2(128)	
PRIMARY_FLAG	integer	
DATE_CREATED	date	Insert date
DATE_MODIFIED	date	Last update date
CREATED_BY	integer	By user ID
MODIFIED_BY	integer	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	integer	Annotation identifier - sequence number.
VERSION	integer	Contains the version number
TEMPLATE_ID	integer	

Column Name	Datatype	Comment
TEMPLATE_NAME	varchar2(256)	
THREAT_CATEGORY_NAME	varchar2(128)	
THREAT_TYPE_NAME	varchar2(128)	
HEADLINE	Clob	
FIRST_PUBLISHED	Date	
LAST_PUBLISHED	Date	
STATUS	varchar2(32)	
URGENCY_ID	integer	
CREDIBILITY_ID	integer	
SEVERITY_ID	integer	
SUMMARY	Clob	
LEGAL_DISCLAIMER	Clob	
COPYRIGHT	varchar2(1024)	
BEGIN_EFFECTIVE_DATE	date	
END_EFFECTIVE_DATE	date	
DATE_CREATED	date	Insert date
DATE_MODIFIED	date	Last update date
CREATED_BY	integer	By user ID
MODIFIED_BY	integer	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	integer	
ALERT_ID	integer	
DATE_CREATED	date	Insert date
DATE_MODIFIED	date	Last update date
CREATED_BY	integer	By user ID
MODIFIED_BY	integer	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	integer	
CVE	varchar2(13)	
DATE_CREATED	date	Insert date
DATE_MODIFIED	date	Last update date
CREATED_BY	integer	By user ID
MODIFIED_BY	integer	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	integer	

Column Name	Datatype	Comment
ATTACK_ID	integer	
SERVICE_PACK_ID	integer	
ATTACK_NAME	varchar2(256)	
ATTACK_CODE	varchar2(256)	
DATE_CREATED	Date	Insert date
DATE_MODIFIED	date	Last update date
CREATED_BY	integer	By user ID
MODIFIED_by	integer	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	integer	
ATTACK_ID	integer	
SERVICE_PACK_ID	integer	
PLUGIN_ID	varchar2(256)	
PLUGIN_NAME	varchar2(256)	
DATE_CREATED	date	Insert date
DATE_MODIFIED	date	Last update date
CREATED_BY	integer	By user ID
MODIFIED_BY	integer	By user ID

ADV_ATTACK_RPT_V

View references ADV_ATTACK table that stores Advisor attack information.

Column Name	Datatype	Comment
ALERT_ID	integer	
TRUSECURE_ATTACK_NAME	Varchar2(512)	
FEED_DATE_CREATED	date	
FEED_DATE_UPDATED	date	
ATTACK_CATEGORY	varchar2(256)	
URGENCY_ID	integer	
SEVERITY_ID	integer	
LOCAL	integer	
REMOTE	integer	
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
CREATED_BY	integer	By user ID

Column Name	Datatype	Comment
MODIFIED_BY	integer	By user ID

ADV_CREDIBILITY_RPT_V

View references ADV_CREDIBILITY table that stores Advisor credibility information.

Column Name	Datatype	Comment
CREDIBILITY_ID	integer	
CREDIBILITY_RATING	varchar2(64)	
CREDIBILITY_EXPLANATION	varchar2(512)	
DATE_CREATED	date	Insert date
DATE_MODIFIED	date	Last update date
CREATED_BY	integer	By user ID
MODIFIED_BY	integer	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(128)	
FEED_FILE	varchar2(256)	
BEGIN_DATE	date	
END_DATE	date	
FEED_INSERT	integer	
FEED_UPDATE	integer	
FEED_EXPIRE	integer	

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	integer	
VENDOR_ID	integer	
PRODUCT_CATEGORY_ID	integer	
PRODUCT_CATEGORY_NAME	varchar2(128)	
PRODUCT_TYPE_ID	integer	
PRODUCT_TYPE_NAME	varchar2(256)	
PRODUCT_NAME	varchar2(128)	
PRODUCT_DESCRIPTION	varchar2(512)	
FEED_DATE_CREATED	date	
FEED_DATE_UPDATED	date	
ACTIVE_FLAG	integer	
DATE_CREATED	date	Insert date
DATE_MODIFIED	date	Last update date
CREATED_BY	integer	By user ID
MODIFIED_BY	integer	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	integer	
VERSION_ID	integer	Contains the version ID number
SERVICE_PACK_NAME	varchar2(32)	
FEED_DATE_CREATED	date	
FEED_DATE_UPDATED	date	
ACTIVE_FLAG	integer	
BEGIN_EFFECTIVE_DATE	date	
END_EFFECTIVE_DATE	date	
DATE_CREATED	date	Insert date
DATE_MODIFIED	date	Last update date
CREATED_BY	integer	By user ID
MODIFIED_BY	integer	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	integer	Contains the version ID number
PRODUCT_ID	integer	
VERSION_NAME	varchar2(128)	
FEED_DATE_CREATED	date	
FEED_DATE_UPDATED	date	
ACTIVE_FLAG	integer	
DATE_CREATED	date	Insert date
DATE_MODIFIED	date	Last update date
CREATED_BY	integer	By user ID
MODIFIED_BY	integer	By user ID

ADV_SEVERITY_RPT_V

View references ADV_SEVERITY table that stores Advisor severity rating information.

Column Name	Datatype	Comment
SEVERITY_ID	integer	
SEVERITY_RATING	varchar(64)	
SEVERITY_EXPLANATION	varchar2(512)	
DATE_CREATED	date	Insert date
DATE_MODIFIED	date	Last update date
CREATED_BY	integer	By user ID
MODIFIED_BY	integer	By user ID

ADV_SUBALERT_RPT_V

View references ADV_SUBALERT table.

Column Name	Datatype	Comment
ALERT_ID	integer	
SUBALERT_ID	integer	
CHANGED_SECTIONS	varchar2(1024)	
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	integer	By user ID
MODIFIED_BY	integer	By user ID

ADV_URGENCY_RPT_V

View references ADV_URGENCY table.

Column Name	Datatype	Comment
URGENCY_ID	integer	
URGENCY_RATING	varchar2(64)	
URGENCY_EXPLANATION	varchar2(512)	
DATE_CREATED	date	Insert date
DATE_MODIFIED	date	Last update date
CREATED_BY	integer	By user ID
MODIFIED_BY	integer	By user ID

ADV_VENDOR_RPT_V

View references ADV_VENDOR table that stores Advisor address information.

Column Name	Datatype	Comment
VENDOR_ID	integer	
VENDOR_NAME	varchar2(128)	
CONTACT_PERSON	varchar2(128)	
ADDRESS_LINE_1	varchar2(128)	
ADDRESS_LINE_2	varchar2(128)	
ADDRESS_LINE_3	varchar2(128)	
ADDRESS_LINE_4	varchar2(128)	
CITY	varchar2(128)	
STATE	varchar2(128)	
COUNTRY	varchar2(128)	
ZIP_CODE	varchar2(128)	
URL	varchar2(256)	
PHONE	varchar2(32)	

Column Name	Datatype	Comment
FAX	varchar2(32)	
EMAIL	varchar2(128)	
PAGER	varchar2(32)	
FEED_DATE_CREATED	date	
FEED_DATE_UPDATED	date	
ACTIVE_FLAG	integer	
DATE_CREATED	date	Insert date
DATE_MODIFIED	date	Last update date
CREATED_BY	integer	By user ID
MODIFIED_BY	integer	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	integer	
SERVICE_PACK_ID	integer	
DATE_CREATED	date	Insert date
DATE_MODIFIED	date	Last update date
CREATED_BY	integer	By user ID
MODIFIED_BY	integer	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	integer	Annotation identifier - sequence number.
TEXT	varchar2(4000)	Documentation or notes.
DATE_CREATED	date	Insert date
DATE_MODIFIED	date	Last update date
MODIFIED_BY	integer	Last updating user ID
CREATED_BY	integer	Inserting user ID
ACTION	varchar2(255)	Action

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
CUST_ID	integer	Customer identifier
DATE_CREATED	date	Last update date
DATE_MODIFIED	date	Last updating user ID
CREATED_BY	integer	Inserting user ID
MODIFIED_BY	integer	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	integer	Asset IP address
CUST_ID	integer	Customer identifier
DATE_CREATED	date	Insert date
DATE_MODIFIED	date	Last update date
CREATED_BY	integer	Inserting user ID
MODIFIED_BY	integer	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	integer	Location identifier
CUST_ID	integer	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
DATE_MODIFIED	date	Last update date
CREATED_BY	integer	Inserting user ID
MODIFIED_BY	integer	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
CUST_ID	integer	Customer identifier
ASSET_NAME	varchar2(255)	Asset name
PHYSICAL_ASSET_ID	varchar2(36)	Physical asset identifier
PRODUCT_ID	integer	Product identifier
ASSET_CATEGORY_ID	integer	Asset category identifier
ENVIRONMENT_IDENTITY_ID	integer	Environment identify code
PHYSICAL_ASSET_IND	integer(1)	Physical asset indicator
ASSET_VALUE_ID	integer	Asset value code
CRITICALITY_ID	integer	Asset criticality code
SENSITIVITY_ID	integer	Asset sensitivity code
DATE_CREATED	date	Insert date
DATE_MODIFIED	date	Last update date

Column Name	Datatype	Comment
CREATED_BY	integer	Inserting user ID
MODIFIED_BY	integer	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_ID	integer	Asset value code
ASSET_VALUE_NAME	varchar2(50)	Asset value name
DATE_CREATED	date	Insert date
DATE_MODIFIED	date	Last update date
CREATED_BY	integer	Inserting user ID
MODIFIED_BY	integer	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_SEQUENCE	integer	Order of persons under a particular role
DATE_CREATED	date	Insert date
DATE_MODIFIED	date	Last update date
CREATED_BY	integer	Inserting user ID
MODIFIED_BY	integer	Last updating user

ASSOCIATIONS_RPT_V

View references ASSOCIATIONS table that associates users to incidents, incidents to annotations and so on.

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
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(36)	Event Universal Unique Identifier (UUID) of parent event
CHILD_EVT_ID	varchar2(36)	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	integer	Inserting user ID
MODIFIED_BY	integer	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(36)	Event Universal Unique Identifier (UUID) of parent event
CHILD_EVT_ID	varchar2(36)	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	integer	Inserting user ID
MODIFIED_BY	integer	Last updating user ID

CRITICALITY_RPT_V

View references CRIT_LKUP table that contains information about asset criticality.

Column Name	Datatype	Comment
CRITICALITY_ID	integer	Asset criticality code
CRITICALITY_NAME	varchar2(50)	Asset criticality name
DATE_CREATED	date	Insert date
DATE_MODIFIED	date	Last update date
CREATED_BY	integer	By user ID
MODIFIED_BY	integer	By user ID

CUST_RPT_V

View references CUST table that stores customer information for MSSPs.

Column Name	Datatype	Comment
CUST_ID	integer	Customer identifier
CUSTOMER_NAME	varchar2(255)	Customer name
DATE_CREATED	date	Insert date
DATE_MODIFIED	date	Last update date
CREATED_BY	integer	Inserting user ID
MODIFIED_BY	integer	Last updating user ID

CUST_HIERARCHY_V

View references CUST_HIERARCHY table that stores information about MSSP customer hierarchy.

Column Name	Datatype	Comment
CUST_HIERARCHY_ID	integer	Customer hierarchy ID
CUST_NAME	varchar2(255)	Customer
CUST_HIERARCHY_LVL1	varchar2(255)	Customer hierarchy level 1
CUST_HIERARCHY_LVL2	varchar2(255)	Customer hierarchy level 2
CUST_HIERARCHY_LVL3	varchar2(255)	Customer hierarchy level 3
CUST_HIERARCHY_LVL4	varchar2(255)	Customer hierarchy 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

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	integer	Inserting by user ID
MODIFIED_BY	integer	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_ID	integer	Environment identity code
ENVIRONMENT_IDENTITY_NAME	varchar2(255)	Environment identity name
DATE_CREATED	date	Insert date
DATE_MODIFIED	date	Last update date
CREATED_BY	integer	Inserting user by ID
MODIFIED_BY	integer	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	integer	Position of tag within display.
WIDTH	integer	The column width

Column Name	Datatype	Comment
ALIGNMENT	integer	The horizontal alignment
FORMAT	integer	The enumerated formatter for displaying the property
ENABLED	varchar2(1)	Indicates if the tag is shown.
TYPE	integer	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	integer	Inserting user id.
MODIFIED_BY	integer	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	integer	Per http://www.iana.org/assignments/port-numbers, the numerical representation of the port. This port integer is typically associated with the Transport Protocol level in the TCP/IP stack.
PROTOCOL_NUMBER	integer	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	integer	Inserting User ID.
MODIFIED_BY	integer	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	integer	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.
PROTOCOL_DESCRIPTION	varchar2(512)	IP packet protocol description.
DATE_CREATED	date	Insert date.
DATE_MODIFIED	date	Last update date.
CREATED_BY	integer	Inserting User ID.
MODIFIED_BY	integer	Last updating User ID.

ESEC_SEQUENCE_RPT_V

View references ESEC_SEQUENCE table that's used to generate primary key sequence numbers for Sentinel tables.

Column Name	Datatype	Comment
TABLE_NAME	varchar2(32)	Name of the table.
COLUMN_NAME	varchar2(255)	Name of the column
SEED	integer	Current value of primary key field.
DATE_CREATED	date	Insert date.
DATE_MODIFIED	date	Last update date.
CREATED_BY	integer	Inserting user ID.
MODIFIED_BY	integer	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(36)	Event identifier
RESOURCE_NAME	varchar2(255)	Resource name
SUB_RESOURCE	varchar2(255)	Subresource name
SEVERITY	integer	Event severity
EVENT_PARSE_TIME	date	Event time
EVENT_DATETIME	date	Event time
EVENT_DEVICE_TIME	date	
SENTINEL_PROCESS_TIME	date	
BEGIN_TIME	date	
END_TIME	date	
REPEAT_COUNT	integer	
DESTINATION_PORT_INT	integer	
SOURCE_PORT_INT	integer	

Column Name	Datatype	Comment
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	integer	Source IP address in numeric format
SOURCE_HOST_NAME	varchar2(255)	Source host name
SOURCE_PORT	varchar2(32)	Source port
DESTINATION_IP	integer	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)	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	integer	Customer Tag 3
RESERVED_TAG_1	VARCHAR2(255)	Reserved Tag 1 Reserved for future use by Novell. This field is used for Advisor information concerning attack descriptions.
RESERVED_TAG_2	varchar2(255)	Reserved for future use by Novell. Use of this field for any other purpose may result in data being overwritten by future functionality.
RESERVED_TAG_3	integer	Reserved for future use by Novell. Use of this field for any other purpose may result in data being overwritten by future functionality.
SOURCE_UUID	varchar2(36)	Source UUID
PORT	varchar2(64)	Collector port
AGENT	varchar2(64)	Collector name
VULNERABILITY_RATING	integer	Vulnerability rating
CRITICALITY_RATING	integer	Criticality rating
DATE_CREATED	date	Insert date

Column Name	Datatype	Comment
DATE_MODIFIED	date	Last update date
CREATED_BY	integer	By user ID
MODIFIED_BY	integer	By user ID
RV01 - 10	integer	Reserved Value 1 - 10 Reserved for future use by Novell. 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 Novell. Use of this field for any other purpose may result in data being overwritten by future functionality.
RV21 - 25	varchar2(36)	Reserved Value 21 - 25 Reserved for future use by Novell 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 Novell. 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.

Column Name	Datatype	Comment
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.
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 Novell. 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 Novell. 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 Novell. Use of this field for any other purpose may result in data being overwritten by future functionality.
RV50	varchar2(100)	Taxonomy level 1
RV51	varchar2(100)	Taxonomy level 2
RV52	varchar2(100)	Taxonomy level 3
RV53	varchar2(100)	Taxonomy level 4
CV01 - 10	integer	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 - 29	varchar2(255)	Custom Value 21 – 100 Reserved for use by Customer, typically for association of Business relevant data
CV30 - 34	varchar2(4000)	
CV35 - 100	varchar2(255)	

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(36)	Event identifier
RESOURCE_NAME	varchar2(255)	Resource name
SUB_RESOURCE	varchar2(255)	Subresource name
SEVERITY	integer	Event severity
EVENT_PARSE_TIME	date	Event time
EVENT_DATETIME	date	Event time
EVENT_DEVICE_TIME	date	
SENTINEL_PROCESS_TIME	date	
BEGIN_TIME	date	
END_TIME	date	
REPEAT_COUNT	integer	
DESTINATION_PORT_INT	integer	
SOURCE_PORT_INT	integer	
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
CUST_ID	integer	
SOURCE_ASSET_ID	integer	
DESTINATION_ASSET_ID	integer	
AGENT_ID	integer	Collector identifier
PROTOCOL_ID	integer	
ARCHIVE_ID	integer	
SOURCE_IP	integer	Source IP address in numeric format
SOURCE_HOST_NAME	varchar2(255)	Source host name
SOURCE_PORT	varchar2(32)	Source port
DESTINATION_IP	integer	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	integer	Customer Tag 3

Column Name	Datatype	Comment
RESERVED_TAG_1	varchar2(255)	Reserved Tag 1 Reserved for future use by Novell. This field is used for Advisor information concerning attack descriptions.
RESERVED_TAG_2	varchar2(255)	Reserved for future use by Novell. Use of this field for any other purpose may result in data being overwritten by future functionality.
RESERVED_TAG_3	integer	Reserved for future use by Novell. Use of this field for any other purpose may result in data being overwritten by future functionality.
VULNERABILITY_RATING	integer	Vulnerability rating
CRITICALITY_RATING	integer	Criticality rating
DATE_CREATED	date	Insert date
DATE_MODIFIED	date	Last update date
CREATED_BY	integer	Inserting user ID.
MODIFIED_BY	integer	Last updating user ID.
RV01 - 10	integer	Reserved Value 1 - 10 Reserved for future use by Novell. 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 Novell. Use of this field for any other purpose may result in data being overwritten by future functionality.
RV21 - 25	varchar2(36)	Reserved Value 21 - 25 Reserved for future use by Novell 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 Novell. 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.

Column Name	Datatype	Comment
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 Novell. 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.

Column Name	Datatype	Comment
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 Novell. 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 Novell. Use of this field for any other purpose may result in data being overwritten by future functionality.
TAXONOMY_ID	integer	
REFERENCE_ID_01 - 20	integer	Reserved for future use by Novell. Use of this field for any other purpose may result in data being overwritten by future functionality.
CV01 - 10	integer	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 - 29	varchar2(255)	Custom Value 21 - 100Reserved for use by Customer, typically for association of Business relevant data
CV30 - 34	varchar2(4000)	
CV35 – 100	varchar2(255)	

EVT_AGENT_RPT_V

View references EVT_AGENT table that stores information about Collectors.

Column Name	Datatype	Comment
AGENT_ID	integer	Collector identifier

Column Name	Datatype	Comment
CUST_ID	integer	
AGENT	varchar2(64)	Collector name
PORT	varchar2(64)	Collector 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_CATEGORY	varchar2(255)	Device category
SOURCE_UUID	varchar2(36)	Source component Universal Unique Identifier (UUID)
DATE_CREATED	date	Insert date
DATE_MODIFIED	date	Last update date
CREATED_BY	integer	Inserting user ID
MODIFIED_BY	integer	Last updating user ID

EVT_ASSET_RPT_V

View references EVT_ASSET table that stores asset information.

Column Name	Datatype	Comment
EVENT_ASSET_ID	integer	Event asset identifier
CUST_ID	integer	
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.
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

Column Name	Datatype	Comment
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	integer	Inserting user ID
MODIFIED_BY	integer	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	integer	Destination IP address
DESTINATION_EVENT_ASSET_ID	integer	Event asset identifier
TAXONOMY_ID	integer	Taxonomy identifier
EVENT_NAME_ID	integer	Event name identifier
SEVERITY	integer	Event severity
CUST_ID	integer	Customer identifier
EVENT_TIME	date	Event time
EVENT_COUNT	integer	Event count
DATE_CREATED	date	Insert date
DATE_MODIFIED	date	Last update date
CREATED_BY	integer	Inserting user ID
MODIFIED_BY	integer	Last updating user ID

EVT_DEST_SMRY_1_RPT_V

View contains event destination summary information.

Column Name	Datatype	Comment
DESTINATION_IP	integer	Destination IP address
DESTINATION_EVENT_ASSET_ID	integer	Event asset identifier
DESTINATION_PORT	varchar2(32)	Destination port
DESTINATION_USER_ID	integer	Destination user identifier
TAXONOMY_ID	integer	Taxonomy identifier
EVENT_NAME_ID	integer	Event name identifier
RESOURCE_ID	integer	Resource identifier
AGENT_ID	integer	Collector identifier
PROTOCOL_ID	integer	Protocol identifier
SEVERITY	integer	Event severity
CUST_ID	integer	Customer identifier
EVENT_TIME	date	Event time
EVENT_COUNT	integer	Event count
DATE_CREATED	date	Insert date
DATE_MODIFIED	date	Last update date
CREATED_BY	integer	Inserting user ID
MODIFIED_BY	integer	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	integer	Destination IP address
DESTINATION_EVENT_ASSET_ID	integer	Event asset identifier
TAXONOMY_ID	integer	Taxonomy identifier
SEVERITY	integer	Event severity
CUST_ID	integer	Customer identifier
EVENT_TIME	date	Event time
EVENT_COUNT	integer	Event count
DATE_CREATED	date	Insert date
DATE_MODIFIED	date	Last update date
CREATED_BY	integer	Inserting user ID
MODIFIED_BY	integer	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	integer	Event name identifier
EVENT_NAME	varchar2(255)	Event name
DATE_CREATED	date	Insert date
DATE_MODIFIED	date	Last update date
CREATED_BY	integer	Inserting user ID
MODIFIED_BY	integer	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	integer	Event severity
CUST_ID	integer	Customer identifier
EVENT_TIME	date	Event time
EVENT_COUNT	integer	Event count
DATE_CREATED	date	Insert date
DATE_MODIFIED	date	Last update date
CREATED_BY	integer	Inserting user ID
MODIFIED_BY	integer	Last updating user ID

EVT_PRTCL_RPT_V

View references EVT_PRTCL table that stores event protocol information.

Column Name	Datatype	Comment
PROTOCOL_ID	integer	Protocol identifier
PROTOCOL_NAME	varchar2(255)	Protocol name
DATE_CREATED	date	Insert date
DATE_MODIFIED	date	Last update date
CREATED_BY	integer	Inserting user ID

Column Name	Datatype	Comment
MODIFIED_BY	integer	Last updating user ID

EVT_RSRC_RPT_V

View references EVT_RSRC table that stores event resource information.

Column Name	Datatype	Comment
RESOURCE_ID	integer	Resource identifier
RESOURCE_NAME	varchar2(255)	Resource name
SUBRESOURCE_NAME	varchar2(255)	Subresource name
CUST_ID	integer	Customer Identifier
DATE_CREATED	date	Insert date
DATE_MODIFIED	date	Last update date
CREATED_BY	integer	Inserting user ID
MODIFIED_BY	integer	Last updating user ID

EVT_SEV_SMRY_1_RPT_V

View summarizes event count by severity and event time.

Column Name	Datatype	Comment
SEVERITY	integer	Event severity
CUST_ID	integer	Customer identifier
EVENT_TIME	date	Event time
EVENT_COUNT	integer	Event count
DATE_CREATED	date	Insert date
DATE_MODIFIED	date	Last update date
CREATED_BY	integer	Inserting user ID
MODIFIED_BY	integer	Last updating user ID

EVT_SRC_SMRY_1_RPT_V

View contains event source and destination summary information.

Column Name	Datatype	Comment
SOURCE_IP	integer	Source IP address
SOURCE_EVENT_ASSET_ID	integer	Source event asset identifier
SOURCE_PORT	varchar2(32)	Source port
SOURCE_USER_ID	integer	Source user identifier
TAXONOMY_ID	integer	Taxonomy identifier
EVENT_NAME_ID	integer	Event name identifier
RESOURCE_ID	integer	Resource identifier
AGENT_ID	integer	Collector identifier
PROTOCOL_ID	integer	Protocol identifier
SEVERITY	integer	Event severity
CUST_ID	integer	Customer identifier
EVENT_TIME	date	Event time
EVENT_COUNT	integer	Event count
DATE_CREATED	date	Insert date
DATE_MODIFIED	date	Last update date
CREATED_BY	integer	Inserting user ID
MODIFIED_BY	integer	Last updating user ID

EVT_TXNMY_RPT_V

View references EVT_TXNMY table that stores event taxonomy information.

Column Name	Datatype	Comment
TAXONOMY_ID	integer	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
DEVICE_CATEGORY	varchar2(255)	
DATE_CREATED	date	Insert date
DATE_MODIFIED	date	Last update date
CREATED_BY	integer	Inserting user ID
MODIFIED_BY	integer	Last updating user ID

EVT_USR_RPT_V

View references EVT_USR table that stores event user information.

Column Name	Datatype	Comment
USER_ID	integer	User identifier
USER_NAME	varchar2(255)	User name
CUST_ID	integer	Customer identifier
DATE_CREATED	date	Insert date
DATE_MODIFIED	date	Last update date
CREATED_BY	integer	Inserting user ID
MODIFIED_BY	integer	Last updating user ID

EXTERNAL_DATA_RPT_V

View references EXTERNAL_DATA table that stores external data.

Column Name	Datatype	Comment
EXTERNAL_DATA_ID	integer	External data identifier
SOURCE_NAME	varchar2(50)	Source name
SOURCE_DATA_ID	varchar2(255)	Source data identifier
EXTERNAL_DATA	clob	External data
EXTERNAL_DATA_TYPE	varchar2(10)	External data type
DATE_CREATED	date	Insert date
DATE_MODIFIED	date	Last update date
CREATED_BY	integer	Inserting user ID
MODIFIED_BY	integer	Last updating user ID

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	integer	Inserting user ID

Column Name	Datatype	Comment
MODIFIED_BY	integer	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	integer	Incident identifier – sequence number
ASSET_ID	varchar2(36)	Asset Universal Unique Identifier (UUID)
DATE_CREATED	date	Insert date
DATE_MODIFIED	date	Last update date
CREATED_BY	integer	Inserting user ID
MODIFIED_BY	integer	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	integer	Incident identifier – sequence number
EVT_ID	varchar2(36)	Event Universal Unique Identifier (UUID)
EVT_TIME	date	Event time
DATE_CREATED	date	Insert date
DATE_MODIFIED	date	Last update date
CREATED_BY	integer	Inserting user ID
MODIFIED_BY	integer	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	integer	Incident identifier – sequence number
NAME	varchar2(255)	Incident name
SEVERITY	integer	Incident severity
STT_ID	integer	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 Novell. 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 Novell. Use of this field for any other purpose may result in data being overwritten by future functionality.
DATE_CREATED	date	Insert date

Column Name	Datatype	Comment
DATE_MODIFIED	date	Last update date
CREATED_BY	integer	Inserting user ID
MODIFIED_BY	integer	Last updating user ID
INC_DESC	varchar2(4000)	Incident description
INC_CAT	varchar2(255)	Incident category
INC_PRIORITY	integer	Incident priority
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	integer	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	integer	Inserting user ID
MODIFIED_BY	integer	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.

Column Name	Datatype	Comment
LOG_ID	integer	Sequence number
TIME	date	Date of Log
MODULE	varchar2(64)	Module log is for
TEXT	varchar2(4000)	Log text

MSSP_ASSOCIATIONS_V

View references MSSP_ASSOCIATIONS table that associates an integer key in one table to a uuid in another table.

Column Name	Datatype	Comment
TABLE1	varchar2(64)	Table name 1
ID1	integer	ID1
TABLE2	varchar2(64)	Table name 2
ID2	varchar2(36)	ID2
DATE_CREATED	date	Insert date
DATE_MODIFIED	date	Last update date

Column Name	Datatype	Comment
CREATED_BY	number	Inserting user ID
MODIFIED_BY	number	Last updating user ID

NETWORK_IDENTITY_RPT_V

View references NETWORK_IDENTITY_LKUP table that stores asset network identity information.

Column Name	Datatype	Comment
NETWORK_IDENTITY_ID	integer	Network identity code
NETWORK_IDENTITY_NAME	varchar2(255)	Network identify name
DATE_CREATED	date	Insert date
DATE_MODIFIED	date	Last update date
CREATED_BY	integer	Inserting user ID
MODIFIED_BY	integer	Last updating user ID

ORGANIZATION_RPT_V

View references ORGANIZATION table that stores organization (asset) information.

Column Name	Datatype	Comment
ORGANIZATION_ID	varchar2(36)	Organization identifier
ORGANIZATION_NAME	varchar2(100)	Organization name
CUST_ID	integer	Customer identifier
DATE_CREATED	date	Insert date
DATE_MODIFIED	date	Last update date
CREATED_BY	integer	Inserting user ID
MODIFIED_BY	integer	Last updating user ID

PERSON_RPT_V

View references PERSION table that stores personal (asset) information.

Column Name	Datatype	Comment
PERSON_ID	varchar2(36)	Person identifier
FIRST_NAME	varchar2(255)	First name
LAST_NAME	varchar2(255)	Last name
CUST_ID	integer	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	integer	Inserting user ID
MODIFIED_BY	integer	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(36)	Physical asset identifier
CUST_ID	integer	Customer identifier
HOST_NAME	varchar2(255)	Host name

Column Name	Datatype	Comment
IP_ADDRESS	integer	IP address
LOCATION_ID	integer	Location identifier
NETWORK_IDENTITY_ID	integer	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	integer	Inserting user ID
MODIFIED_BY	integer	Last updating user ID

PRODUCT_RPT_V

View references PRDT table that stores asset product information.

Column Name	Datatype	Comment
PRODUCT_ID	integer	Product identifier
PRODUCT_NAME	varchar2(255)	Product name
PRODUCT_VERSION	varchar2(100)	Product version
VENDOR_ID	integer	Vendor identifier
DATE_CREATED	date	Insert date
DATE_MODIFIED	date	Last update date
CREATED_BY	integer	Inserting user ID
MODIFIED_BY	integer	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(15)	Role code
ROLE_NAME	varchar2(255)	Role name
DATE_CREATED	date	Insert date
DATE_MODIFIED	date	Last update date
CREATED_BY	integer	Inserting user ID
MODIFIED_BY	integer	Last updating user ID

SENSITIVITY_RPT_V

View references SENSITIVITY_LKUP table that stores asset sensitivity information.

Column Name	Datatype	Comment
SENSITIVITY_ID	integer	Asset sensitivity code
SENSITIVITY_NAME	varchar2(50)	Asset sensitivity name
DATE_CREATED	date	Insert date
DATE_MODIFIED	date	Last update date
CREATED_BY	integer	By user ID
MODIFIED_BY	integer	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	integer	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	integer	Inserting user ID
CREATED_BY	integer	Last updating user ID

UNASSIGNED_INCIDENTS_RPT_V

View references CASES and INCIDENTS tables to report on unassigned cases.

Name	Datatype
INC_ID	integer
NAME	varchar2(255)
SEVERITY	integer
STT_ID	integer
SEVERITY_RATING	varchar2(32)
VULNERABILITY_RATING	varchar2(32)
CRITICALITY_RATING	varchar2(32)
DATE_CREATED	date
DATE_MODIFIED	date
CREATED_BY	integer
MODIFIED_BY	integer
INC_DESC	varchar2(4000)
INC_PRIORITY	integer
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	integer	User identifier – Sequence integer
NAME	varchar2(64)	Short, unique user name used as a login
CNT_ID	integer	Contact ID – Sequence integer
STT_ID	integer	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	integer	Inserting user ID
MODIFIED_BY	integer	Last updating user ID
PERMISSIONS	varchar2(4000)	Permissions currently assigned to the

Column Name	Datatype	Comment
		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	integer	Domain authentication indication

VENDOR_RPT_V

View references VNDR table that stores information about asset product vendors.

Column Name	Datatype	Comment
VENDOR_ID	integer	Vendor identifier
VENDOR_NAME	varchar2(255)	Vendor name
DATE_CREATED	date	Insert date
DATE_MODIFIED	date	Last update date
CREATED_BY	integer	Inserting user ID
MODIFIED_BY	integer	Last updating user ID

VULN_CALC_SEVERITY_RPT_V

View references VULN_RSRC and VULN to calculate Sentinel vulnerability severity rating base on current vulnerabilities.

Column Name	Datatype
RSRC_ID	varchar2(36)
IP	varchar2(32)
HOST_NAME	varchar2(255)
CRITICALITY	number
ASSIGNED_VULN_SEVERITY	number
VULN_COUNT	number
CALC_SEVERITY	number

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

Column Name	Datatype
VULN_INFO_ID	varchar2(36)
VULN_ID	varchar2(36)
VULN_INFO_TYPE	varchar2(36)
VULN_INFO_VALUE	varchar2(4000)
DATE_CREATED	date
DATE_MODIFIED	date
CREATED_BY	integer
MODIFIED_BY	integer

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)

Column Name	Datatype
IP	varchar2(32)
HOST_NAME	varchar2(255)
LOCATION	varchar2(128)
DEPARTMENT	varchar2(128)
BUSINESS_SYSTEM	varchar2(128)
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)
LOAD_STATUS	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

Column Name	Datatype
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

10 Sentinel Database Views for Microsoft SQL Server

This chapter lists the Sentinel 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\nvarchar(13)	
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\nvarchar(128)	
PRODUCT	varchar\nvarchar(128)	
VERSION	varchar\nvarchar(128)	Contains the version number
SERVICE_PACK	varchar\nvarchar(128)	
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
TEMPLATE_ID	int	
TEMPLATE_NAME	varchar\nvarchar(256)	
THREAT_CATEGORY_NAME	varchar\nvarchar(128)	
THREAT_TYPE_NAME	varchar\nvarchar(128)	
HEADLINE	ntext	
FIRST_PUBLISHED	datetime	
LAST_PUBLISHED	datetime	
STATUS	varchar\nvarchar(32)	
URGENCY_ID	int	
CREDIBILITY_ID	int	
SEVERITY_ID	int	
SUMMARY	ntext	
LEGAL_DISCLAIMER	ntext	
COPYRIGHT	varchar\nvarchar(1024)	
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	

Column Name	Datatype	Comment
CVE	varchar\nvarchar(13)	
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\nvarchar(256)	
ATTACK_CODE	varchar\nvarchar(256)	
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\nvarchar(256)	
PLUGIN_NAME	varchar\nvarchar(256)	
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	varchar\nvarchar(512)	
FEED_DATE_CREATED	datetime	
FEED_DATE_UPDATED	datetime	
ATTACK_CATEGORY	varchar\nvarchar(256)	
URGENCY_ID	int	
SEVERITY_ID	int	
LOCAL	int	

Column Name	Datatype	Comment
REMOTE	int	
BEGIN_EFFECTIVE_DATE	datetime	
END_EFFECTIVE_DATE	datetime	
DESCRIPTION	ntext	
SCENARIO	ntext	
IMPACT	ntext	
SAFEGUARDS	ntext	
PATCHES	ntext	
FALSE_POSITIVES	ntext	
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\nvarchar(64)	
CREDIBILITY_EXPLANATION	varchar\nvarchar(512)	
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\nvarchar(128)	
FEED_FILE	varchar\nvarchar(256)	
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	

Column Name	Datatype	Comment
VENDOR_ID	int	
PRODUCT_CATEGORY_ID	int	
PRODUCT_CATEGORY_NAME	varchar\nvarchar (128)	
PRODUCT_TYPE_ID	int	
PRODUCT_TYPE_NAME	varchar\nvarchar (256)	
PRODUCT_NAME	varchar\nvarchar (128)	
PRODUCT_DESCRIPTION	varchar\nvarchar (512)	
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_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\nvar char (32)	
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\nvarch ar (128)	
FEED_DATE_CREATED	datetime	
FEED_DATE_UPDATED	datetime	
ACTIVE_FLAG	int	

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_SEVERITY_RPT_V

View references ADV_SEVERITY table that stores Advisor severity rating information.

Column Name	Datatype	Comment
SEVERITY_ID	int	
SEVERITY_RATING	varchar\nvarchar (64)	
SEVERITY_EXPLANATION	varchar\nvarchar (512)	
DATE_CREATED	datetime	Insert date
DATE_MODIFIED	datetime	Last update date
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\nvarchar (1024)	
VARIANTS	ntext	
VIRUS_NAME	ntext	
DESCRIPTION	ntext	
IMPACT	ntext	
WARNING_INDICATORS	ntext	
TECHNICAL_INFO	ntext	
TRUSECURE_COMMENTS	ntext	
VENDOR_ANNOUNCEMENTS	ntext	
SAFEGUARDS	ntext	
PATCHES_SOFTWARE	ntext	
ALERT_HISTORY	ntext	
BACKGROUND_INFO	ntext	
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	

Column Name	Datatype	Comment
URGENCY_RATING	varchar\nvarchar (64)	
URGENCY_EXPLANATION	varchar\nvarchar (512)	
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\nvarchar (128)	
CONTACT_PERSON	varchar\nvarchar (128)	
ADDRESS_LINE_1	varchar\nvarchar (128)	
ADDRESS_LINE_2	varchar\nvarchar (128)	
ADDRESS_LINE_3	varchar\nvarchar (128)	
ADDRESS_LINE_4	varchar\nvarchar (128)	
CITY	varchar\nvarchar (128)	
STATE	varchar\nvarchar (128)	
COUNTRY	varchar\nvarchar (128)	
ZIP_CODE	varchar\nvarchar (128)	
URL	varchar\nvarchar (256)	
PHONE	varchar\nvarchar (32)	
FAX	varchar\nvarchar (32)	
EMAIL	varchar\nvarchar (128)	
PAGER	varchar\nvarchar (32)	
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

Column Name	Datatype	Comment
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\nvarchar(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\nvarchar(255)	Action

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\nvarchar(255)	Host name
CUST_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

Column Name	Datatype	Comment
CUST_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
CUST_ID	bigint	Customer identifier
BUILDING_NAME	varchar\nvarchar(255)	Building name
ADDRESS_LINE_1	varchar\nvarchar(255)	Address line 1
ADDRESS_LINE_2	varchar\nvarchar(255)	Address line 2
CITY	varchar\nvarchar(100)	City
STATE	varchar\nvarchar(100)	State
COUNTRY	varchar\nvarchar(100)	Country
ZIP_CODE	varchar\nvarchar(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
CUST_ID	bigint	Customer identifier
ASSET_NAME	varchar\nvarchar (255)	Asset name
PHYSICAL_ASSET_ID	uniqueidentifier	Physical asset identifier
PRODUCT_ID	bigint	Product identifier
ASSET_CATEGORY_ID	bigint	Asset category identifier
ENVIRONMENT_IDENTITY_ID	bigint	Environment identify code
PHYSICAL_ASSET_IND	bit	Physical asset indicator
ASSET_VALUE_ID	bigint	Asset value code
CRITICALITY_ID	bigint	Asset criticality code
SENSITIVITY_ID	bigint	Asset sensitivity code
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

ASSET_VALUE_RPT_V

View references ASSET_VAL_LKUP table that stores information about the asset value.

Column Name	Datatype	Comment
ASSET_VALUE_ID	bigint	Asset value code
ASSET_VALUE_NAME	varchar\nvarchar(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\nvarchar(5)	Role code
ASSET_ID	uniqueidentifier	Asset identifier
ENTITY_TYPE_CODE	varchar\nvarchar(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 and so on.

Column Name	Datatype	Comment
TABLE1	varchar\nvarchar(64)	Table name 1. For example, incidents
ID1	int	ID1. For example, incident ID.
TABLE2	varchar\nvarchar(64)	Table name 2. For example, users.
ID2	int	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\nvarchar(255)	Attachment name
SOURCE_REFERENCE	varchar\nvarchar(64)	Source reference
TYPE	varchar\nvarchar(32)	Attachment type
SUB_TYPE	varchar\nvarchar(32)	Attachment subtype
FILE_EXTENSION	varchar\nvarchar(32)	File extension
ATTACHMENT_DESCRIPTION	varchar\nvarchar(255)	Attachment description
DATA	ntext	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\nvarchar(32)	User name.
APPLICATION	varchar\nvarchar(255)	Application identifier
UNIT	varchar\nvarchar(64)	Application unit
VALUE	varchar\nvarchar(255)	Text value if any
DATA	ntext	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\nvarchar(20)	Contact first name.
LAST_NAME	varchar\nvarchar(30)	Contact last name.
TITLE	varchar\nvarchar(128)	Contact title
DEPARTMENT	varchar\nvarchar(128)	Department
PHONE	varchar\nvarchar(64)	Contact phone
EMAIL	varchar\nvarchar(255)	Contact email
PAGER	varchar\nvarchar(64)	Contact pager
CELL	varchar\nvarchar(64)	Contact cell phone
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

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	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

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_ID	bigint	Asset criticality code
CRITICALITY_NAME	varchar\nvarchar(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_HIERARCHY_V

View references CUST_HIERARCHY table that stores information about MSSP customer hierarchy.

Column Name	Datatype	Comment
CUST_HIERARCHY_ID	bigint	Customer hierarchy ID
CUST_NAME	varchar\nvarchar (255)	Customer
CUST_HIERARCHY_LVL1	varchar\nvarchar (255)	Customer hierarchy level 1
CUST_HIERARCHY_LVL2	varchar\nvarchar (255)	Customer hierarchy level 2
CUST_HIERARCHY_LVL3	varchar\nvarchar (255)	Customer hierarchy level 3
CUST_HIERARCHY_LVL4	varchar\nvarchar (255)	Customer hierarchy 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

CUST_RPT_V

View references CUST table that stores customer information for MSSPs.

Column Name	Datatype	Comment
CUST_ID	bigint	Customer identifier
CUSTOMER_NAME	varchar\nvarchar(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\nvarchar(5)	Entity type code
ENTITY_TYPE_NAME	varchar\nvarchar(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_ID	int	Environment identity code
ENVIRONMENT_IDENTITY_NAME	varchar\nvarchar(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\NVARCHAR(32)	The parent object of the property
TAG	VARCHAR\NVARCHAR(32)	The native tag name of the property
LABEL	VARCHAR\NVARCHAR(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	varchar\nvarchar(1)	Indicates if the tag is shown.
TYPE	INT	Indicates datatype of tag. 1 = string 2 = ulong 3 = date 4 = uuid 5 = ipv4
DESCRIPTION	VARCHAR\NVARCHAR(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\NVARCHAR(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.

Column Name	Datatype	Comment
PORT_KEYWORD	VARCHAR\NVARCHAR(64)	Per http://www.iana.org/assignments/port-numbers, the keyword representation of the port.
PORT_DESCRIPTION	VARCHAR\NVARCHAR(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\NVARCHAR(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\NVARCHAR(512)	IP packet protocol description.
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_SEQUENCE_RPT_V

View references ESEC_SEQUENCE table that's used to generate primary key sequence numbers for Sentinel tables.

Column Name	Datatype	Comment
TABLE_NAME	VARCHAR\NVARCHAR(32)	Name of the table.
COLUMN_NAME	VARCHAR\NVARCHAR(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\nvarchar(255)	Resource name
SUB_RESOURCE	varchar\nvarchar(255)	Subresource name
SEVERITY	int	Event severity
EVENT_PARSE_TIME	datetime	Event time
EVENT_DATETIME	datetime	Event time
SENTINEL_PROCESS_TIME	datetime	
BEGIN_TIME	datetime	
END_TIME	datetime	
REPEAT_COUNT	int	
DESTINATION_PORT_INT	int	
SOURCE_PORT_INT	int	
BASE_MESSAGE	varchar\nvarchar(4000)	Base message
EVENT_NAME	varchar\nvarchar(255)	Name of the event as reported by the sensor
EVENT_TIME	varchar\nvarchar(255)	Event time as reported by the sensor
SENSOR_NAME	varchar\nvarchar(255)	Sensor name
SENSOR_TYPE	varchar\nvarchar(5)	Sensor type: H – host-based N – network-based V – virus O – other
PROTOCOL	varchar\nvarchar(255)	Protocol name
SOURCE_IP	int	Source IP address in numeric format
SOURCE_HOST_NAME	varchar\nvarchar(255)	Source host name
SOURCE_PORT	varchar\nvarchar(32)	Source port
DESTINATION_IP	int	Destination IP address in numeric format
DESTINATION_HOST_NAME	varchar\nvarchar(255)	Destination host name
DESTINATION_PORT	varchar\nvarchar(32)	Destination port
SOURCE_USER_NAME	varchar\nvarchar(255)	Source user name
DESTINATION_USER_NAME	varchar\nvarchar(255)	Destination user name
FILE_NAME	varchar\nvarchar(1000)	File name
EXTENDED_INFO	varchar\nvarchar(1000)	Extened information
REPORT_NAME	varchar\nvarchar(255)	Reporter name
PRODUCT_NAME	varchar\nvarchar(255)	Reporting product name
CUSTOM_TAG_1	varchar\nvarchar(255)	Customer Tag 1
CUSTOM_TAG 2	varchar\nvarchar(255)	Customer Tag 2
CUSTOM_TAG 3	int	Customer Tag 3

Column Name	Datatype	Comment
RESERVED_TAG_1	VARCHAR\NVARCHAR(255)	Reserved Tag 1 Reserved for future use by Sentinel. This field is used for Advisor information concerning attack descriptions.
RESERVED_TAG_2	varchar\nvarchar(255)	Reserved for future use by Sentinel. 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 Sentinel. Use of this field for any other purpose may result in data being overwritten by future functionality.
SOURCE_UUID	uniqueidentifier	Source UUID
PORT	varchar\nvarchar(64)	Collector port
AGENT	varchar\nvarchar(64)	Collector 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 Sentinel. 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 Sentinel. 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 Sentinel to store UUIDs. Use of this field for any other purpose may result in data being overwritten by future functionality.
RV26 - 31	VARCHAR\NVARCHAR(255)	Reserved Value 26 - 31 Reserved for future use by Sentinel. Use of this field for any other purpose may result in data being overwritten by future functionality.

Column Name	Datatype	Comment
RV32	VARCHAR\NVARCHAR(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\NVARCHAR(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\NVARCHAR(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\NVARCHAR(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\NVARCHAR(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\NVARCHAR(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\NVARCHAR(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\NVARCHAR(255)	Reserved Value 39 Reserved for MSSPCustomerName. Use of this field for any other purpose may result in data being overwritten by future functionality.

Column Name	Datatype	Comment
RV40 - 43	VARCHAR\NVARCHAR(255)	Reserved Value 40 - 43 Reserved for future use by Sentinel. Use of this field for any other purpose may result in data being overwritten by future functionality.
RV44	VARCHAR\NVARCHAR(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\NVARCHAR(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\NVARCHAR(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\NVARCHAR(255)	Reserved Value 47 Reserved for future use by Sentinel. Use of this field for any other purpose may result in data being overwritten by future functionality.
RV48	VARCHAR\NVARCHAR(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\NVARCHAR(255)	Reserved Value 49 Reserved for future use by Sentinel. Use of this field for any other purpose may result in data being overwritten by future functionality.
RV50	VARCHAR\NVARCHAR(255)	Taxonomy level 1
RV51	VARCHAR\NVARCHAR(255)	Taxonomy level 2
RV52	VARCHAR\NVARCHAR(255)	Taxonomy level 3

Column Name	Datatype	Comment
RV53	VARCHAR\NVARCHAR(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\NVARCHAR(255)	Custom Value 21 – 100 Reserved 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\nvarchar(255)	Resource name
SUB_RESOURCE	varchar\nvarchar(255)	Subresource name
SEVERITY	int	Event severity
EVENT_PARSE_TIME	datetime	Event time
EVENT_DATETIME	datetime	Event time
BASE_MESSAGE	varchar\nvarchar(4000)	Base message
EVENT_NAME	varchar\nvarchar(255)	Name of the event as reported by the sensor
EVENT_TIME	varchar\nvarchar(255)	Event time as reported by the sensor
AGENT_ID	bigint	Collector identifier

Column Name	Datatype	Comment
SOURCE_IP	int	Source IP address in numeric format
SOURCE_HOST_NAME	varchar\nvarchar(255)	Source host name
SOURCE_PORT	varchar\nvarchar(32)	Source port
DESTINATION_IP	int	Destination IP address in numeric format
DESTINATION_HOST_NAME	varchar\nvarchar(255)	Destination host name
DESTINATION_PORT	varchar\nvarchar(32)	Destination port
SOURCE_USER_NAME	varchar\nvarchar(255)	Source user name
DESTINATION_USER_NAME	varchar\nvarchar(255)	Destination user name
FILE_NAME	varchar\nvarchar(1000)	File name
EXTENDED_INFO	varchar\nvarchar(1000)	Extened information
CUSTOM_TAG_1	varchar\nvarchar(255)	Customer Tag 1
CUSTOM_TAG 2	varchar\nvarchar(255)	Customer Tag 2
CUSTOM_TAG 3	int	Customer Tag 3
RESERVED_TAG_1	VARCHAR\NVARCHAR(255)	Reserved Tag 1 Reserved for future use by Sentinel. This field is used for Advisor information concerning attack descriptions.
RESERVED_TAG_2	varchar\nvarchar(255)	Reserved for future use by Sentinel. 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 Sentinel. 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 Sentinel. Use of this field for any other purpose may result in data being overwritten by future functionality.

Column Name	Datatype	Comment
RV11 - 20	DATETIME	Reserved Value 1 - 31 Reserved for future use by Sentinel. 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 Sentinel to store UUIDs. Use of this field for any other purpose may result in data being overwritten by future functionality.
RV26 - 31	VARCHAR\NVARCHAR(255)	Reserved Value 26 - 31 Reserved for future use by Sentinel. Use of this field for any other purpose may result in data being overwritten by future functionality.
RV33	VARCHAR\NVARCHAR(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\NVARCHAR(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\NVARCHAR(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\NVARCHAR(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\NVARCHAR(255)	Reserved Value 37 Reserved for SourceFunction. Use of this field for any other purpose may result in data being overwritten by future functionality.

Column Name	Datatype	Comment
RV38	VARCHAR\NVARCHAR(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\NVARCHAR(255)	Reserved Value 40 - 43 Reserved for future use by Sentinel. Use of this field for any other purpose may result in data being overwritten by future functionality.
RV44	VARCHAR\NVARCHAR(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\NVARCHAR(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\NVARCHAR(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\NVARCHAR(255)	Reserved Value 47 Reserved for future use by Sentinel. Use of this field for any other purpose may result in data being overwritten by future functionality.
RV48	VARCHAR\NVARCHAR(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\NVARCHAR(255)	Reserved Value 49 Reserved for future use by Sentinel. Use of this field for any other purpose may result in data being overwritten by future functionality.

Column Name	Datatype	Comment
REFERENCE_ID 01 - 20	BIGINT	Reserved for future use by Sentinel. 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\NVARCHAR(255)	Custom Value 21 – 100 Reserved for use by Customer, typically for association of Business relevant data

EVT_AGENT_RPT_V

View references EVT_AGENT table that stores information about Collectors.

Column Name	Datatype	Comment
AGENT_ID	bigint	Collector identifier
AGENT	varchar\nvarchar(64)	Collector name
PORT	varchar\nvarchar(64)	Collector port
REPORT_NAME	varchar\nvarchar(255)	Reporter name
PRODUCT_NAME	varchar\nvarchar(255)	Product name
SENSOR_NAME	varchar\nvarchar(255)	Sensor name
SENSOR_TYPE	varchar\nvarchar(5)	Sensor type: H - host-based N - network-based V - virus O - other
DEVICE_CTGRY	varchar\nvarchar(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

Column Name	Datatype	Comment
ASSET_NAME	varchar\nvarchar(255)	Asset name
PHYSICAL_ASSET_NAME	varchar\nvarchar(255)	Physical asset name
REFERENCE_ASSET_ID	varchar\nvarchar(100)	Reference asset identifier, links to source asset management system.
MAC_ADDRESS	varchar\nvarchar(100)	MAC address
RACK_NUMBER	varchar\nvarchar(50)	Rack number
ROOM_NAME	varchar\nvarchar(100)	Room name
BUILDING_NAME	varchar\nvarchar(255)	Building name
CITY	varchar\nvarchar(100)	City
STATE	varchar\nvarchar(100)	State
COUNTRY	varchar\nvarchar(100)	Country
ZIP_CODE	varchar\nvarchar(50)	Zip code
ASSET_CATEGORY_NAME	varchar\nvarchar(100)	Asset category name
NETWORK_IDENTITY_NAME	varchar\nvarchar(255)	Asset network identity name
ENVIRONMENT_IDENTITY_NAME	varchar\nvarchar(255)	Environment name
ASSET_VALUE_NAME	varchar\nvarchar(50)	Asset value name
CRITICALITY_NAME	varchar\nvarchar(50)	Asset criticality name
SENSITIVITY_NAME	varchar\nvarchar(50)	Asset sensitivity name
CONTACT_NAME_1	varchar\nvarchar(255)	Name of contact person/organization 1
CONTACT_NAME_2	varchar\nvarchar(255)	Name of contact person/organization 2
ORGANIZATION_NAME_1	varchar\nvarchar(100)	Asset owner organization level 1
ORGANIZATION_NAME_2	varchar\nvarchar(100)	Asset owner organization level 2
ORGANIZATION_NAME_3	varchar\nvarchar(100)	Asset owner organization level 3
ORGANIZATION_NAME_4	varchar\nvarchar(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
CUST_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\nvarchar(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	Collector identifier
PROTOCOL_ID	bigint	Protocol identifier
SEVERITY	int	Event severity
CUST_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
CUST_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_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\nvarchar(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\nvarchar(32)	Destination port
SEVERITY	int	Event severity
CUST_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_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\nvarchar(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\nvarchar(255)	Resource name

Column Name	Datatype	Comment
SUB_RESOURCE_NAME	varchar\nvarchar(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
CUST_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\nvarchar(32)	Source port
SOURCE_USER_ID	bigint	User identifier
TAXONOMY _ID	bigint	Taxonomy identifier
EVENT_NAME_ID	bigint	Event name identifier
RESOURCE_ID	bigint	Resource identifier
AGENT_ID	bigint	Collector identifier
PROTOCOL _ID	bigint	Protocol identifier
SEVERITY	int	Event severity
CUST_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\nvarchar(100)	Taxonomy level 1

Column Name	Datatype	Comment
TAXONOMY _ LEVEL _2	varchar\nvarchar(100)	Taxonomy level 2
TAXONOMY _ LEVEL _3	varchar\nvarchar(100)	Taxonomy level 3
TAXONOMY _ LEVEL _4	varchar\nvarchar(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\nvarchar(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\nvarchar(50)	Source name
SOURCE_DATA_ID	varchar\nvarchar(255)	Source data identifier
EXTERNAL_DATA	ntext	External data
EXTERNAL_DATA_TYPE	varchar\nvarchar(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\NVARCHAR(128)	Image name
TYPE	VARCHAR\NVARCHAR(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)
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\NVARCHAR(255)	Incident name
SEVERITY	INT	Incident severity
STT_ID	INT	Incident State ID
SEVERITY_RATING	VARCHAR\NVARCHAR(32)	Average of all the event severities that comprise an incident.

Column Name	Datatype	Comment
VULNERABILITY_RATING	VARCHAR\NVARCHAR(32)	Reserved for future use by Sentinel. Use of this field for any other purpose may result in data being overwritten by future functionality.
CRITICALITY_RATING	VARCHAR\NVARCHAR(32)	Reserved for future use by Sentinel. 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\nvarchar(4000)	Incident description
INC_PRIORITY	int	Incident priority
INC_RES	varchar\nvarchar(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
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\NVARCHAR(32)	Resource name
STATS_NAME	VARCHAR\NVARCHAR(32)	Statistic name
STATS_VALUE	VARCHAR\NVARCHAR(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.

Column Name	Datatype	Comment
LOG_ID	NUMBER	Sequence number
TIME	DATE	Date of Log
MODULE	VARCHAR\NVARCHA	Module log is for

Column Name	Datatype	Comment
	R(64)	
TEXT	VARCHAR\NVARCHAR(4000)	Log ntext

MSSP_ASSOCIATIONS_V

View references MSSP_ASSOCIATIONS table that associates an integer key in one table to a uuid in another table.

Column Name	Datatype	Comment
TABLE1	varchar\nvarchar (64)	Table name 1
ID1	bigint	ID1
TABLE2	varchar\nvarchar (64)	Table name 2
ID2	uniqueidentifier	ID2
DATE_CREATED	datetime	Insert date
DATE_MODIFIED	datetime	Last update date
CREATED_BY	int	Inserting user ID
MODIFIED_BY	int	Last updating user ID

NETWORK_IDENTITY_RPT_V

View references NETWORK_IDENTITY_LKUP table that stores asset network identity information.

Column Name	Datatype	Comment
NETWORK_IDENTITY_CD	varchar\nvarchar(5)	Network identity code
NETWORK_IDENTITY_NAME	varchar\nvarchar(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\nvarchar(100)	Organization name
CUST_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

Column Name	Datatype	Comment
PERSON_ID	uniqueidentifier	Person identifier
FIRST_NAME	varchar\nvarchar(255)	First name
LAST_NAME	varchar\nvarchar(255)	Last name
CUST_ID	bigint	Customer identifier
PHONE_NUMBER	varchar\nvarchar(50)	Phone number
EMAIL_ADDRESS	varchar\nvarchar(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
CUST_ID	int	Customer identifier
LOCATION_ID	bigint	Location identifier
HOST_NAME	varchar\nvarchar(255)	Host name
IP_ADDRESS	int	IP address
NETWORK_IDENTITY_CD	varchar\nvarchar(5)	Network identity code
MAC_ADDRESS	varchar\nvarchar(100)	MAC address
RACK_NUMBER	varchar\nvarchar(50)	Rack number
ROOM_NAME	varchar\nvarchar(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\nvarchar(255)	Product name
PRODUCT _VERSION	varchar\nvarchar(100)	Product version
VENDOR _ID	bigint	Vendor identifier
DATE_CREATED	datetime	Insert date
DATE_MODIFIED	datetime	Last update date

Column Name	Datatype	Comment
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\nvarchar(5)	Role code
ROLE_NAME	varchar\nvarchar(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_ID	varchar\nvarchar(5)	Asset sensitivity code
SENSITIVITY_NAME	varchar\nvarchar(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\NVARCHAR(64)	Context of the state. That is case, incident, user.
NAME	VARCHAR\NVARCHAR(64)	Name of the state.
TERMINAL_FLAG	VARCHAR\NVARCHAR(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\NVARCHAR(255)
SEVERITY	INT
STT_ID	INT
SEVERITY_RATING	VARCHAR\NVARCHAR(32)
VULNERABILITY_RATING	VARCHAR\NVARCHAR(32)
CRITICALITY_RATING	VARCHAR\NVARCHAR(32)
DATE_CREATED	DATETIME
DATE_MODIFIED	DATETIME
CREATED_BY	INT
MODIFIED_BY	INT
INC_DESC	VARCHAR\NVARCHAR(4000)
INC_PRIORITY	INT
INC_CAT	VARCHAR\NVARCHAR(255)
INC_RES	VARCHAR\NVARCHAR(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\NVARCHAR(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\NVARCHAR(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\NVARCHAR(4000)	Permissions currently assigned to the Sentinel user
FILTER	VARCHAR\NVARCHAR(128)	Current security filter assigned to the Sentinel user
UPPER_NAME	VARCHAR\NVARCHAR(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\nvarchar(255)	Vendor name
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

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
IP	VARCHAR\NVARCHAR(32)
HOST_NAME	VARCHAR\NVARCHAR(255)
CRITICALITY	int
ASSIGNED_VULN_SEVERITY	int
VULN_COUNT	int
CALC_SEVERITY	int

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\NVARCHAR(36)
VULN_ID	VARCHAR\NVARCHAR(36)
VULN_CODE_TYPE	VARCHAR\NVARCHAR(64)
VULN_CODE_VALUE	VARCHAR\NVARCHAR(255)
URL	VARCHAR\NVARCHAR(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\NVARCHAR(36)
VULN_ID	VARCHAR\NVARCHAR(36)
VULN_INFO_TYPE	VARCHAR\NVARCHAR(36)
VULN_INFO_VALUE	VARCHAR\NVARCHAR(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\NVARCHAR(36)

Column Name	Datatype
RSRC_ID	VARCHAR\NVARCHAR(36)
PORT_NAME	VARCHAR\NVARCHAR(64)
PORT_NUMBER	INT
NETWORK_PROTOCOL	INT
APPLICATION_PROTOCOL	VARCHAR\NVARCHAR(64)
ASSIGNED_VULN_SEVERITY	INT
COMPUTED_VULN_SEVERITY	INT
VULN_DESCRIPTION	CLOB
VULN_SOLUTION	CLOB
VULN_SUMMARY	VARCHAR\NVARCHAR(1000)
BEGIN_EFFECTIVE_DATE	DATETIME
END_EFFECTIVE_DATE	DATETIME
DETECTED_OS	VARCHAR\NVARCHAR(64)
DETECTED_OS_VERSION	VARCHAR\NVARCHAR(64)
SCANNED_APP	VARCHAR\NVARCHAR(64)
SCANNED_APP_VERSION	VARCHAR\NVARCHAR(64)
VULN_USER_NAME	VARCHAR\NVARCHAR(64)
VULN_USER_DOMAIN	VARCHAR\NVARCHAR(64)
VULN_TAXONOMY	VARCHAR\NVARCHAR(1000)
SCANNER_CLASSIFICATION	VARCHAR\NVARCHAR(255)
VULN_NAME	VARCHAR\NVARCHAR(300)
VULN_MODULE	VARCHAR\NVARCHAR(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\NVARCHAR(36)
SCANNER_ID	VARCHAR\NVARCHAR(36)
IP	VARCHAR\NVARCHAR(32)
HOST_NAME	VARCHAR\NVARCHAR(255)
LOCATION	VARCHAR\NVARCHAR(128)
DEPARTMENT	VARCHAR\NVARCHAR(128)
BUSINESS_SYSTEM	VARCHAR\NVARCHAR(128)
OPERATIONAL_ENVIRONMENT	VARCHAR\NVARCHAR(64)
CRITICALITY	INT
REGULATION	VARCHAR\NVARCHAR(128)
REGULATION_RATING	VARCHAR\NVARCHAR(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\NVARCHAR(36)
SCAN_ID	VARCHAR\NVARCHAR(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\NVARCHAR(36)
SCANNER_ID	VARCHAR\NVARCHAR(36)
SCAN_TYPE	VARCHAR\NVARCHAR(10)
SCAN_START_DATE	DATETIME
SCAN_END_DATE	DATETIME
CONSOLIDATION_SERVER	VARCHAR\NVARCHAR(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\NVARCHAR(36)
VULN_ID	VARCHAR\NVARCHAR(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\NVARCHAR(36)
PRODUCT_NAME	VARCHAR\NVARCHAR(100)
PRODUCT_VERSION	VARCHAR\NVARCHAR(64)
SCANNER_TYPE	VARCHAR\NVARCHAR(64)
VENDOR	VARCHAR\NVARCHAR(100)
SCANNER_INSTANCE	VARCHAR\NVARCHAR(64)
DATE_CREATED	DATETIME

Column Name	Datatype
DATE_MODIFIED	DATETIME
CREATED_BY	INT
MODIFIED_BY	INT

A Sentinel Troubleshooting Checklist

This checklist is provided to aid in diagnosing a problem. By filling in this checklist, you can solve common issues or reduce the amount of time needed to solve more complex issues.

Checklist Item	Information	Example
Novell Version:		V6.0
Novell Platform and OS Version:		SuSE Linux Enterprise Server 10
Database Platform and OS Version:		Oracle 10.2.0.3 with critical patch #5881721
Sentinel Server Hardware Configuration ▪ Processor ▪ Memory ▪ Other		4 CPU @ 3 GHz 5 GB RAM
Database Server Hardware Configuration ▪ Processor ▪ Memory ▪ Other (if separate Box)		4 CPU @ 3.0 GHz 8 GB RAM
Database Storage Configuration (NAS, SAN, Local and so on.)		Local with offsite backup
Reporting Server OS and Configuration (Crystal Server)		Crystal XI SuSE Linux Enterprise Server 10 with MySQL

NOTE: Depending upon how your Sentinel system is configured (distributed), you may have to expand the above table. For instance additional information may be needed for DAS, Advisor, Sentinel Control Center, Collector Builder and communication layer.

1. Check the Novell Customer Center (http://support.novell.com/phone.html?sourceidint=suplnav4_phonesup)f or your particular issue:
   - Is this a known issue with a work-around?
   - Is this issue fixed in the latest patch release or hot-fix?
   - Is this issue currently scheduled to be fixed in a future release?
2. Determine the nature of the problem.
   - Can it be reproduced? Can the steps to reproduce the problem be enumerated?
   - What user action, if any, will cause the problem?
   - Is the issue periodic in nature?
3. Determine the severity of this problem.

- Is the system still useable?

4. Understand the environment and systems involved.
   - What platforms and product versions are involved?
   - Are there any non-standard or custom components involved?
   - Is it a high event rate environment?
   - What is the rate of events being collected?
   - What is the event rate of insertion into the database?
   - How many concurrent users are there?
   - Is Crystal reporting used? When are reports run?
   - Is correlation used? How many rules are deployed?

   Collect configuration files, log files and system information from appropriate subdirectories in $ESEC_HOME or %ESEC_HOME%. Assemble this information for possible future knowledge transfer.
5. Check the health of the system.
   - Can you log into the Sentinel Control Center?
   - Are events being generated and inserted into the database?
   - Can events be seen on the Sentinel Control Center?
   - Can events be retrieved from the database using quick query?
   - Check the RAM usage, disk space, process activity, CPU usage and network connectivity of the hosts involved.
   - Verify all expected Sentinel processes are running. Microsoft Task Manager can be used in a Windows environment. In Unix, the command `ps –ef|grep esecadm` can be used.
   - Check for any core dumps in any of the sub-directories of ESEC_HOME. Find out which process core dumped. (cd $ESEC_HOME, find . –name core –print)
   - Check for the sqlplus net access. Check for the tablespaces.
   - Make sure the Sonic broker is running. Connectivity can be verified using the Sonic management console. Check that the various connections are active from Novell processes. Make sure that a lock file is not preventing Sonic from starting. Optionally telnet to that server on the sonic port (that is telnet sentinel.company.com 10012)
   - Check whether the wrapper service is running on the server. (`ps –ef | grep wrapper`)
   - Are any errors visible in the Servers View of the Sentinel Control Center? Are any errors visible in the Event Source Management Live View in the Sentinel Control Center? What is the OS resource consumption on the Collector Managers?
6. Is there a problem with the Database?
   - Using sqlplus, can you log into the database?
   - Does the database allow a sqlplus login using the Novell dba account into the ESEC schema?
     - Does querying on one of the table succeed?
   - Does a select statement on a database table succeed?
   - Check the JDBC drivers, their locations and class path settings.

- If Oracle, do they have Partitioning installed (enter “select * from v$version;”) and used?
- Is the database being maintained by an administrator? By anyone?
- Has the database been modified by that administrator?
- Is SDM being used to maintain the partitions and archive/delete the partitions to make more room in the database?
- Using SDM what is the current partition? Is it P_MAX?

7. Inspect whether the product environment settings are correct.
   - Verify the sanity of User login shell scripts, environment variables, configurations, java home settings.
   - Are the environment variable set to run the correct jvm?
   - Verify the proper permissions on the folders for the installed product.
   - Check if any cron jobs are setup causing interference with our product’s functionality.
   - If the product is installed on NFS mounts, check the sanity of NFS mounts & NFS/NIS services.
8. Is there a possible memory leak?
   - Obtain the statistics on how fast the memory is being consumed and by which process.
   - Gather the metrics of the events throughput per Collector.
   - Run the prstat command on Solaris. This will give the process runtime statistics.
   - In Windows you can check the process size and handle count in task manager.

   This issue, if not resolved, is now ready for escalation. Possible results of escalation are:
   - Configuration file changes
   - Hot fixes or patches to your system
   - Enhancement request
   - Temporary workaround.

B Sentinel Service Logon Account

The purpose of this document is to describe in detail of how to set up Sentinel service logon account as NT AUTHORITY\NetworkService instead of Domain user account. This has been tested on the Windows 2003 platform only.

Sentinel Services

Sentinel Services should be set to run in order to use Sentinel application. To run a service you need to login to the machine where Sentinel is installed using a logon Account. The different logon accounts and advantages of using a logon account are discussed in this document.

Introduction to Service Logon Accounts

A service must log on to an account to access resources and objects on the operating system. If you select an account that does not have permission to log on as a service, the Services snap-in automatically grants that account the user rights that are required to log on as a service on the computer that you are managing. However, this does not guarantee that the service will start. For example, it is recommended that the user accounts that are used to log on as a service have the **Password never expires** check box selected in their properties dialog box and that they have strong passwords. If account lockout policy is enabled and the account is locked out, the service will malfunction.

The following table describes the service logon accounts and how they are used.

Logon Account	Description
Local System Account	The Local System account is a powerful account that has full access to the system, including the directory service on domain controllers. If a service logs onto the Local System account on a domain controller, that service has access to the entire domain. Some services are configured by default to log on to the Local System account. Do not change the default service setting. Local System account is a predefined local account that is used to start a service and provide the security context for that service. The name of the account is NT AUTHORITY\System. This account does not have a password and any password information that you supply is ignored. The Local System account has full access to the system, including the directory service on domain controllers. Because the Local System account acts as a computer on the network, it has access to network resources.

Logon Account	Description
Local Service Account	The Local Service account is a special built-in account that is similar to an authenticated user account. The Local Service account has the same level of access to resources and objects as members of the Users group. This limited access helps safeguard your system if individual services or processes are compromised. Services that run as the Local Service account access network resources as a null session with no credentials. Local Service account is a predefined local account that is used to start a service and provide the security context for that service. The name of the account is NT AUTHORITY\LocalService. The Local Service account has limited access to the local computer and Anonymous access to network resources.
Network Service Account	The Network Service account is a special, built-in account that is similar to an authenticated user account. The Network Service account has the same level of access to resources and objects as members of the Users group. This limited access helps safeguard your system if individual services or processes are compromised. Services that run as the Network Service account access network resources using the credentials of the computer account. Network Service account is a predefined local account that is used to start a service and provide the security context for that service. The name of the account is NT AUTHORITY\NetworkService. The Network Service account has limited access to the local computer and authenticated access (as the computer account) to network resources.

Disadvantages of running a service in the context of a user logon

1. The account must be created before the service can run. If the setup program for the service creates the account, Setup must run from an account that has sufficient administrative credentials to create accounts in the directory service.
2. Service account names and passwords are stored on each computer on which the service is installed. If the password for a service account on a computer is changed or expires, the service cannot start on that computer until the password is set to the new password for that service. The recommendation is to use LocalService and Network Service instead of using an account that requires a password: this simplifies password management.
3. If a service account is renamed, locked out, disabled, or deleted, the service cannot start on that computer until the account is reset.

Due to the above disadvantages, Novell has tested out running Sentinel service under NT AUTHORITY\NetworkService account. NT AUTHORITY\LocalService account does not have enough privilege for this

purpose, since DAS processes need to communicate to database server on the network.

NOTE: Novell has tested and recommends choosing Network Service account option.

To Setup NT AUTHORITY\NetworkService as the Logon Account for Sentinel Service

To setup NT AUTHORITY\NetworkService as the logon account for Sentinel service, you will have to perform the following:

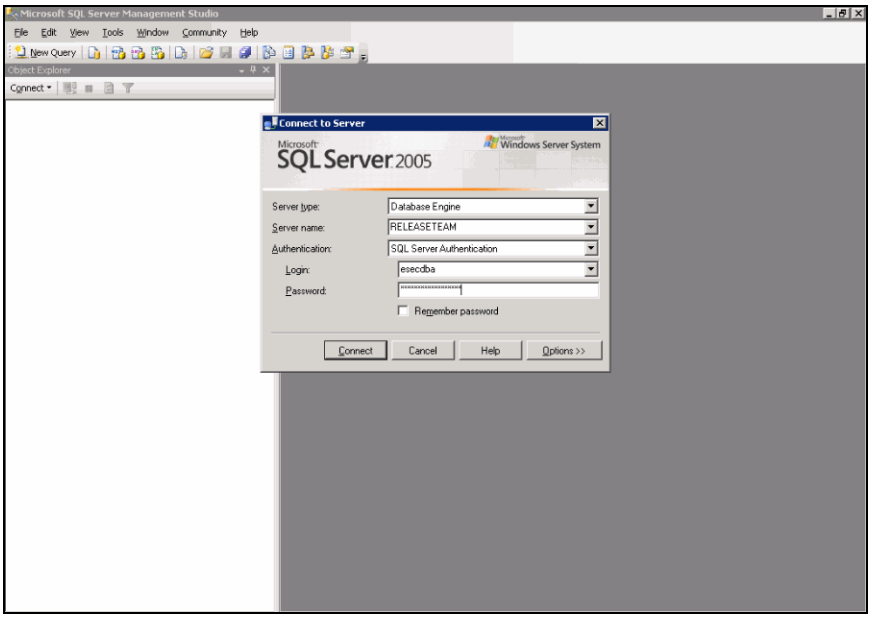
- Add the machine that runs Sentinel Service as a login account to ESEC and ESEC_WF database instances (performed on the database machine)
- Change the logon account for Sentinel service to NT AUTHORITY\NetworkService (performed on your remote machine)
- Setting the Sentinel startup (performed on your remote machine)

Adding Sentinel Service as a Login Account to ESEC and ESEC_WF DB Instances

To add a login of a remote machine to the database server:

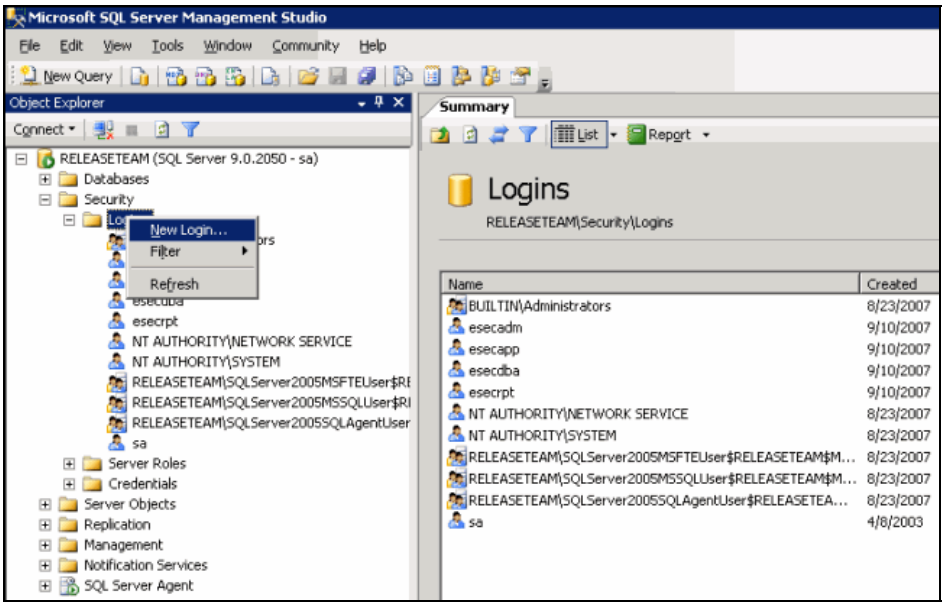
NOTE: As an example, the following are steps to add secnet\case1 as a login to the database server.

1. On your database machine, open up SQL Server Management Studio. Enter the user credentials in the Login Window.

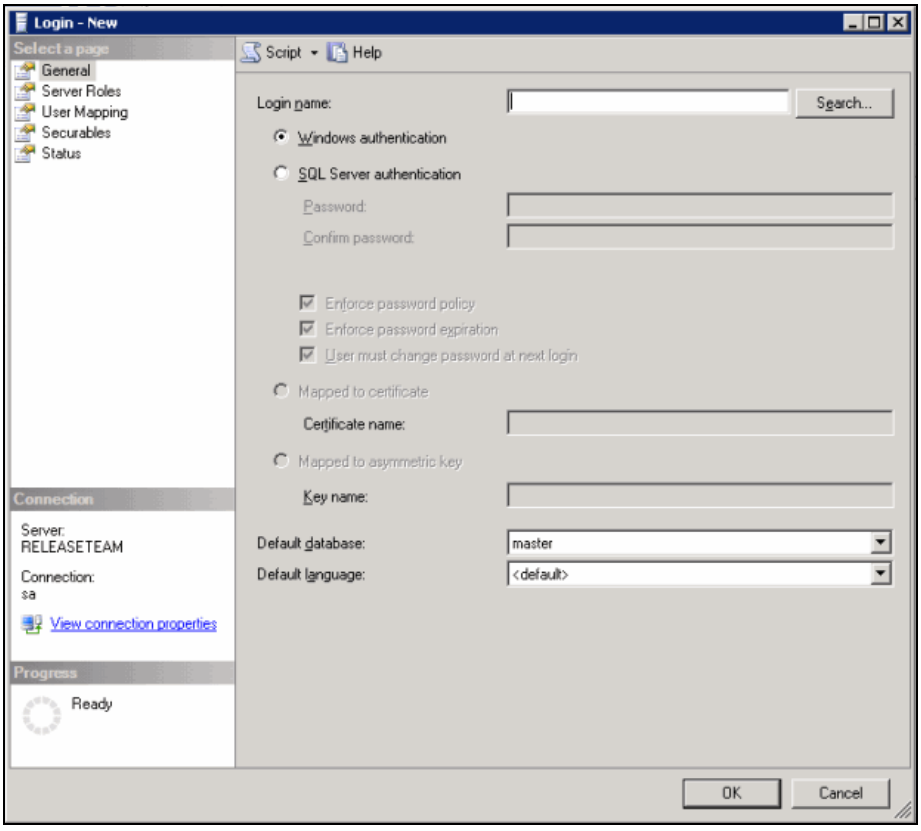


Click *Connect*

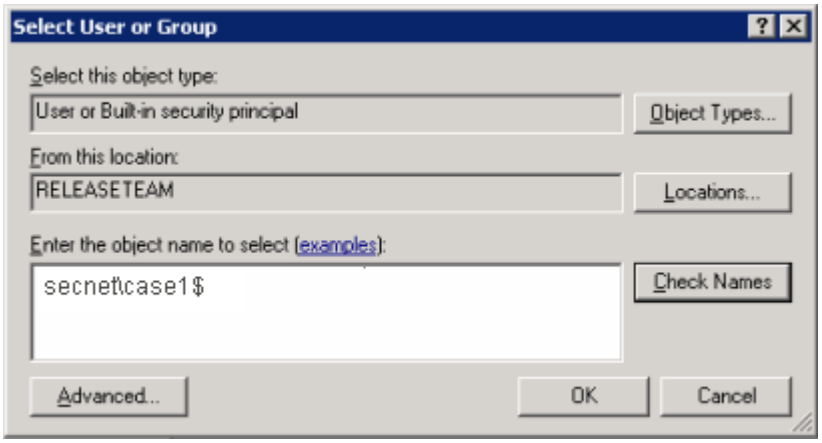
2. In the object explorer pane, under SQL Server Group, expand Security folder and highlight Logins folder.
3. Right-click *Logins* > *New login*.



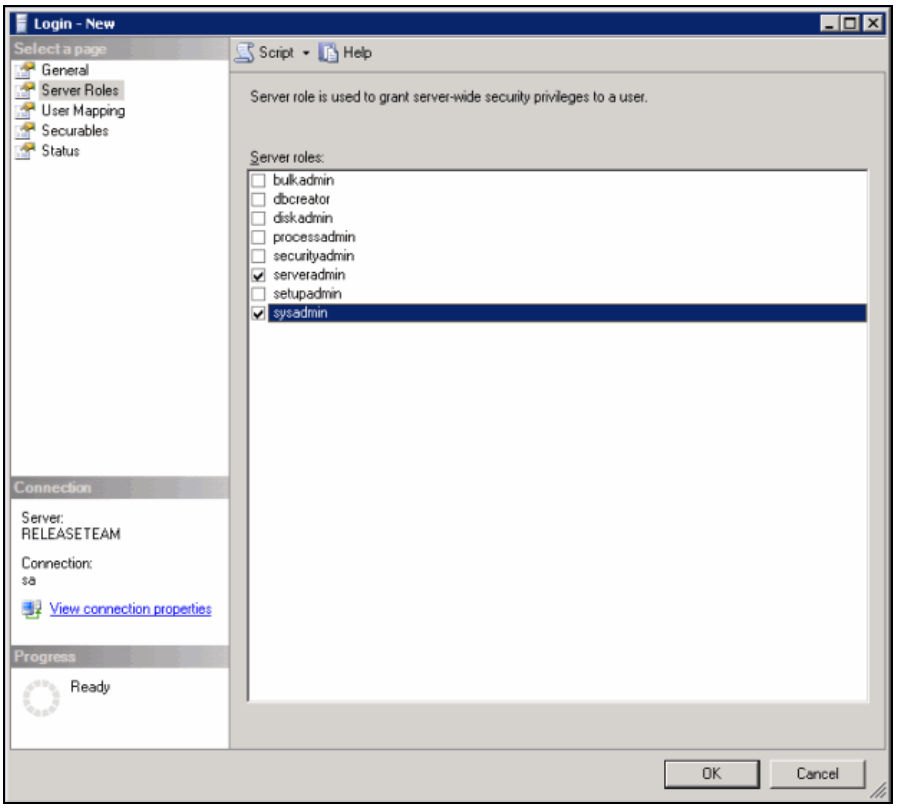
4. In the *Login-New* window, enter the login name.



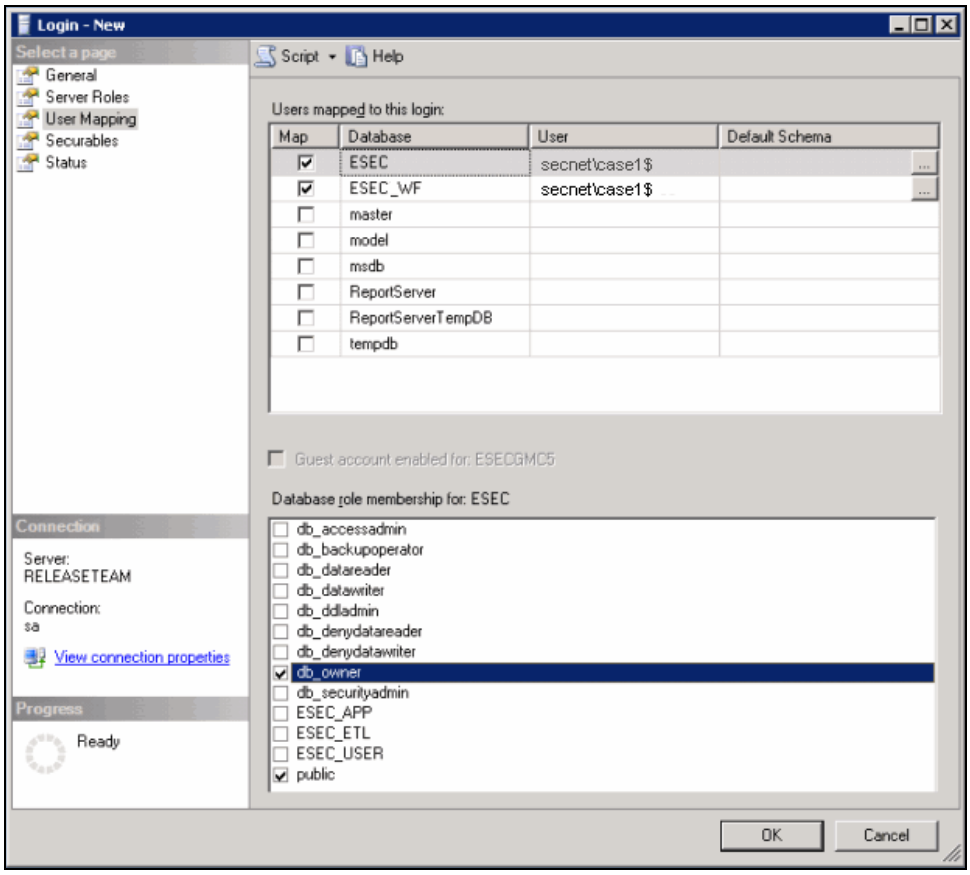
Alternatively, you can click the *Search* button next to the *Login name* field. The following screen displays:



5. In the Enter the object name to select field, enter a domain name and user name (secnet\case1$ is provided as an example). This is the machine <domain name>\<name of machine>$ you are adding as a login to the database server. Click *OK*.
6. Click *Server Roles* in the Select a page navigation pane. Select "sysadmin" and "serveradmin" as Server Roles as shown below:



7. Click *User Mapping* in the Select a page navigation pane. Select access to ESEC and ESEC_WF as "public" and "db_owner" as shown below:

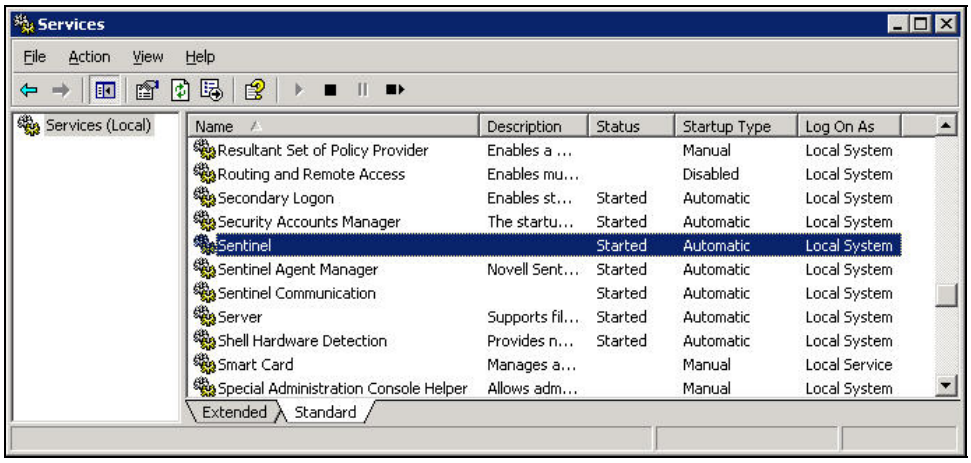


Click *OK*.

Changing logon account

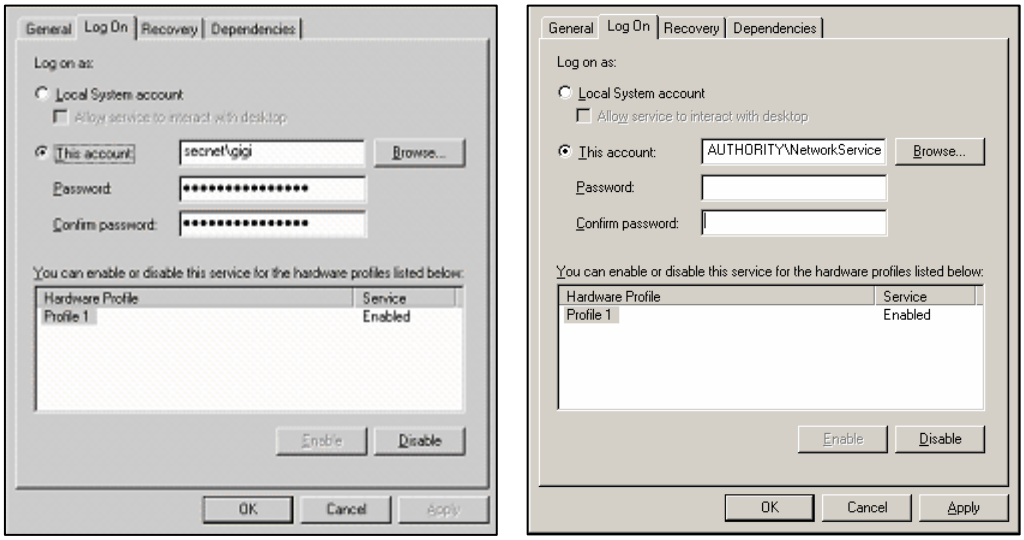
To change the logon for Sentinel Service to NT AUTHORITY\NetworkService:

1. On your remote machine you are connecting to the database, click *Start > Programs >Administrative Tools > Services*.

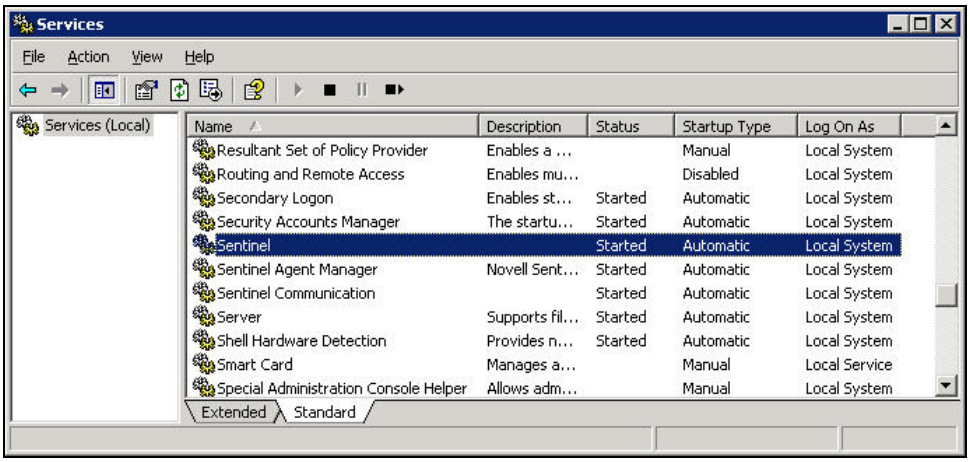


2. Stop the Sentinel service, right-click > *Properties* > *Log On* tab.

3. Click *This account* and in the field enter '*NT AUTHORITY\NetworkService*'. Clear the 'Password' and 'Confirm password' fields.



4. Click *OK*. The Services window for the Sentinel Service should indicate 'Network Service' under the 'Log On As' column.



Setting the Sentinel Service to Start Successfully

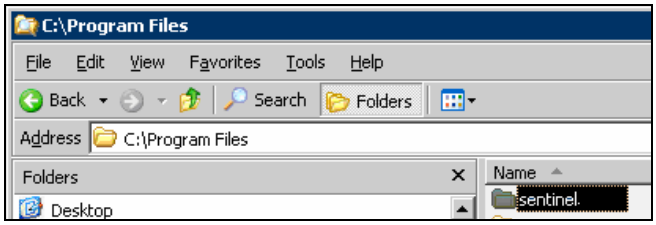
In order for the Sentinel Service to start successfully, NT AUTHORITY\NetworkService account should have write permission to %ESEC_HOME%. According to Microsoft documentation, the NetworkService account has the following privileges:

- SE_ASSIGNPRIMARYTOKEN_NAME (disabled)
- SE_AUDIT_NAME (disabled)
- SE_CHANGE_NOTIFY_NAME (enabled)
- SE_CREATE_GLOBAL_NAME (enabled)
- SE_IMPERSONATE_NAME (enabled)
- SE_INCREASE_QUOTA_NAME (disabled)
- SE_SHUTDOWN_NAME (disabled)
- SE_UNDOCK_NAME (disabled)
- Any privileges assigned to users and authenticated users

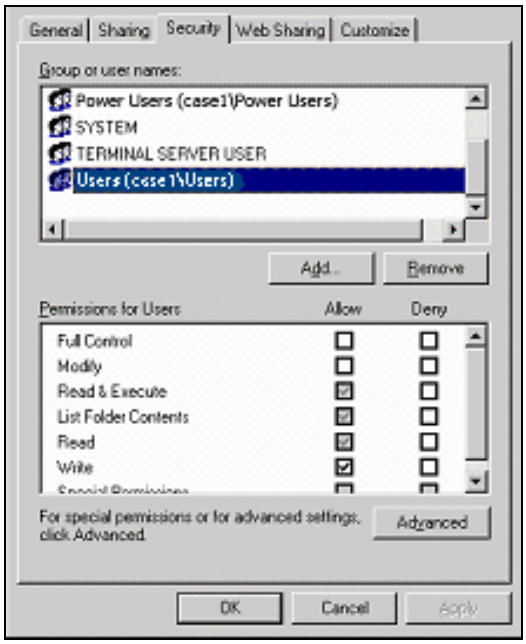
You will have to grant write access to %ESEC_HOME% to the Users group.

To set the Sentinel Service to start successfully:

1. Open Window's Explorer and navigate to %ESEC_HOME%.
2. Right-click the Sentinel parent folder (Typically named sentinel6) > *Properties* > *Security* tab.



3. Highlight Users group. Grant Read & Execute, List Folder Contents, Read, Write Permission.



Click *OK*.

4. In the Services window, restart the Sentinel service.

C Sentinel Service Permission Tables

The purpose of this document is to describe in detail various Sentinel Services and the Permissions they require for their functioning.

Advisor

Sentinel Component	Sentinel Service	Sentinel Process	Function summary	Permission's required	Permission Explanation
Advisor	Sentinel	java	Download (optional) and processes Advisor attack data.	Network access Internet access over port 443 (optional) File read access to: - ESEC_HOME/config - ESEC_HOME/lib - ESEC_HOME/jre File write access to: - ESEC_HOME/data - ESEC_HOME/log	It connects to the database to read and insert data. It communicates over the network with iSCALE to notify other processes it is down processing a feed. It reads local configuration files and uses the java executable. It writes log files as well as caches data in the local file system.

Collector Manager

Sentinel Component	Sentinel Service	Sentinel Process	Function summary	Permissions required	Permission Explanation
Collector Manager	Sentinel	java agentengine (child process)	Manages Connectors and Collectors. It spawns off an agentengine process for each Collector it manages. Collector Manager also publishes system status messages, performs global filtering of events, and performs referential mappings. The agentengine process runs as an interpreter for Collector scripts, which normalize unprocessed (raw) events from security devices and systems producing event, vulnerability, and asset data that Sentinel can analyze and store in it's database.	Network access (both outgoing access and local access to bind to ports greater than 1024) File read access to: - ESEC_HOME/config - ESEC_HOME/lib - ESEC_HOME/jre File write access to: - ESEC_HOME/data - ESEC_HOME/log **NOTE**: Additionally, will need access to other resources depending which Connectors it is configured to run and which Event Sources it connecting to. Please refer to the individual Connector documentation for any additional permission requirements.	It communicates with iSCALE for configuration, event processing, and mapping data. It reads local configuration files and uses the java executable. It writes log files as well as caches data in the local file system.

Correlation Engine

Sentinel Component	Sentinel Service	Sentinel Process	Function summary	Permission's required	Permission Explanation
Correlation Engine	Sentinel	java	Receives events from the Collector Manager and publishes correlated events based on user-defined correlation rules.	Network access File read access to: - ESEC_HOME/config - ESEC_HOME/lib - ESEC_HOME/jre File write access to: - ESEC_HOME/data - ESEC_HOME/log	It communicates over the network with iSCALE for configuration, event processing, and correlated event generation. It reads local configuration files and uses the java executable. It writes log files as well as caches data in the local file system.

Data Access Server (DAS)

Sentinel Component	Sentinel Service	Sentinel Process	Function summary	Permission's required	Permission Explanation
DAS	Sentinel	java (das_binary)	Responsible for event insertion.	Network access Database Acess File read access to: - ESEC_HOME/config - ESEC_HOME/lib - ESEC_HOME/jre File write access to: - ESEC_HOME/data - ESEC_HOME/log	It connects to the database to read and insert data. It communicates over the network with iSCALE for configuration and event processing and other general data processing. It reads local configuration files and uses the java executable. It writes log files as well as caches data in the local file system.
		java (das_query)	Provides general database access services, map data server, exploit detection data generation, Sentinel user login, and other general services.		
		java (das_rt)	Provides data that drives the Active View charts.		
		java (das_itrac)	Provides services to use and manage iTRAC workflow processes.		
		java (das_aggregation)	Summaries event data into summary database tables, primarily for use by reports.		

Sentinel Communication Server

Sentinel Component	Sentinel Service	Sentinel Process	Function summary	Permission's required	Permission Explanation
Communication Server (iSCALE / MOM)	Sentinel	java (Sonic)	iSCALE - A Message Oriented Middleware (MOM). The iSCALE component provides a Java Message Service (JMS) framework for inter-process communication. Processes communicate through a broker, which is responsible for routing and buffering messages.	Network access (binds to port greater than 1024) File read access to: - ESEC_HOME/jre File write access to: - ESEC_HOME/3rdparty/SonicMQ/MQ7.0	It binds to local ports to accept TCP connections in order to perform its duties as a communication server. It reads local configuration files and uses the java executable. It writes to Sonic's internal database on the local file system.
		java (das_proxy)	iSCALE also has an SSL proxy that acts as an SSL bridge between the message bus and a client connecting through SSL.	Network access (binds to ports greater than 1024) File read access to: - ESEC_HOME/config - ESEC_HOME/lib - ESEC_HOME/jre File write access to: - ESEC_HOME/3rdparty/SonicMQ/MQ7.0 - ESEC_HOME/data - ESEC_HOME/log - ESEC_HOME/config	It binds to local ports to accept SSL connections in order to perform its duties as a communication server. It reads local configuration files and uses the java executable. It writes log files, caches data, and writes to Sonic's internal database on the local file system. It also will write certificates to config directory when required.

Sentinel Service

Sentinel Component	Sentinel Service	Sentinel Process	Function summary	Permission's required	Permission Explanation
Sentinel Service	Sentinel	wrapper	Registers as a service with the operating system and, when executed, launches the java Sentinel Service.	Network access File read access to: - ESEC_HOME/config - ESEC_HOME/lib - ESEC_HOME/jre File write access to: - ESEC_HOME/log	It communicates over the network with iSCALE for configuration and status reporting. It reads local configuration files and uses the java executable. It writes log files to the local file system.
		java (sentinel)	The java Sentinel Service process that is responsible for launching, restarting, and reporting status on the other Sentinel Server processes.		

Reporting Server

Sentinel Component	Sentinel Application	Sentinel Service	Sentinel Process	Function summary	Permission's required	Permission Explanation
Reports	-	-	-	Crystal Reports XI or Crystal Enterprise 9 Standard is one of the reporting tools with Sentinel.	-	Needs to have the odbc driver or oracle driver pointing to the sentinel db

D Microsoft SQL Users, Roles & Access Permissions for Sentinel

The purpose of this document is to provide a detailed breakdown of Sentinel database users, roles and their access permissions.

Sentinel Database Instance

ESEC

Users:

- esecadm
- esecapp
- esecdba
- esecrpt
- Other users

NOTE: Other users are created through User Manager. For detailed access permissions, see "Sentinel Database Roles".

Roles:

- **ESEC_APP:** The same permission as db_owner
- **ESEC_ETL:**
- **ESEC_USER**

ESEC_WF

- **Users**: esecapp – For detailed access permissions see the "Sentinel Database Users" section.
- **Roles**: ESEC_APP – For detailed access permissions see the "Sentinel Database Roles" section.

Sentinel Database Users

Summary

User Name	Group Name	Login Name	Default DB Name
Esecadm	ESEC_USER	esecadm	ESEC
Esecapp	ESEC_APP	esecapp	ESEC
Esecapp	ESEC_ETL	esecapp	ESEC
Esecapp	db_owner	esecapp	ESEC
Esecdba	db_owner	esecdba	ESEC
Esecrpt	ESEC_USER	esecrpt	ESEC

esecadm

Login Name	DB Name	User Name	User of Alias
Esecadm	ESEC	ESEC_USER	MemberOf
Esecadm	ESEC	esecadm	User

esecapp

Login Name	DB Name	User Name	User of Alias
Esecapp	ESEC	ESEC_APP	MemberOf
Esecapp	ESEC	ESEC_ETL	MemberOf
Esecapp	ESEC	esecapp	User
Esecapp	ESEC	db_owner	MemberOf
Esecapp	ESEC_WF	ESEC_APP	MemberOf
Esecapp	ESEC_WF	esecapp	User

esecdba

Login Name	DB Name	User Name	User of Alias
Esecdba	ESEC	db_owner	MemberOf
Esecdba	ESEC	esecdba	User

esecrpt

Login Name	DB Name	User Name	User of Alias
Esecrpt	ESEC	ESEC_USER	MemberOf
Esecrpt	ESEC	esecrpt	User

Sentinel Database Roles

Summary

- **ESEC_APP:** It is a database role for ESEC and ESEC_WF. It has the same permission as db_owner for ESEC instance.
- **ESEC_ETL:** It is a database role for ESEC instance.
- **ESEC_USER:** A role for ESEC instance.

ESEC_APP

For ESEC instance, ESEC_APP has the same permission as db_owner. ESEC_APP performs the activities of all database roles, as well as other maintenance and configuration activities in the database. The permissions of this role span all of the other fixed database roles.

For ESEC_WF instance, these are the permission for ESEC_APP role:

Role Name	Object Name	Action	Type
ESEC_APP	Activities	193 SELECT	U User table
ESEC_APP	Activities	195 INSERT	U User table
ESEC_APP	Activities	196 DELETE	U User table
ESEC_APP	Activities	197 UPDATE	U User table
ESEC_APP	ActivityData	193 SELECT	U User table
ESEC_APP	ActivityData	195 INSERT	U User table
ESEC_APP	ActivityData	196 DELETE	U User table
ESEC_APP	ActivityData	197 UPDATE	U User table
ESEC_APP	ActivityDataBLOBs	193 SELECT	U User table
ESEC_APP	ActivityDataBLOBs	195 INSERT	U User table
ESEC_APP	ActivityDataBLOBs	196 DELETE	U User table
ESEC_APP	ActivityDataBLOBs	197 UPDATE	U User table
ESEC_APP	ActivityDataWOB	193 SELECT	U User table
ESEC_APP	ActivityDataWOB	195 INSERT	U User table

Role Name	Object Name	Action	Type
ESEC_APP	ActivityDataWOB	196 DELETE	U User table
ESEC_APP	ActivityDataWOB	197 UPDATE	U User table
ESEC_APP	ActivityStateEventAudits	193 SELECT	U User table
ESEC_APP	ActivityStateEventAudits	195 INSERT	U User table
ESEC_APP	ActivityStateEventAudits	196 DELETE	U User table
ESEC_APP	ActivityStateEventAudits	197 UPDATE	U User table
ESEC_APP	ActivityStates	193 SELECT	U User table
ESEC_APP	ActivityStates	195 INSERT	U User table
ESEC_APP	ActivityStates	196 DELETE	U User table
ESEC_APP	ActivityStates	197 UPDATE	U User table
ESEC_APP	AndJoinTable	193 SELECT	U User table
ESEC_APP	AndJoinTable	195 INSERT	U User table
ESEC_APP	AndJoinTable	196 DELETE	U User table
ESEC_APP	AndJoinTable	197 UPDATE	U User table
ESEC_APP	AssignmentEventAudits	193 SELECT	U User table
ESEC_APP	AssignmentEventAudits	195 INSERT	U User table
ESEC_APP	AssignmentEventAudits	196 DELETE	U User table
ESEC_APP	AssignmentEventAudits	197 UPDATE	U User table
ESEC_APP	AssignmentsTable	193 SELECT	U User table
ESEC_APP	AssignmentsTable	195 INSERT	U User table
ESEC_APP	AssignmentsTable	196 DELETE	U User table
ESEC_APP	AssignmentsTable	197 UPDATE	U User table
ESEC_APP	Counters	193 SELECT	U User table
ESEC_APP	Counters	195 INSERT	U User table
ESEC_APP	Counters	196 DELETE	U User table
ESEC_APP	Counters	197 UPDATE	U User table
ESEC_APP	CreateProcessEventAudits	193 SELECT	U User table
ESEC_APP	CreateProcessEventAudits	195 INSERT	U User table
ESEC_APP	CreateProcessEventAudits	196 DELETE	U User table
ESEC_APP	CreateProcessEventAudits	197 UPDATE	U User table
ESEC_APP	DataEventAudits	193 SELECT	U User table
ESEC_APP	DataEventAudits	195 INSERT	U User table
ESEC_APP	DataEventAudits	196 DELETE	U User table
ESEC_APP	DataEventAudits	197 UPDATE	U User table
ESEC_APP	Deadlines	193 SELECT	U User table
ESEC_APP	Deadlines	195 INSERT	U User table
ESEC_APP	Deadlines	196 DELETE	U User table
ESEC_APP	Deadlines	197 UPDATE	U User table
ESEC_APP	EventTypes	193 SELECT	U User table
ESEC_APP	EventTypes	195 INSERT	U User table
ESEC_APP	EventTypes	196 DELETE	U User table
ESEC_APP	EventTypes	197 UPDATE	U User table
ESEC_APP	GroupGroupTable	193 SELECT	U User table
ESEC_APP	GroupGroupTable	195 INSERT	U User table
ESEC_APP	GroupGroupTable	196 DELETE	U User table
ESEC_APP	GroupGroupTable	197 UPDATE	U User table
ESEC_APP	GroupTable	193 SELECT	U User table
ESEC_APP	GroupTable	195 INSERT	U User table
ESEC_APP	GroupTable	196 DELETE	U User table

Role Name	Object Name	Action	Type
ESEC_APP	GroupTable	197 UPDATE	U User table
ESEC_APP	GroupUser	193 SELECT	U User table
ESEC_APP	GroupUser	195 INSERT	U User table
ESEC_APP	GroupUser	196 DELETE	U User table
ESEC_APP	GroupUser	197 UPDATE	U User table
ESEC_APP	GroupUserPackLevelParticipant	193 SELECT	U User table
ESEC_APP	GroupUserPackLevelParticipant	195 INSERT	U User table
ESEC_APP	GroupUserPackLevelParticipant	196 DELETE	U User table
ESEC_APP	GroupUserPackLevelParticipant	197 UPDATE	U User table
ESEC_APP	GroupUserProcLevelParticipant	193 SELECT	U User table
ESEC_APP	GroupUserProcLevelParticipant	195 INSERT	U User table
ESEC_APP	GroupUserProcLevelParticipant	196 DELETE	U User table
ESEC_APP	GroupUserProcLevelParticipant	197 UPDATE	U User table
ESEC_APP	LockTable	193 SELECT	U User table
ESEC_APP	LockTable	195 INSERT	U User table
ESEC_APP	LockTable	196 DELETE	U User table
ESEC_APP	LockTable	197 UPDATE	U User table
ESEC_APP	NewEventAuditData	193 SELECT	U User table
ESEC_APP	NewEventAuditData	195 INSERT	U User table
ESEC_APP	NewEventAuditData	196 DELETE	U User table
ESEC_APP	NewEventAuditData	197 UPDATE	U User table
ESEC_APP	NewEventAuditDataBLOBs	193 SELECT	U User table
ESEC_APP	NewEventAuditDataBLOBs	195 INSERT	U User table
ESEC_APP	NewEventAuditDataBLOBs	196 DELETE	U User table
ESEC_APP	NewEventAuditDataBLOBs	197 UPDATE	U User table
ESEC_APP	NewEventAuditDataWOB	193 SELECT	U User table
ESEC_APP	NewEventAuditDataWOB	195 INSERT	U User table
ESEC_APP	NewEventAuditDataWOB	196 DELETE	U User table
ESEC_APP	NewEventAuditDataWOB	197 UPDATE	U User table
ESEC_APP	NextXPDLVersions	193 SELECT	U User table
ESEC_APP	NextXPDLVersions	195 INSERT	U User table
ESEC_APP	NextXPDLVersions	196 DELETE	U User table
ESEC_APP	NextXPDLVersions	197 UPDATE	U User table
ESEC_APP	NormalUser	193 SELECT	U User table
ESEC_APP	NormalUser	195 INSERT	U User table
ESEC_APP	NormalUser	196 DELETE	U User table
ESEC_APP	NormalUser	197 UPDATE	U User table
ESEC_APP	ObjectId	193 SELECT	U User table
ESEC_APP	ObjectId	195 INSERT	U User table
ESEC_APP	ObjectId	196 DELETE	U User table
ESEC_APP	ObjectId	197 UPDATE	U User table
ESEC_APP	OldEventAuditData	193 SELECT	U User table
ESEC_APP	OldEventAuditData	195 INSERT	U User table
ESEC_APP	OldEventAuditData	196 DELETE	U User table
ESEC_APP	OldEventAuditData	197 UPDATE	U User table
ESEC_APP	OldEventAuditDataBLOBs	193 SELECT	U User table
ESEC_APP	OldEventAuditDataBLOBs	195 INSERT	U User table
ESEC_APP	OldEventAuditDataBLOBs	196 DELETE	U User table
ESEC_APP	OldEventAuditDataBLOBs	197 UPDATE	U User table

Role Name	Object Name	Action	Type
ESEC_APP	OldEventAuditDataWOB	193 SELECT	U User table
ESEC_APP	OldEventAuditDataWOB	195 INSERT	U User table
ESEC_APP	OldEventAuditDataWOB	196 DELETE	U User table
ESEC_APP	OldEventAuditDataWOB	197 UPDATE	U User table
ESEC_APP	PackLevelParticipant	193 SELECT	U User table
ESEC_APP	PackLevelParticipant	195 INSERT	U User table
ESEC_APP	PackLevelParticipant	196 DELETE	U User table
ESEC_APP	PackLevelParticipant	197 UPDATE	U User table
ESEC_APP	PackLevelXPDLApp	193 SELECT	U User table
ESEC_APP	PackLevelXPDLApp	195 INSERT	U User table
ESEC_APP	PackLevelXPDLApp	196 DELETE	U User table
ESEC_APP	PackLevelXPDLApp	197 UPDATE	U User table
ESEC_APP	PackLevelXPDLAppTAAppDetail	193 SELECT	U User table
ESEC_APP	PackLevelXPDLAppTAAppDetail	195 INSERT	U User table
ESEC_APP	PackLevelXPDLAppTAAppDetail	196 DELETE	U User table
ESEC_APP	PackLevelXPDLAppTAAppDetail	197 UPDATE	U User table
ESEC_APP	PackLevelXPDLAppTAAppDetailUsr	193 SELECT	U User table
ESEC_APP	PackLevelXPDLAppTAAppDetailUsr	195 INSERT	U User table
ESEC_APP	PackLevelXPDLAppTAAppDetailUsr	196 DELETE	U User table
ESEC_APP	PackLevelXPDLAppTAAppDetailUsr	197 UPDATE	U User table
ESEC_APP	PackLevelXPDLAppTAAppUser	193 SELECT	U User table
ESEC_APP	PackLevelXPDLAppTAAppUser	195 INSERT	U User table
ESEC_APP	PackLevelXPDLAppTAAppUser	196 DELETE	U User table
ESEC_APP	PackLevelXPDLAppTAAppUser	197 UPDATE	U User table
ESEC_APP	PackLevelXPDLAppToolAgentApp	193 SELECT	U User table
ESEC_APP	PackLevelXPDLAppToolAgentApp	195 INSERT	U User table
ESEC_APP	PackLevelXPDLAppToolAgentApp	196 DELETE	U User table
ESEC_APP	PackLevelXPDLAppToolAgentApp	197 UPDATE	U User table
ESEC_APP	ProcessData	193 SELECT	U User table
ESEC_APP	ProcessData	195 INSERT	U User table
ESEC_APP	ProcessData	196 DELETE	U User table
ESEC_APP	ProcessData	197 UPDATE	U User table
ESEC_APP	ProcessDataBLOBs	193 SELECT	U User table
ESEC_APP	ProcessDataBLOBs	195 INSERT	U User table
ESEC_APP	ProcessDataBLOBs	196 DELETE	U User table
ESEC_APP	ProcessDataBLOBs	197 UPDATE	U User table
ESEC_APP	ProcessDataWOB	193 SELECT	U User table
ESEC_APP	ProcessDataWOB	195 INSERT	U User table
ESEC_APP	ProcessDataWOB	196 DELETE	U User table
ESEC_APP	ProcessDataWOB	197 UPDATE	U User table
ESEC_APP	ProcessDefinitions	193 SELECT	U User table
ESEC_APP	ProcessDefinitions	195 INSERT	U User table
ESEC_APP	ProcessDefinitions	196 DELETE	U User table
ESEC_APP	ProcessDefinitions	197 UPDATE	U User table
ESEC_APP	Processes	193 SELECT	U User table
ESEC_APP	Processes	195 INSERT	U User table
ESEC_APP	Processes	196 DELETE	U User table
ESEC_APP	Processes	197 UPDATE	U User table
ESEC_APP	ProcessRequesters	193 SELECT	U User table

Role Name	Object Name	Action	Type
ESEC_APP	ProcessRequesters	195 INSERT	U User table
ESEC_APP	ProcessRequesters	196 DELETE	U User table
ESEC_APP	ProcessRequesters	197 UPDATE	U User table
ESEC_APP	ProcessStateEventAudits	193 SELECT	U User table
ESEC_APP	ProcessStateEventAudits	195 INSERT	U User table
ESEC_APP	ProcessStateEventAudits	196 DELETE	U User table
ESEC_APP	ProcessStateEventAudits	197 UPDATE	U User table
ESEC_APP	ProcessStates	193 SELECT	U User table
ESEC_APP	ProcessStates	195 INSERT	U User table
ESEC_APP	ProcessStates	196 DELETE	U User table
ESEC_APP	ProcessStates	197 UPDATE	U User table
ESEC_APP	ProcLevelParticipant	193 SELECT	U User table
ESEC_APP	ProcLevelParticipant	195 INSERT	U User table
ESEC_APP	ProcLevelParticipant	196 DELETE	U User table
ESEC_APP	ProcLevelParticipant	197 UPDATE	U User table
ESEC_APP	ProcLevelXPDLApp	193 SELECT	U User table
ESEC_APP	ProcLevelXPDLApp	195 INSERT	U User table
ESEC_APP	ProcLevelXPDLApp	196 DELETE	U User table
ESEC_APP	ProcLevelXPDLApp	197 UPDATE	U User table
ESEC_APP	ProcLevelXPDLAppTAAppDetail	193 SELECT	U User table
ESEC_APP	ProcLevelXPDLAppTAAppDetail	195 INSERT	U User table
ESEC_APP	ProcLevelXPDLAppTAAppDetail	196 DELETE	U User table
ESEC_APP	ProcLevelXPDLAppTAAppDetail	197 UPDATE	U User table
ESEC_APP	ProcLevelXPDLAppTAAppDetailUsr	193 SELECT	U User table
ESEC_APP	ProcLevelXPDLAppTAAppDetailUsr	195 INSERT	U User table
ESEC_APP	ProcLevelXPDLAppTAAppDetailUsr	196 DELETE	U User table
ESEC_APP	ProcLevelXPDLAppTAAppDetailUsr	197 UPDATE	U User table
ESEC_APP	ProcLevelXPDLAppTAAppUser	193 SELECT	U User table
ESEC_APP	ProcLevelXPDLAppTAAppUser	195 INSERT	U User table
ESEC_APP	ProcLevelXPDLAppTAAppUser	196 DELETE	U User table
ESEC_APP	ProcLevelXPDLAppTAAppUser	197 UPDATE	U User table
ESEC_APP	ProcLevelXPDLAppToolAgentApp	193 SELECT	U User table
ESEC_APP	ProcLevelXPDLAppToolAgentApp	195 INSERT	U User table
ESEC_APP	ProcLevelXPDLAppToolAgentApp	196 DELETE	U User table
ESEC_APP	ProcLevelXPDLAppToolAgentApp	197 UPDATE	U User table
ESEC_APP	ResourcesTable	193 SELECT	U User table
ESEC_APP	ResourcesTable	195 INSERT	U User table
ESEC_APP	ResourcesTable	196 DELETE	U User table
ESEC_APP	ResourcesTable	197 UPDATE	U User table
ESEC_APP	StateEventAudits	193 SELECT	U User table
ESEC_APP	StateEventAudits	195 INSERT	U User table
ESEC_APP	StateEventAudits	196 DELETE	U User table
ESEC_APP	StateEventAudits	197 UPDATE	U User table
ESEC_APP	ToolAgentApp	193 SELECT	U User table
ESEC_APP	ToolAgentApp	195 INSERT	U User table
ESEC_APP	ToolAgentApp	196 DELETE	U User table
ESEC_APP	ToolAgentApp	197 UPDATE	U User table
ESEC_APP	ToolAgentAppDetail	193 SELECT	U User table
ESEC_APP	ToolAgentAppDetail	195 INSERT	U User table

Role Name	Object Name	Action	Type
ESEC_APP	ToolAgentAppDetail	196 DELETE	U User table
ESEC_APP	ToolAgentAppDetail	197 UPDATE	U User table
ESEC_APP	ToolAgentAppDetailUser	193 SELECT	U User table
ESEC_APP	ToolAgentAppDetailUser	195 INSERT	U User table
ESEC_APP	ToolAgentAppDetailUser	196 DELETE	U User table
ESEC_APP	ToolAgentAppDetailUser	197 UPDATE	U User table
ESEC_APP	ToolAgentAppUser	193 SELECT	U User table
ESEC_APP	ToolAgentAppUser	195 INSERT	U User table
ESEC_APP	ToolAgentAppUser	196 DELETE	U User table
ESEC_APP	ToolAgentAppUser	197 UPDATE	U User table
ESEC_APP	ToolAgentUser	193 SELECT	U User table
ESEC_APP	ToolAgentUser	195 INSERT	U User table
ESEC_APP	ToolAgentUser	196 DELETE	U User table
ESEC_APP	ToolAgentUser	197 UPDATE	U User table
ESEC_APP	UserGroupTable	193 SELECT	U User table
ESEC_APP	UserGroupTable	195 INSERT	U User table
ESEC_APP	UserGroupTable	196 DELETE	U User table
ESEC_APP	UserGroupTable	197 UPDATE	U User table
ESEC_APP	UserPackLevelParticipant	193 SELECT	U User table
ESEC_APP	UserPackLevelParticipant	195 INSERT	U User table
ESEC_APP	UserPackLevelParticipant	196 DELETE	U User table
ESEC_APP	UserPackLevelParticipant	197 UPDATE	U User table
ESEC_APP	UserProcLevelParticipant	193 SELECT	U User table
ESEC_APP	UserProcLevelParticipant	195 INSERT	U User table
ESEC_APP	UserProcLevelParticipant	196 DELETE	U User table
ESEC_APP	UserProcLevelParticipant	197 UPDATE	U User table
ESEC_APP	UserTable	193 SELECT	U User table
ESEC_APP	UserTable	195 INSERT	U User table
ESEC_APP	UserTable	196 DELETE	U User table
ESEC_APP	UserTable	197 UPDATE	U User table
ESEC_APP	XPDLApplicationPackage	193 SELECT	U User table
ESEC_APP	XPDLApplicationPackage	195 INSERT	U User table
ESEC_APP	XPDLApplicationPackage	196 DELETE	U User table
ESEC_APP	XPDLApplicationPackage	197 UPDATE	U User table
ESEC_APP	XPDLApplicationProcess	193 SELECT	U User table
ESEC_APP	XPDLApplicationProcess	195 INSERT	U User table
ESEC_APP	XPDLApplicationProcess	196 DELETE	U User table
ESEC_APP	XPDLApplicationProcess	197 UPDATE	U User table
ESEC_APP	XPDLData	193 SELECT	U User table
ESEC_APP	XPDLData	195 INSERT	U User table
ESEC_APP	XPDLData	196 DELETE	U User table
ESEC_APP	XPDLData	197 UPDATE	U User table
ESEC_APP	XPDLHistory	193 SELECT	U User table
ESEC_APP	XPDLHistory	195 INSERT	U User table
ESEC_APP	XPDLHistory	196 DELETE	U User table
ESEC_APP	XPDLHistory	197 UPDATE	U User table
ESEC_APP	XPDLHistoryData	193 SELECT	U User table
ESEC_APP	XPDLHistoryData	195 INSERT	U User table
ESEC_APP	XPDLHistoryData	196 DELETE	U User table

Role Name	Object Name	Action	Type
ESEC_APP	XPDLHistoryData	197 UPDATE	U User table
ESEC_APP	XPDLParticipantPackage	193 SELECT	U User table
ESEC_APP	XPDLParticipantPackage	195 INSERT	U User table
ESEC_APP	XPDLParticipantPackage	196 DELETE	U User table
ESEC_APP	XPDLParticipantPackage	197 UPDATE	U User table
ESEC_APP	XPDLParticipantProcess	193 SELECT	U User table
ESEC_APP	XPDLParticipantProcess	195 INSERT	U User table
ESEC_APP	XPDLParticipantProcess	196 DELETE	U User table
ESEC_APP	XPDLParticipantProcess	197 UPDATE	U User table
ESEC_APP	XPDLReferences	193 SELECT	U User table
ESEC_APP	XPDLReferences	195 INSERT	U User table
ESEC_APP	XPDLReferences	196 DELETE	U User table
ESEC_APP	XPDLReferences	197 UPDATE	U User table
ESEC_APP	XPDLS	193 SELECT	U User table
ESEC_APP	XPDLS	195 INSERT	U User table
ESEC_APP	XPDLS	196 DELETE	U User table
ESEC_APP	XPDLS	197 UPDATE	U User table

ESEC_ETL

Role Name	Object Name	Action	Type
ESEC_ETL	ACTVY	193 SELECT	U User table
ESEC_ETL	ACTVY_PARM	193 SELECT	U User table
ESEC_ETL	ACTVY_REF	193 SELECT	U User table
ESEC_ETL	ACTVY_REF_PARM_VAL	193 SELECT	U User table
ESEC_ETL	ADV_ALERT	193 SELECT	U User table
ESEC_ETL	ADV_ALERT_CVE	193 SELECT	U User table
ESEC_ETL	ADV_ALERT_PRODUCT	193 SELECT	U User table
ESEC_ETL	ADV_ATTACK	193 SELECT	U User table
ESEC_ETL	ADV_ATTACK_ALERT	193 SELECT	U User table
ESEC_ETL	ADV_ATTACK_CVE	193 SELECT	U User table
ESEC_ETL	ADV_ATTACK_MAP	193 SELECT	U User table
ESEC_ETL	ADV_ATTACK_PLUGIN	193 SELECT	U User table
ESEC_ETL	ADV_CREDIBILITY	193 SELECT	U User table
ESEC_ETL	ADV_FEED	193 SELECT	U User table
ESEC_ETL	ADV_PRODUCT	193 SELECT	U User table
ESEC_ETL	ADV_PRODUCT_SERVICE_PACK	193 SELECT	U User table
ESEC_ETL	ADV_PRODUCT_VERSION	193 SELECT	U User table
ESEC_ETL	ADV_SEVERITY	193 SELECT	U User table
ESEC_ETL	ADV_SUBALERT	193 SELECT	U User table
ESEC_ETL	ADV_URGENCY	193 SELECT	U User table
ESEC_ETL	ADV_VENDOR	193 SELECT	U User table
ESEC_ETL	ADV_VULN_PRODUCT	193 SELECT	U User table
ESEC_ETL	ANNOTATIONS	193 SELECT	U User table
ESEC_ETL	ASSET	193 SELECT	U User table
ESEC_ETL	ASSET_CTGRY	193 SELECT	U User table
ESEC_ETL	ASSET_HOSTNAME	193 SELECT	U User table
ESEC_ETL	ASSET_IP	193 SELECT	U User table
ESEC_ETL	ASSET_LOC	193 SELECT	U User table
ESEC_ETL	ASSET_VAL_LKUP	193 SELECT	U User table

Role Name	Object Name	Action	Type
ESEC_ETL	ASSET_X_ENTITY_X_ROLE	193 SELECT	U User table
ESEC_ETL	ASSOCIATIONS	193 SELECT	U User table
ESEC_ETL	ATTACHMENTS	193 SELECT	U User table
ESEC_ETL	AUDIT_RECORD	193 SELECT	U User table
ESEC_ETL	CONFIGS	193 SELECT	U User table
ESEC_ETL	CONTACTS	193 SELECT	U User table
ESEC_ETL	CORR_ACT_DEF	193 SELECT	U User table
ESEC_ETL	CORR_ACT_META	193 SELECT	U User table
ESEC_ETL	CORR_ACT_PARM	193 SELECT	U User table
ESEC_ETL	CORR_ACT_PARM_DEF	193 SELECT	U User table
ESEC_ETL	CORR_DEPLOY_CONFIG	193 SELECT	U User table
ESEC_ETL	CORR_ENGINE_CONFIG	193 SELECT	U User table
ESEC_ETL	CORR_RULE	193 SELECT	U User table
ESEC_ETL	CORR_RULE_CFG	193 SELECT	U User table
ESEC_ETL	CORRELATED_EVENTS_P_MAX	193 SELECT	U User table
ESEC_ETL	CORRELATED_EVENTS_P_MIN	193 SELECT	U User table
ESEC_ETL	CRIT_LKUP	193 SELECT	U User table
ESEC_ETL	CUST	193 SELECT	U User table
ESEC_ETL	CUST_HIERARCHY	193 SELECT	U User table
ESEC_ETL	ENTITY_TYP_LKUP	193 SELECT	U User table
ESEC_ETL	ENV_IDENTITY_LKUP	193 SELECT	U User table
ESEC_ETL	ESEC_ARCHIVE_CONFIG	193 SELECT	U User table
ESEC_ETL	ESEC_ARCHIVE_LOG_FILES	193 SELECT	U User table
ESEC_ETL	ESEC_ARCHIVE_LOGS	193 SELECT	U User table
ESEC_ETL	ESEC_DB_PATCHES	193 SELECT	U User table
ESEC_ETL	ESEC_DB_VERSION	193 SELECT	U User table
ESEC_ETL	ESEC_DISPLAY	193 SELECT	U User table
ESEC_ETL	ESEC_JOB_CONFIG	193 SELECT	U User table
ESEC_ETL	ESEC_JOB_STS	193 SELECT	U User table
ESEC_ETL	ESEC_NAMESPACE	193 SELECT	U User table
ESEC_ETL	ESEC_NAMESPACE_LEAF	193 SELECT	U User table
ESEC_ETL	ESEC_PARTITION_CONFIG	193 SELECT	U User table
ESEC_ETL	ESEC_PORT_REFERENCE	193 SELECT	U User table
ESEC_ETL	ESEC_PROTOCOL_REFERENCE	193 SELECT	U User table
ESEC_ETL	ESEC_SDM_LOCK	193 SELECT	U User table
ESEC_ETL	ESEC_SEQUENCE	193 SELECT	U User table
ESEC_ETL	ESEC_TABLE_GROUPS	193 SELECT	U User table
ESEC_ETL	ESEC_UUID_UUID_ASSOC	193 SELECT	U User table
ESEC_ETL	EVENTS_P_MAX	193 SELECT	U User table
ESEC_ETL	EVENTS_P_MIN	193 SELECT	U User table
ESEC_ETL	EVT_AGENT	193 SELECT	U User table
ESEC_ETL	EVT_ASSET	193 SELECT	U User table
ESEC_ETL	EVT_DEST_EVT_NAME_SMRY_1_P_MAX	193 SELECT	U User table
ESEC_ETL	EVT_DEST_EVT_NAME_SMRY_1_P_MAX	195 INSERT	U User table
ESEC_ETL	EVT_DEST_EVT_NAME_SMRY_1_P_MAX	196 DELETE	U User table
ESEC_ETL	EVT_DEST_EVT_NAME_SMRY_1_P_MAX	197 UPDATE	U User table
ESEC_ETL	EVT_DEST_EVT_NAME_SMRY_1_P_MIN	193 SELECT	U User table
ESEC_ETL	EVT_DEST_SMRY_1_P_MAX	193 SELECT	U User table
ESEC_ETL	EVT_DEST_SMRY_1_P_MAX	195 INSERT	U User table

Role Name	Object Name	Action	Type
ESEC_ETL	EVT_DEST_SMRY_1_P_MAX	196 DELETE	U User table
ESEC_ETL	EVT_DEST_SMRY_1_P_MAX	197 UPDATE	U User table
ESEC_ETL	EVT_DEST_SMRY_1_P_MIN	193 SELECT	U User table
ESEC_ETL	EVT_DEST_TXNMY_SMRY_1_P_MAX	193 SELECT	U User table
ESEC_ETL	EVT_DEST_TXNMY_SMRY_1_P_MAX	195 INSERT	U User table
ESEC_ETL	EVT_DEST_TXNMY_SMRY_1_P_MAX	196 DELETE	U User table
ESEC_ETL	EVT_DEST_TXNMY_SMRY_1_P_MAX	197 UPDATE	U User table
ESEC_ETL	EVT_DEST_TXNMY_SMRY_1_P_MIN	193 SELECT	U User table
ESEC_ETL	EVT_NAME	193 SELECT	U User table
ESEC_ETL	EVT_NAME	195 INSERT	U User table
ESEC_ETL	EVT_NAME	196 DELETE	U User table
ESEC_ETL	EVT_NAME	197 UPDATE	U User table
ESEC_ETL	EVT_PORT_SMRY_1_P_MAX	193 SELECT	U User table
ESEC_ETL	EVT_PORT_SMRY_1_P_MAX	195 INSERT	U User table
ESEC_ETL	EVT_PORT_SMRY_1_P_MAX	196 DELETE	U User table
ESEC_ETL	EVT_PORT_SMRY_1_P_MAX	197 UPDATE	U User table
ESEC_ETL	EVT_PORT_SMRY_1_P_MIN	193 SELECT	U User table
ESEC_ETL	EVT_PRTCL	193 SELECT	U User table
ESEC_ETL	EVT_RSRC	193 SELECT	U User table
ESEC_ETL	EVT_SEV_SMRY_1_P_MAX	193 SELECT	U User table
ESEC_ETL	EVT_SEV_SMRY_1_P_MAX	195 INSERT	U User table
ESEC_ETL	EVT_SEV_SMRY_1_P_MAX	196 DELETE	U User table
ESEC_ETL	EVT_SEV_SMRY_1_P_MAX	197 UPDATE	U User table
ESEC_ETL	EVT_SEV_SMRY_1_P_MIN	193 SELECT	U User table
ESEC_ETL	EVT_SRC	193 SELECT	U User table
ESEC_ETL	EVT_SRC_COLLECTOR	193 SELECT	U User table
ESEC_ETL	EVT_SRC_GRP	193 SELECT	U User table
ESEC_ETL	EVT_SRC_MGR	193 SELECT	U User table
ESEC_ETL	EVT_SRC_OFFSET	193 SELECT	U User table
ESEC_ETL	EVT_SRC_SMRY_1_P_MAX	193 SELECT	U User table
ESEC_ETL	EVT_SRC_SMRY_1_P_MAX	195 INSERT	U User table
ESEC_ETL	EVT_SRC_SMRY_1_P_MAX	196 DELETE	U User table
ESEC_ETL	EVT_SRC_SMRY_1_P_MAX	197 UPDATE	U User table
ESEC_ETL	EVT_SRC_SMRY_1_P_MIN	193 SELECT	U User table
ESEC_ETL	EVT_SRC_SRVR	193 SELECT	U User table
ESEC_ETL	EVT_TXNMY	193 SELECT	U User table
ESEC_ETL	EVT_USR	193 SELECT	U User table
ESEC_ETL	EVT_USR	195 INSERT	U User table
ESEC_ETL	EVT_USR	196 DELETE	U User table
ESEC_ETL	EVT_USR	197 UPDATE	U User table
ESEC_ETL	EXT_DATA	193 SELECT	U User table
ESEC_ETL	HIST_CORRELATED_EVENTS_P_MAX	193 SELECT	U User table
ESEC_ETL	HIST_EVENTS_P_MAX	193 SELECT	U User table
ESEC_ETL	IMAGES	193 SELECT	U User table
ESEC_ETL	INCIDENTS	193 SELECT	U User table
ESEC_ETL	INCIDENTS_ASSETS	193 SELECT	U User table
ESEC_ETL	INCIDENTS_EVENTS	193 SELECT	U User table
ESEC_ETL	INCIDENTS_VULN	193 SELECT	U User table
ESEC_ETL	L_STAT	193 SELECT	U User table

Role Name	Object Name	Action	Type
ESEC_ETL	LOGS	193 SELECT	U User table
ESEC_ETL	MD_CONFIG	193 SELECT	U User table
ESEC_ETL	MD_EVT_FILE_STS	193 SELECT	U User table
ESEC_ETL	MD_EVT_FILE_STS	195 INSERT	U User table
ESEC_ETL	MD_EVT_FILE_STS	196 DELETE	U User table
ESEC_ETL	MD_EVT_FILE_STS	197 UPDATE	U User table
ESEC_ETL	MD_SMRY_STS	193 SELECT	U User table
ESEC_ETL	MD_SMRY_STS	195 INSERT	U User table
ESEC_ETL	MD_SMRY_STS	196 DELETE	U User table
ESEC_ETL	MD_SMRY_STS	197 UPDATE	U User table
ESEC_ETL	MD_VIEW_CONFIG	193 SELECT	U User table
ESEC_ETL	MSSP_ASSOCIATIONS	193 SELECT	U User table
ESEC_ETL	NETWORK_IDENTITY_LKUP	193 SELECT	U User table
ESEC_ETL	NLS_CONFIG	193 SELECT	U User table
ESEC_ETL	NLS_MSG_TRANSLATION	193 SELECT	U User table
ESEC_ETL	NORM_ATTACK_CD_MAP	193 SELECT	U User table
ESEC_ETL	OBJ_STORE	193 SELECT	U User table
ESEC_ETL	OFFLINE_QRY_STS	193 SELECT	U User table
ESEC_ETL	ORGANIZATION	193 SELECT	U User table
ESEC_ETL	PERSON	193 SELECT	U User table
ESEC_ETL	PHYSICAL_ASSET	193 SELECT	U User table
ESEC_ETL	PRDT	193 SELECT	U User table
ESEC_ETL	ROLE_LKUP	193 SELECT	U User table
ESEC_ETL	RPT_TRANSLATION	193 SELECT	U User table
ESEC_ETL	SENSITIVITY_LKUP	193 SELECT	U User table
ESEC_ETL	SENTINEL	193 SELECT	U User table
ESEC_ETL	SENTINEL_HOST	193 SELECT	U User table
ESEC_ETL	SENTINEL_PLUGIN	193 SELECT	U User table
ESEC_ETL	STATES	193 SELECT	U User table
ESEC_ETL	TXNMY_NODE	193 SELECT	U User table
ESEC_ETL	USERS	193 SELECT	U User table
ESEC_ETL	VNDR	193 SELECT	U User table
ESEC_ETL	VULN	193 SELECT	U User table
ESEC_ETL	VULN_CODE	193 SELECT	U User table
ESEC_ETL	VULN_INFO	193 SELECT	U User table
ESEC_ETL	VULN_RSRC	193 SELECT	U User table
ESEC_ETL	VULN_RSRC_SCAN	193 SELECT	U User table
ESEC_ETL	VULN_SCAN	193 SELECT	U User table
ESEC_ETL	VULN_SCAN_VULN	193 SELECT	U User table
ESEC_ETL	VULN_SCANNER	193 SELECT	U User table
ESEC_ETL	WORKFLOW_DEF	193 SELECT	U User table
ESEC_ETL	WORKFLOW_INFO	193 SELECT	U User table

ESEC_USER

Role Name	Object Name	Action	Type
ESEC_USER	ADV_ALERT_CVE_RPT_V	193 SELECT	V View
ESEC_USER	ADV_ALERT_PRODUCT_RPT_V	193 SELECT	V View
ESEC_USER	ADV_ALERT_RPT_V	193 SELECT	V View
ESEC_USER	ADV_ATTACK_ALERT_RPT_V	193 SELECT	V View

Role Name	Object Name	Action	Type
ESEC_USER	ADV_ATTACK_CVE_RPT_V	193 SELECT	V View
ESEC_USER	ADV_ATTACK_PLUGIN_RPT_V	193 SELECT	V View
ESEC_USER	ADV_ATTACK_RPT_V	193 SELECT	V View
ESEC_USER	ADV_CREDIBILITY_RPT_V	193 SELECT	V View
ESEC_USER	ADV_FEED_RPT_V	193 SELECT	V View
ESEC_USER	ADV_PRODUCT_RPT_V	193 SELECT	V View
ESEC_USER	ADV_PRODUCT_SERVICE_PACK_RPT_V	193 SELECT	V View
ESEC_USER	ADV_PRODUCT_VERSION_RPT_V	193 SELECT	V View
ESEC_USER	ADV_SEVERITY_RPT_V	193 SELECT	V View
ESEC_USER	ADV_SUBALERT_RPT_V	193 SELECT	V View
ESEC_USER	ADV_URGENCY_RPT_V	193 SELECT	V View
ESEC_USER	ADV_VENDOR_RPT_V	193 SELECT	V View
ESEC_USER	ADV_VULN_PRODUCT_RPT_V	193 SELECT	V View
ESEC_USER	ANNOTATIONS_RPT_V	193 SELECT	V View
ESEC_USER	ASSET_CATEGORY_RPT_V	193 SELECT	V View
ESEC_USER	ASSET_HOSTNAME_RPT_V	193 SELECT	V View
ESEC_USER	ASSET_IP_RPT_V	193 SELECT	V View
ESEC_USER	ASSET_LOCATION_RPT_V	193 SELECT	V View
ESEC_USER	ASSET_RPT_V	193 SELECT	V View
ESEC_USER	ASSET_VALUE_RPT_V	193 SELECT	V View
ESEC_USER	ASSET_X_ENTITY_X_ROLE_RPT_V	193 SELECT	V View
ESEC_USER	ASSOCIATIONS_RPT_V	193 SELECT	V View
ESEC_USER	ATTACHMENTS_RPT_V	193 SELECT	V View
ESEC_USER	CONFIGS_RPT_V	193 SELECT	V View
ESEC_USER	CONTACTS_RPT_V	193 SELECT	V View
ESEC_USER	CORRELATED_EVENTS	193 SELECT	V View
ESEC_USER	CORRELATED_EVENTS_RPT_V	193 SELECT	V View
ESEC_USER	CORRELATED_EVENTS_RPT_V1	193 SELECT	V View
ESEC_USER	CRITICALITY_RPT_V	193 SELECT	V View
ESEC_USER	CUST_HIERARCHY_V	193 SELECT	V View
ESEC_USER	CUST_RPT_V	193 SELECT	V View
ESEC_USER	ENTITY_TYPE_RPT_V	193 SELECT	V View
ESEC_USER	ENV_IDENTITY_RPT_V	193 SELECT	V View
ESEC_USER	ESEC_DISPLAY_RPT_V	193 SELECT	V View
ESEC_USER	ESEC_PORT_REFERENCE_RPT_V	193 SELECT	V View
ESEC_USER	ESEC_PROTOCOL_REFERENCE_RPT_V	193 SELECT	V View
ESEC_USER	ESEC_SEQUENCE_RPT_V	193 SELECT	V View
ESEC_USER	esec_check_patch	224 EXECUTE	FN Function
ESEC_USER	get_string	224 EXECUTE	FN Function
ESEC_USER	esec_toBase	224 EXECUTE	FN Function
ESEC_USER	esec_toDecimal	224 EXECUTE	FN Function
ESEC_USER	esec_toIpChar	224 EXECUTE	FN Function
ESEC_USER	esec_toIpNum	224 EXECUTE	FN Function
ESEC_USER	getAlertId	224 EXECUTE	FN Function
ESEC_USER	getCve	224 EXECUTE	FN Function
ESEC_USER	isArchived	224 EXECUTE	FN Function
ESEC_USER	getArchSeq	224 EXECUTE	FN Function
ESEC_USER	fn_hex_to_char	224 EXECUTE	FN Function
ESEC_USER	esec_get_next_partition_name	224 EXECUTE	FN Function

Role Name	Object Name	Action	Type
ESEC_USER	isSQL2005	224 EXECUTE	FN Function
ESEC_USER	EVENTS	193 SELECT	V View
ESEC_USER	EVENTS_ALL_RPT_V	193 SELECT	V View
ESEC_USER	EVENTS_ALL_RPT_V1	193 SELECT	V View
ESEC_USER	EVENTS_ALL_V	193 SELECT	V View
ESEC_USER	EVENTS_RPT_V	193 SELECT	V View
ESEC_USER	EVENTS_RPT_V1	193 SELECT	V View
ESEC_USER	EVENTS_RPT_V2	193 SELECT	V View
ESEC_USER	EVT_AGENT_RPT_V	193 SELECT	V View
ESEC_USER	EVT_ASSET_RPT_V	193 SELECT	V View
ESEC_USER	EVT_DEST_EVT_NAME_SMRY_1	193 SELECT	V View
ESEC_USER	EVT_DEST_EVT_NAME_SMRY_1_RPT_V	193 SELECT	V View
ESEC_USER	EVT_DEST_SMRY_1	193 SELECT	V View
ESEC_USER	EVT_DEST_SMRY_1_RPT_V	193 SELECT	V View
ESEC_USER	EVT_DEST_TXNMY_SMRY_1	193 SELECT	V View
ESEC_USER	EVT_DEST_TXNMY_SMRY_1_RPT_V	193 SELECT	V View
ESEC_USER	EVT_NAME_RPT_V	193 SELECT	V View
ESEC_USER	EVT_PORT_SMRY_1	193 SELECT	V View
ESEC_USER	EVT_PORT_SMRY_1_RPT_V	193 SELECT	V View
ESEC_USER	EVT_PRTCL_RPT_V	193 SELECT	V View
ESEC_USER	EVT_RSRC_RPT_V	193 SELECT	V View
ESEC_USER	EVT_SEV_SMRY_1	193 SELECT	V View
ESEC_USER	EVT_SEV_SMRY_1_RPT_V	193 SELECT	V View
ESEC_USER	EVT_SRC_SMRY_1	193 SELECT	V View
ESEC_USER	EVT_SRC_SMRY_1_RPT_V	193 SELECT	V View
ESEC_USER	EVT_TXNMY_RPT_V	193 SELECT	V View
ESEC_USER	EVT_USR_RPT_V	193 SELECT	V View
ESEC_USER	EXTERNAL_DATA_RPT_V	193 SELECT	V View
ESEC_USER	HIST_CORRELATED_EVENTS	193 SELECT	V View
ESEC_USER	HIST_CORRELATED_EVENTS_RPT_V	193 SELECT	V View
ESEC_USER	HIST_EVENTS	193 SELECT	V View
ESEC_USER	HIST_EVENTS_RPT_V	193 SELECT	V View
ESEC_USER	HIST_EVT_DEST_EVT_NAME_SMRY_1	193 SELECT	V View
ESEC_USER	HIST_EVT_DEST_SMRY_1	193 SELECT	V View
ESEC_USER	HIST_EVT_DEST_TXNMY_SMRY_1	193 SELECT	V View
ESEC_USER	HIST_EVT_PORT_SMRY_1	193 SELECT	V View
ESEC_USER	HIST_EVT_SEV_SMRY_1	193 SELECT	V View
ESEC_USER	HIST_EVT_SRC_SMRY_1	193 SELECT	V View
ESEC_USER	IMAGES_RPT_V	193 SELECT	V View
ESEC_USER	INCIDENTS_ASSETS_RPT_V	193 SELECT	V View
ESEC_USER	INCIDENTS_EVENTS_RPT_V	193 SELECT	V View
ESEC_USER	INCIDENTS_RPT_V	193 SELECT	V View
ESEC_USER	INCIDENTS_VULN_RPT_V	193 SELECT	V View
ESEC_USER	L_STAT_RPT_V	193 SELECT	V View
ESEC_USER	LOGS_RPT_V	193 SELECT	V View
ESEC_USER	MSSP_ASSOCIATIONS_V	193 SELECT	V View
ESEC_USER	NETWORK_IDENTITY_RPT_V	193 SELECT	V View
ESEC_USER	ORGANIZATION_RPT_V	193 SELECT	V View
ESEC_USER	PERSON_RPT_V	193 SELECT	V View

Role Name	Object Name	Action	Type
ESEC_USER	PHYSICAL_ASSET_RPT_V	193 SELECT	V View
ESEC_USER	PRODUCT_RPT_V	193 SELECT	V View
ESEC_USER	ROLE_RPT_V	193 SELECT	V View
ESEC_USER	RPT_LABELS_RPT_V	193 SELECT	V View
ESEC_USER	SENSITIVITY_RPT_V	193 SELECT	V View
ESEC_USER	STATES_RPT_V	193 SELECT	V View
ESEC_USER	UNASSIGNED_INCIDENTS_RPT_V	193 SELECT	V View
ESEC_USER	USERS_RPT_V	193 SELECT	V View
ESEC_USER	VENDOR_RPT_V	193 SELECT	V View
ESEC_USER	VULN_CALC_SEVERITY_RPT_V	193 SELECT	V View
ESEC_USER	VULN_CODE_RPT_V	193 SELECT	V View
ESEC_USER	VULN_INFO_RPT_V	193 SELECT	V View
ESEC_USER	VULN_RPT_V	193 SELECT	V View
ESEC_USER	VULN_RSRC_RPT_V	193 SELECT	V View
ESEC_USER	VULN_RSRC_SCAN_RPT_V	193 SELECT	V View
ESEC_USER	VULN_SCAN_RPT_V	193 SELECT	V View
ESEC_USER	VULN_SCAN_VULN_RPT_V	193 SELECT	V View
ESEC_USER	VULN_SCANNER_RPT_V	193 SELECT	V View

Sentinel Server Roles

Server Role	Description	Sentinel User
sysadmin	System Administrators	esecdba
securityadmin	Security Administrators	esecapp
serveradmin	Server Administrators	esecdba
setupadmin	Setup Administrators	
processadmin	Process Administrators	
diskadmin	Disk Administrators	
dbcreator	Database Creators	
bulkadmin	Bulk Insert Administrators	

Windows Domain Authentication DB users and permissions

A domain user will be associated with esecadm, esecapp, esecdba and esecrpt user according to the configuration at install time. Those domain users will have the same privilege as specified by the previous sections.

Note: The installer takes care of the database user permissions

E Sentinel Log Locations

The purpose of this document is to provide information of the log file locations for the following components of Sentinel.

- Sentinel Data Manager
- iTRAC
- Advisor
- Event Insertion
- Database Queries
- Active ViewsAggregation
- Wrapper (formerly Sentinel Watchdog)
- Collector Manager
- Correlation
- Sentinel Control Center
- DAS Proxy

The naming convention for the log files is that they include with the name of the process, the instance number (almost always 0 unless there are multiple instances of das_binary installed), and the log number in the log rotation sequence. For examples, see below.

Sentinel Data Manager

Logs activities executed using Sentinel Data Manager for the specific client running on that machine.

For Windows:

```
%ESEC_HOME%\log\SDM0.*.log
```

For UNIX:

```
$ESEC_HOME/log/SDM0.*.log
```

iTRAC

Logs activities related to iTRAC.

For Windows:

```
%ESEC_HOME%\log\das_itrac0.*.log
%ESEC_HOME%\log\itrac_engine.log
```

For UNIX:

```
$ESEC_HOME/log/das_itrac0.*.log
$ESEC_HOME/log/itrac_engine.log
```

Advisor

Logs activities related to advisor data download and process.

For Windows:

```
%ESEC_HOME%\log\advisor_script.log
%ESEC_HOME%\log\advisor0.*.log
```

For UNIX:

```
$ESEC_HOME/log/advisor_script.log
$ESEC_HOME/log/advisor0.*.log
```

Event Insertion

Logs activities related to event insertion into the database.

For Windows:

```
%ESEC_HOME%\log\das_binary0.*.log
```

For UNIX:

```
$ESEC_HOME/log/das_binary0.*.log
```

Database Queries

Logs activities related to database queries, Collector, Collector manager health, and all other DAS activities not performed by other DAS components.

For Windows:

```
%ESEC_HOME%\log\das_query0.*.log
```

For UNIX:

```
$ESEC_HOME/log/das_query0.*.log
```

Active Views

Logs activities related to Active Views.

For Windows:

```
%ESEC_HOME%\log\das_rt0.*.log
```

For UNIX:

```
$ESEC_HOME/log/das_rt0.*.log
```

Aggregation

Logs activities related to aggregation.

For Windows:

```
%ESEC_HOME%\log\das_aggregation0.*.log
```

For UNIX:

```
$ESEC_HOME/log/das_aggregation0.*.log
```

Wrapper

Logs activities related to Wrapper.

NOTE: sentinel_wrapper.log is for the service wrapper.

For Windows:

```
%ESEC_HOME%\log\sentinel0.*.log
%ESEC_HOME%\log\sentinel_wrapper.log
```

For UNIX:

```
$ESEC_HOME/log/sentinel0.*.log
$ESEC_HOME/log/sentinel_wrapper.log
```

Collector Manager

Logs activities related to Collector Manager.

For Windows:

```
%ESEC_HOME%\log\collector-mgr0.*.log
```

For UNIX:

```
$ESEC_HOME/log/collector-mgr0.*.log
```

Correlation Engine

Logs activities related to Correlation Engine.

For Windows:

```
%ESEC_HOME%\log\correlation-engine0.*.log
```

For UNIX:

```
$ESEC_HOME/log/correlation-engine0.*.log
```

Sentinel Control Center

Logs activities related to the Sentinel Control Center.

For Windows:

```
%ESEC_HOME%\log\control_center0.*.log
```

For UNIX:

```
$ESEC_HOME/log/control-center0.*.log
```

DAS Proxy

Logs activities related to Proxy Communication.

For Windows:

```
%ESEC_HOME%\log\das_proxy0.*.log
```

For UNIX:

```
$ESEC_HOME/log/das_proxy0.*.log
```

Multiple Instances

In some environments, there may be multiple instances of a process running, such as DAS Binary, the Sentinel Control Center, or Sentinel Data Manager. If so, the first instance's log files would be named as described above (For example, das_binary0.0.log). The second instance would substitute a 1 for the first 0 in the log file name (For example, das_binary1.0.log).

If other processes have log files indicating that more than one instance is running, that could indicate a system problem.

Glossary

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

See Collector

Agent Builder

See Collector Builder

Agent Engine

See Collector Engine

Agent Manager

Collector Manager

Aggregation and Event Normalization

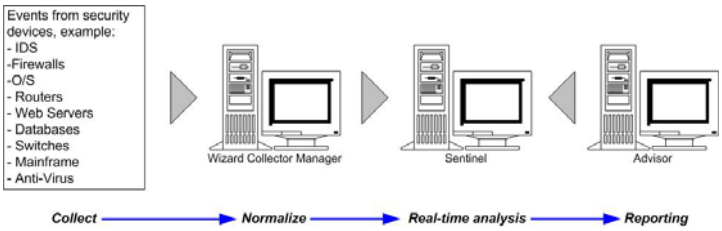
Aggregation is the process of taking individual low-relevant data items and combining them, resulting in a data item that could possibly be high importance. The individual parts of event, such as the event name, event date, Source IP, Destination IP, UUID, Sensor Type and so on, by themselves may not 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 Sentinel 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.
Collector Host	Any machine that has the Collector Manager software installed.

Collector

A Collector 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.



There are three levels of Collectors, they are:

- Supported Collectors (T1)
- Documented Collectors (T2)
- Sample Collectors (T3)

Collectors are made of:

- template files
- parameter files
- lookup files
- mapping files

Collector Builder

A GUI that allows you to create rules based Collectors to collect, filter and normalize data from many different sources and securely communicate relevant information to the Sentinel Server that can be used to monitor traffic.

Collector Engine

Processes the template logic for each port. A Collector engine runs a corresponding port.

Collector Manager

A back-end that manages Collectors and system status messages.

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.

Sentinel 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 Collector 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 Collector Manager and requests to retrieve and store configuration information.
das_aggregation.xml	Used for aggregation operation.
das_binary.xml	Used for event and correlated event insertion operation.
das_itrac.xml	used for executing and configuring the activity service and for configuring the workflow service
das_query.xml	Specifies configuration parameters for the Data Access Service (DAS), a component of Sentinel 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

Sentinel 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:

- Collector - done through the script using the Collector builder
- global filter - Applied equally to all events generated by all Collectors 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

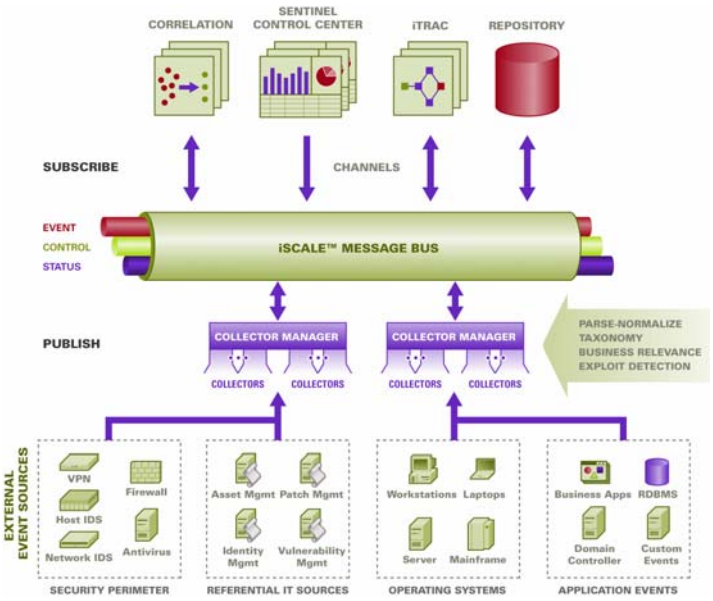
A grouping a set of events together as a whole representing something of interest (group of similar events or set of different events that indicate a pattern of interest such an attack).

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. Sentinel 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 Collectors, 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.

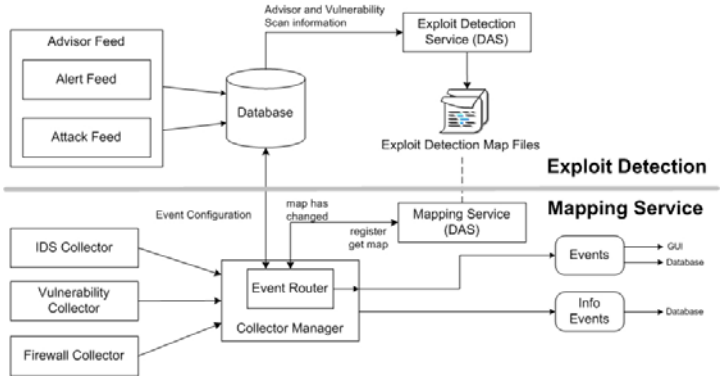
Optionally, parsing commands may be associated with the match string. The parsing commands are executed if a match is found.

Mapping Files

For Collectors, Mapping files are optional files (.csv) that allow for fast lookup of key entries. The csv file is a relative path from a Collector's script directory. The editing of these files is currently not within Collector Builder, but the files can be edited using Excel.

Mapping Service

Sentinel'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.



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 Collector processing.

meta-tag

Meta-tags store meta data.

Parameter Files

For Collectors, 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 Collector'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 Collectors list.

parsing command

In Collectors, a high-level scripting interface that allows manipulation of data. Parsing is the process of breaking an event down into its components.

Port

Ports enable a Collector 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 Collector 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 Collector 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

A compiled file (*.asd) that is made up of the Collector 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 Collectors, 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 Collectors from Collector 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 a Collector to locate Collector 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 Collector 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
- Collector 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 Collectors, 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.
workflow	See iTRAC™.