



Cisco Info Mediator Reference

This appendix contains reference information for Cisco Info Mediators, including:

- [Common Cisco Info Mediator Properties and Command Line Options](#), page E-1
- [Cisco Info Mediator Rules File Processing](#), page E-6
- [Troubleshooting Cisco Info Mediators](#), page E-30
- [Cisco Info Mediator Information and Error Messages](#), page E-36.

The information in this appendix applies to all Cisco Info Mediators and TSMs. For introductory information about Cisco Info Mediators, see [Chapter 5, “Info Mediators.”](#)

For more information about specific Cisco Info Mediators, see the individual guides available for each Cisco Info Mediator on the Cisco Systems Support Site.

Common Cisco Info Mediator Properties and Command Line Options

This section describes the properties and command line options common to all Cisco Info Mediators. For the properties and command line options specific to a particular Cisco Info Mediator, see the individual guides for each Cisco Info Mediator available on the Cisco Systems Support Site.

Editing Cisco Info Mediator Properties

You can edit Cisco Info Mediator properties using a text editor or the Properties Editor. The Properties Editor enables you to edit multiple properties files, validates changes you make, and saves the file in the correct format in the correct directory. The Properties Editor is described in [Properties Editor](#), page 3-35.

You can edit the properties file while the Cisco Info Mediator is running. If you do this, the changes you make will take effect the next time you run the Cisco Info Mediator. For an introduction to Cisco Info Mediator properties, see [Properties File](#), page 5-6.

How Properties and Command Line Options Are Processed at Startup

There are default values for each property. In an unedited properties file, all properties are set to the default values, and are commented out with a hash symbol (#) to the beginning of the line.

If you change a setting in the properties file and remove the hash symbol, this overrides the default.

If you change a setting on the command line, this overrides both the default value and the setting in the properties file. To avoid errors, add as many properties as possible to the properties file rather than using the command line options.

If **-name** is specified, its value determines the name used for the properties file (`$OMNIHOME/probes/arch/name.props`), rules file (`$OMNIHOME/probes/arch/name.rules`), and log file (`$OMNIHOME/log/name.log`).

If **-propsfile** is specified, its value overrides the name setting for the properties file.

[Table E-1](#) lists the common properties and command line options available to all Cisco Info Mediators, and their default settings.

Table E-1 Common Cisco Info Mediator Properties and Command Line Options

Property	Command Line Option	Description
AuthPassword <string>	N/A	The password associated with the username used to authenticate the Cisco Info Mediator when it connects to a Cisco Info Server running in secure mode. This password must be encrypted with the nco_crypt utility. Secure mode is described in Secure Mode, page 5-10 .
AuthUserName <string>	N/A	A username used to authenticate the Cisco Info Mediator when it connects to a Cisco Info Server running in secure mode. Secure mode is described in Secure Mode, page 5-10 .
AutoSAF <integer>	-autosaf -noautosaf	Turns on automatic store and forward mode. By default, automatic store and forward mode is not enabled. Store and forward mode is described in Store and Forward Mode, page 5-9 .
Buffering <boolean>	-buffer -nobuffer	Specifies whether or not to use buffering when sending alerts. By default, buffering is not enabled.
BufferSize <integer>	-buffersize <integer>	Number of alerts to buffer. The default is 10 .
N/A	-help	Displays the supported command line options and exits.

Table E-1 Common Cisco Info Mediator Properties and Command Line Options (continued)

Property	Command Line Option	Description
LookupTableMode <integer>	-lookupmode <integer>	<p>Specifies how table lookups are performed. It can be set to 1, 2, or 3. The default is 1.</p> <p>If set to 1, all external lookup tables are assumed to have a single value column. Tabs are not used as column delimiters.</p> <p>If set to 2, all external lookup tables are assumed to have multiple columns. If the number of columns on each line is not the same, an error is generated that includes the file name and the line on which the error occurred.</p> <p>If set to 3, the rules engine attempts to determine the number of columns in the external lookup table. An error is generated for each line that has a different column count from the previous line. The error includes the file name and the line on which the error occurred.</p> <p>Lookup tables are described in Lookup Table Operations, page E-20.</p>
Manager <string>	-manager <string>	Specifies the value of the Manager field for the alert.
MaxLogFileSize <integer>	N/A	Specifies the maximum size the log file can grow. The default is 1 Mbyte . Once the size of the log file reaches the size specified, a second log file is started. When the second file reaches the maximum size, the first file is overwritten with a new log file and the process starts again.
MaxRawFileSize <integer>	N/A	<p>Specifies the maximum size of the raw capture file in Kbytes. The default is unlimited (-1).</p> <p>If RawCaptureFileBackup is disabled, this property is ignored.</p> <p>If RawCaptureFileBackup is enabled, this specifies the approximate file size at which a new file will be started.</p> <p>Raw capture mode is described in Raw Capture Mode, page 5-10.</p>
MaxSAFFileSize <integer>	N/A	<p>Specifies the maximum size the store and forward file can grow to. The default is 1 Mbyte.</p> <p>Store and forward mode is described in Store and Forward Mode, page 5-9.</p>

Table E-1 Common Cisco Info Mediator Properties and Command Line Options (continued)

Property	Command Line Option	Description
MessageLevel <string>	-messagelevel <string>	Specifies the lowest level of messages to be written to the message log. Values in ascending order are: debug , info , warn , error , fatal . The default is warn .
MessageLog <string>	-messagelog <string>	Specifies the name of the file to which to send debug and other information messages. This can also be set to STDOUT and STDERR . The default is \$OMNIHOME/log/name.log .
MsgDailyLog	-msgdailylog	If enabled, the MsgDailyLog property turns on daily logging. By default, daily logging is not enabled. Because the time is checked regularly, when MsgDailyLog is set, there will be a slight reduction in performance.
MsgTimeLog	-msgtimelog <string>	Specifies the MsgTimeLog property specifies the time after which the daily log is created. It is set to 0000 (midnight) by default. If MsgDailyLog is not set, this property is ignored.
Name <string>	-name <string>	Specifies the name of the Cisco Info Mediator. If set on the command line, the properties file, rules file, and message log file used will be: \$OMNIHOME/probes/arch/name.props , \$OMNIHOME/probes/arch/name.rules , and \$OMNIHOME/log/name.log , respectively.
NetworkTimeout <integer>	-networktimeout <integer>	Specifies a time in seconds after which the connection to the Cisco Info Server will time out, should a network failure occur. The default value for this property is 0 , meaning that no time out occurs.
PollServer <integer>	-pollserver <integer>	If the Cisco Info Mediator is connected to a backup Cisco Info Server because a failover occurred, the Cisco Info Mediator will periodically attempt to reconnect to the primary Cisco Info Server. This property specifies the frequency in seconds at which the Cisco Info Mediator polls for the return of the primary Cisco Info Server. The default is 0 , meaning that no polling occurs. The BackupInfo Server property, used to designate a backup Cisco Info Server, is described in Specifying Cisco Info Server Properties , page 1-7.

Table E-1 Common Cisco Info Mediator Properties and Command Line Options (continued)

Property	Command Line Option	Description
N/A	-propsfile <string>	Specifies the name of the properties file. The default is \$OMNIHOME/probes/arch/name.props .
RawCapture <boolean>	-raw -noraw	Controls the raw capture mode. By default, raw capture mode is disabled. Raw capture mode is described in Raw Capture Mode, page 5-10 .
RawCaptureFile <string>	-capturefile <string>	Specifies the name of the raw capture file. The default is \$OMNIHOME/var/name.cap . Raw capture mode is described in Raw Capture Mode, page 5-10 .
RawCaptureFileAppend <boolean>	N/A	Specifies whether new data is appended to the existing raw capture file, instead of overwriting it. By default, the file is overwritten. Raw capture mode is described in Raw Capture Mode, page 5-10 .
RawCaptureFileBackup <boolean>	N/A	Specifies whether the raw capture file is backed up when it exceeds its maximum file size, determined by the MaxRawFileSize property. By default, the file is not backed up. Raw capture mode is described in Raw Capture Mode, page 5-10 .
RetryConnectionCount <integer>	N/A	Specifies the number of events the Cisco Info Mediator should process in store and forward mode before trying to reconnect to the Cisco Info Server. The default is 15 . Store and forward mode is described in Store and Forward Mode, page 5-9 .
RetryConnectionTimeout <integer>	N/A	Specifies the number of seconds the Cisco Info Mediator should process events in store and forward mode before trying to reconnect to the Cisco Info Server. The default is 30 .
RulesFile <string>	-rulesfile <string>	Specifies the name of the rules file. The default is: \$OMNIHOME/probes/arch/name.rules .
SAFFileName <string>	-saffilename <string>	Specifies the name of the store and forward file. The default is \$OMNIHOME/var/name.server.store In this file name, name is the name of the Cisco Info Mediator and server is the name of the target Cisco Info Server. Store and forward mode is described in Store and Forward Mode, page 5-9 .

Table E-1 Common Cisco Info Mediator Properties and Command Line Options (continued)

Property	Command Line Option	Description
Server <string>	-server <string>	Specifies the name of the Cisco Info Server that receives alerts. The default is NCOMS . You can specify more than one Cisco Info Server by using colons to separate the names, for example: Server: “ NCOMS:NCOMS_HIGH:NCOMS_LOW ”. This enables you to use multiple Cisco Info Servers in Cisco Info Mediator rules files, as described in Connecting to Multiple Cisco Info Servers , page E-15.
Server Backup <string>	N/A	Specifies a secondary Cisco Info Server should the primary Cisco Info Server connection fail. If NetworkTimeout is set and a virtual Cisco Info Server is in use, use ServerBackup to refer to your secondary Cisco Info Server.
StoreAndForward <boolean>	-saf -nosaf	Controls the store and forward operations. By default, store and forward mode is enabled . Store and forward mode is described in Store and Forward Mode , page 5-9.
N/A	-version	Displays version information and exits.

Cisco Info Mediator Rules File Processing

A Cisco Info Mediator takes an event stream and parses it into elements. Event elements are converted into Cisco Info Server fields using Cisco Info Mediator rules. The **Identifier** field, used by the Cisco Info Server for deduplication, is also created. The fields are then inserted into alerts.status table as an alert.

For introductory information about rules files, see [The Rules File](#), page 5-7. For information about the Identifier field, see [Creating a Unique Identifier](#), page 5-9.

This section describes rules file syntax. It contains the following sections:

- [Elements, Fields, Properties, and Arrays in Rules Files](#), page E-7
- [Conditional Statements in Rules Files](#), page E-9
- [Including Multiple Rules Files](#), page E-10
- [Rules File Functions and Operators](#), page E-10
- [Testing Rules Files](#), page E-27
- [Debugging Rules Files](#), page E-27
- [Rules File Examples](#), page E-27.

Changes made to the rules file require the Cisco Info Mediator to be restarted or made to re-read the rules file (the latter is preferable, because the Cisco Info Mediator will not lose events). You can force the Cisco Info Mediator to re-read the rules file by issuing a **kill -HUP <pid>** command on the Cisco Info Mediator process ID (PID). See the **ps** and **kill man** pages for more information.

Elements, Fields, Properties, and Arrays in Rules Files

A Cisco Info Mediator splits the event stream into elements, indicated by the **\$** symbol in the rules file. For example, **\$Node** is an element containing the node name of the event source. You can assign elements to Cisco Info Server fields, indicated by the **@** symbol in the rules file.

Assigning Values to Cisco Info Server Fields

You can assign values to Cisco Info Server fields in the following ways:

- Direct assignment, for example: **@Node=\$Node**
- Concatenation, for example: **@Summary=\$Summary + \$Group**
- Adding text, for example: **@Summary=\$Node + "has problem" + \$Summary**

Numeric values can be expressed in decimal or hexadecimal form. The following statements, which set the **Class** field to **100**, are equivalent:

- **@Class=100**
- **@Class=0x64**

In addition to assigning elements to fields, you can use the processing statements, operators, and functions described in this chapter to manipulate these values in rules files before assigning them. Elements are stored as strings, so you need to use the **int** function, described in [Math Functions, page E-18](#), to convert them into integers before performing numeric operations.

Assigning Temporary Elements in Rules Files

You can create a temporary element in a rules file by assigning it to an expression, for example:

```
$tempelement = "message"
```

An element, **tempelement**, is created and assigned the string value **message**.

If you refer to an element that has not been initialized in this way, the element is set to the *null string* (**"**).

To create the element **\$b** and set it to **setnow**:

```
$b="setnow"
```

To then set the element **\$a** to **setnow**:

```
$a=$b
```

In the following example, temporary elements are used to extract information from the **Summary** element, which has the string value: **The Port is down on Port 1 Board 2**.

```
$temp1 = extract ($Summary, "Port ([0-9]+)")
$temp2 = extract ($Summary, "Board ([0-9]+)")
@AlertKey = $temp1 + "." + $temp2
```

The **extract** function is used to assign values to temporary elements **temp1** and **temp2**. Then these elements are concatenated with a **.** separating them, and assigned to the **AlertKey** field. After these statements are executed, the **AlertKey** field has the value **1.2**. The extract function is described in [String Functions, page E-16](#), and the concatenate operator (+) is described in [Math and String Operators, page E-13](#).

Assigning Properties

You can assign the value of a Cisco Info Mediator property, as defined in the properties file or on the command line, to a field value. A property is represented by a **%** symbol in the rules file. For example, you could add the following statement to your rules file:

```
@Summary = "Server = " + %Server
```

In this example, when the rules file is processed, the Cisco Info Mediator searches for a property named **Server**. If the property is found, its value is concatenated to the text string and assigned to the **Summary** field. If the property is not found, a *null string* ("") is assigned.

You can also assign values to a property in the rules file. If the property does not exist, it is created. For example, you could create a property called **Counter** to keep track of the number of events that have been processed as follows:

```
if (match(%Counter, ""))
{
  %Counter = 1
}
else {
  %Counter = int(%Counter) + 1
}
```

These properties retain their values across events and when the rules file is re-read.

Most Cisco Info Mediators read properties once at startup, so changing Cisco Info Mediator properties after startup does not usually affect Cisco Info Mediator behavior. However, the **RawCapture** property can be set in the rules file, so that you can send the raw event data to a file only when certain conditions are met.

```
# Top of rules processing
%RawCapture=0

if (condition) {
  # Want this event written to raw capture file.
  %RawCapture=1
}
```

The **IF** statement is described in [Conditional Statements in Rules Files, page E-9](#).

You can also enable raw capture mode by setting the **-raw** command line option or the **RawCapture** property in the Cisco Info Mediator properties file, as described in [Raw Capture Mode, page 5-10](#).

Using Arrays

You must define arrays at the start of a rules file, before any processing statements. (You must also define tables, described in [Lookup Table Operations, page E-20](#), before any processing statements.) To define an array, use the following syntax:

```
array node_arr;
```

Arrays are one dimensional. Each time an assignment is made for a key value that already exists, the previous value is overwritten. In the example:

```
node_arr["myhost"] = "a";
node_arr["yourhost"] = "b";
node_arr["myhost"] = "c";
```

After the preceding statements have been executed, there are two items in the **node_arr** array. The item with the key **myhost** is set to **c**, and the item with the key **yourhost** is set to **b**. You can make assignments using Cisco Info Mediator elements, for example:

```
node_arr[$Node] = "d";
```

**Note**

Array values are persistent until a Cisco Info Mediator is restarted; if you force the Cisco Info Mediator to re-read the rules file by issuing a **kill -HUP <pid>** command on the Cisco Info Mediator process ID, the array values are maintained.

Conditional Statements in Rules Files

The **IF** and **SWITCH** statements provide condition testing for processing elements in rules files.

The IF Statement

A condition is a combination of expressions and operations that resolve to either **TRUE** or **FALSE**. The **IF** statement allows conditional execution of a set of one or more assignment statements by executing only the rules for the condition that is **TRUE**. It has the following syntax:

```
if (<condition>) {
    rules
} [else if (<condition>) {
    rules
}...]
[else (<condition>) {
    rules
}]
```

You can combine conditions into increasingly complex conditions using the logical **AND** operator (**&&**), which is true only if all of its inputs are true, and **OR** operator (**||**), which is true if any of its inputs are true. For example:

```
if match ($Enterprise, "Acme") && match ($trap-type, "Link-Up") {
@Summary = "Acme Link Up on " + @Node
}
```

The operators used in this example are described in [Rules File Functions and Operators](#), page E-10.

The SWITCH Statement

A **SWITCH** statement transfers control to a set of one or more rules assignment statements depending on the value of an expression. It has the following syntax:

```
switch (expression) {
    case "stringliteral":
        rules
    case "stringliteral":
```

```

rules
...
default:
[rules]
}

```

The **SWITCH** statement tests for exact matches only. This statement should be used wherever possible in place of an **IF** statement because **SWITCH** statements can be processed more efficiently and therefore execute more quickly.

**Note**

Each **SWITCH** statement must contain a default case even if there are no rules associated with it. There is no **BREAK** clause for the **SWITCH** statement, so any rules in the **DEFAULT** case are executed if no other case is matched.

The expression can be any valid expression. For example:

```
switch($node)
```

The *stringliteral* can be any string value. For example:

```
case "jupiter":
```

You can have more than one stringliteral separated by the | operator. For example:

```
case "jupiter" | "mars" | "venus":
```

This case is executed if the expression matches any of the specified strings.

Including Multiple Rules Files

You can include a number of rules files in your main rules file using the include statement:

include "rulesfile"

Specify the path to the rules file as an absolute path. A relative path is relative to the Cisco Info Mediator working directory, which can vary depending on how the Cisco Info Mediator is started. You cannot use environment variables in the path.

```

if(match(@Manager, "ProbeWatch"))
{
    include "/opt/Omnibus/probes/solaris2/probewatch.rules"
}
else
...

```

Rules File Functions and Operators

You can use the operators and functions described in this section to manipulate elements in rules files before assigning them to Cisco Info Server fields.

[Table E-2](#) lists the operators described in the following sections.

Table E-2 Rules File Operators

Operators	Description	See...
*, /, -, +	Perform math and string operations.	Math and String Operators, page E-13.
&, , ^, >>, <<	Perform bitwise operations.	Bit Manipulation Operators, page E-13.
==, !=, <>, <, >, <=, >=	Perform comparison operations.	Comparison Operators, page E-14.
NOT (also !), AND (also &&), OR (also), XOR (also ^)	Perform logical (boolean) operations.	Logical Operators, page E-14.

Table E-3 lists the functions described in the following sections.

Table E-3 Rules File Functions

Function Name	Description	See...
datetotime	Converts a text string into UTC time.	Date and Time Functions, page E-19.
details	Adds information to the alerts.details table.	Details Function, page E-23.
discard	Deletes the entire event.	Deleting Elements or Events, page E-16.
exists	Tests for the existence of an element.	Existence Function, page E-15.
expand	Returns a string (which must be a literal string) with escape sequences expanded.	String Functions, page E-16.
extract	Returns the part of a string (which can be a field, element, or string expression) that matches the parenthesized section of the regular expression.	String Functions, page E-16.
getdate	Returns the current date as a UTC date.	Date and Time Functions, page E-19.
getenv	Returns the value of an environment variable.	Host and Process Utility Functions, page E-20.
getload	Measures the load on the Cisco Info Server.	Monitoring Cisco Info Mediator Loads, page E-26.
getpid	Returns the process ID of the running Cisco Info Mediator.	Host and Process Utility Functions, page E-20.
hostname	Returns the name of the host running the Cisco Info Mediator.	Host and Process Utility Functions, page E-20.
int	Converts a number into an integer.	Math Functions, page E-18.

Table E-3 Rules File Functions (continued)

Function Name	Description	See...
length	Calculates the length of an expression and returns the numeric value.	String Functions, page E-16.
lookup	Uses a lookup table to map additional information to an alert.	Lookup Table Operations, page E-20.
lower	Converts an expression to lowercase.	String Functions, page E-16.
ltrim	Removes white space from the left of an expression.	String Functions, page E-16.
match	Tests for an exact string match.	String Functions, page E-16.
nmatch	Tests for a string match at the beginning of the specified string.	String Functions, page E-16.
printable	Converts any non-printable characters in the given expression into a space character.	String Functions, page E-16.
real	Converts a number into a real number.	Math Functions, page E-18.
recover	Recovers a discarded event.	Deleting Elements or Events, page E-16.
regmatch	Performs full regular expression matching of the value in the regular expression in a string.	String Functions, page E-16.
remove	Removes an element from the event.	Deleting Elements or Events, page E-16.
rtrim	Removes white space from the right of an expression.	String Functions, page E-16.
scanformat	Converts an expression according to the available formats, similar to the scanf family of routines in C.	String Functions, page E-16.
set	Enables you to log messages.	Message Logging Functions, page E-23.
setlog	Enables you to set the message log level.	Message Logging Functions, page E-23.
setobjectserver, setdefaultobjectserver	Sets the Cisco Info Server to which alerts are sent.	Connecting to Multiple Cisco Info Servers, page E-15.
service	Sets the status of a service.	Service Function, page E-24.
substr	Extracts a substring from an expression.	String Functions, page E-16.
timetodate	Converts a UTC time value into a text string.	Date and Time Functions, page E-19.
update	Indicates which fields are updated when an alert is deduplicated.	Update on Deduplication Function, page E-22.
updateload	Updates the load statistics for the Cisco Info Server.	Monitoring Cisco Info Mediator Loads, page E-26.
upper	Converts an expression to uppercase.	String Functions, page E-16.

Math and String Operators

You can use math operators to add, subtract, divide, and multiply numeric operands in expressions. [Table E-4](#) describes the math operators supported in rules files.

Table E-4 Math Operators

Operator	Description	Example
* /	Operators used to multiply (*) or divide (/) two operands.	<code>\$eventid=int(\$<eventid>)*2</code>
+ -	Operators used to add (+) or subtract (-) two operands.	<code>\$eventid=int(\$<eventid>)+1</code>

You can use string operators to manipulate character strings. [Table E-5](#) describes the string operator supported in rules files.

Table E-5 String Operator

Operator	Description	Example
+	Concatenates two or more strings.	<code>@field = \$<element1> + "<message>" + \$<element2></code>

Bit Manipulation Operators

You can use bitwise operators to manipulate integer operands in expressions. [Table E-6](#) describes the bitwise operators supported in rules files.

Table E-6 Bitwise Operators

Operator	Description	Example
& ^	Bitwise AND (&), OR (), and XOR (^). The results are determined bit-by-bit.	<code>\$result1 = int(\$<number1>) & int(\$<number2>)</code>
>> <<	Shift bits right (>>) or left (<<).	<code>\$result2 = int(\$<number3>) >> 1</code>

These operators manipulate the bits in integer expressions. For example, in the statement:

```
$result2 = int($<number3>) >> 1
```

If `<number3>` has the value **17**, `<result2>` resolves to **8**, as shown:

```
16 8 4 2 1
>>1 0 0 0 1
0 1 0 0 0
```

Note the bits do not wrap around; when they are pushed off one end, they are replaced on the other end by a **0**.

Bitwise operators only work with integer expressions. Elements are stored as strings, so you need to use the `int` function, described in [Math Functions, page E-18](#), to convert them into integers before performing these operations.

Comparison Operators

You can use comparison operators to test numeric values for equality and inequality. [Table E-7](#) describes the comparison operators supported in rules files.

Table E-7 Comparison Operators

Operator	Description	Example
<code>==</code>	Tests for equality.	<code>int(\$<eventid>) == 5</code>
<code>!=</code>	Tests for inequality.	<code>int(\$<eventid>) != 0</code>
<code><></code>		
<code><</code> <code>></code> <code><=</code> <code>>=</code>	Tests for greater than (<code>></code>), less than (<code><</code>), greater than or equal to (<code>>=</code>), or less than or equal to (<code><=</code>).	<code>int(\$<eventid>) <=30</code>

Logical Operators

You can use logical operators on boolean values to form expressions that resolve to **TRUE** or **FALSE**. [Table E-8](#) describes the logical operators supported in rules files.

Table E-8 Logical Operators

Operator	Description	Example
NOT (also <code>!</code>)	A NOT expression negates the input value, and is TRUE only if its input is FALSE .	<code>if NOT(<Severity>=0)</code>
AND (also <code>&&</code>)	An AND expression is true only if all of its inputs are TRUE .	<code>if match(\$<Enterprise>,"Acme") && match(\$<trap-type>,"Link-Up")</code>
OR (also <code> </code>)	An OR expression is TRUE if any of its inputs are TRUE .	<code>if match(\$<Enterprise>,"Acme") match(\$Enterprise,"Bo")</code>
XOR (also <code>^</code>)	An XOR expression is TRUE if either of its inputs, but not both, are TRUE .	<code>if match(\$<Enterprise>,"Acme") XOR match(\$<Enterprise>,"Bo")</code>

Connecting to Multiple Cisco Info Servers

The **setobjectserver** function enables you to specify the Cisco Info Server to which an alert will be sent. The **setdefaultobjectserver** function enables you to specify the default Cisco Info Server to which alerts are sent when none is specified.

The syntax to set the Cisco Info Server to which alerts are sent is:

```
setobjectserver(<string_expression>)
```

For example:

```
setobjectserver(upper($ServerName))
```

The syntax to set the default Cisco Info Server is:

```
setdefaultobjectserver(<string_expression>)
```

For example:

```
setdefaultobjectserver("NCOMS2")
```

Any Cisco Info Servers specified in these functions must first be identified in the **Server** property or **-server** command line option, described in [Common Cisco Info Mediator Properties and Command Line Options, page E-1](#). An example properties file entry is:

Server: "TEST1:TEST2:TEST3"

The first Cisco Info Server specified, in this case **TEST1**, is the default Cisco Info Server. You can change both the default Cisco Info Server and the Cisco Info Server to which specific alerts are sent, as shown in the following example:

```
# Once an event of Major severity or higher comes in, set the default Info Mediator to
TEST2
if(int(@Severity) > 3)
{
    setdefaultobjectserver("TEST2")
}
# Send all clear events to TEST3
if (int(@Severity) = 0)
{
    setobjectserver("TEST3")
}
```

Existence Function

You can use the exists function to test for the existence of an element, with the following syntax:

```
exists ($<element>)
```

The function returns **TRUE** if the element was created for this particular event; otherwise it returns **FALSE**.

Deleting Elements or Events

You can use functions to remove elements from an event, discard an entire event, and recover a discarded event. [Table E-9](#) describes these functions.

Table E-9 Delete, Discard, and Recover Functions

Function	Description	Example
remove (<element_name>)	Removes the element from the event.	remove (<element_name>)
discard	Deletes an entire event. This must be in a conditional statement; otherwise, all events are discarded.	if match (@Node,"testnode") { discard }
recover	Recovers a discarded event.	if match (@Node,"testnode") { recover }

String Functions

You can use string functions to manipulate string elements, typically field or element names. [Table E-10](#) describes the string functions supported in rules files.

Table E-10 String Functions

Function	Description	Example
upper (<expression>)	Converts an expression to uppercase.	\$Node = upper (\$Node)
lower (<expression>)	Converts an expression to lowercase.	\$Node = lower (\$Node)
length (<expression>)	Calculates the length of an expression and returns the numeric value.	\$LengthNode = length (\$Node)
rtrim (<expression>)	Removes white space from the right of an expression.	\$TrimNode = rtrim (\$Node)
ltrim (<expression>)	Removes white space from the left of an expression.	\$TrimNode = ltrim (\$Node)

Table E-10 String Functions (continued)

Function	Description	Example
scanformat (<expression>, "<string>")	Converts the expression according to the following formats, similar to the scanf family of routines in C. Conversion specifications are: %% - literal %; do not interpret. %d - matches an optionally signed decimal integer. %u - same as %d; no check is made for sign. %o - matches an optionally signed octal number. %x - matches an optionally signed hexadecimal number. %i - matches an optionally signed integer. %e, %f, %g - matches an optionally signed floating point number. %s - matches a string terminated by white space or end of string.	\$element = "Foo is up in 15 seconds" [\$node, \$state, \$time] = scanformat(\$element, "%s is %s in %d seconds") This sets \$node , \$state , and \$time to Foo, up, and 15, respectively.
substr (<expression>, <n>, <len>)	Extracts a substring, starting at the position specified in the second parameter, for the number of characters specified by the third parameter.	\$Substring = substr(\$Node,2,10)
printable (<expression>)	Converts any non-printable characters in the given expression into a space character.	\$Print = printable(\$Node)
match (<expression>, "string")	TRUE if the expression value matches the string exactly.	if match(\$Node, "New")
nmatch (<expression>, "string")	TRUE if the expression starts with the specified string.	if nmatch(\$Node, "New")
regmatch (<string>, "regexp")	Full regular expression matching of the value in the regular expression in the string (which can be a field, element, or string expression). For more information on regular expressions, see Appendix D, "Regular Expressions."	if (regmatch(\$enterprise, "^Acme Config:[0-9]"))

Table E-10 String Functions (continued)

Function	Description	Example
extract(<string>,"regexp")	The extract function returns the part of the string (which can be a field, element, or string expression) that matches the parenthesized section of the regular expression. Regular expression pattern matching is described in Appendix D, "Regular Expressions."	extract (\$expr,"ab([0-9]+)cd") If \$expr is "ab123cd" then the value returned is 123.
expand("string")	The expand function returns the string (which must be a literal string) with escape sequences expanded. Possible expansions are: \" - double quote \NNN - octal value of NNN \ - backslash \a - alert (BEL) \b - backspace e - escape (033 octal) f - form feed n - new line r - carriage return \t - horizontal tab v - vertical tab This function cannot be used as the regular expression argument in the regmatch or extract functions.	log(debug, expand("Rules file with embedded \\")) sends the following to the log: Sun Oct 21 19:56:15 2001 Debug: Rules file with embedded \"

Math Functions

You can use math functions to perform numeric operations on elements. Elements are stored as strings, so you must use these functions to convert them into integers before performing these operations.

[Table E-11](#) describes the math functions supported in rules files.

Table E-11 Math Functions

Function	Description	Example
int(<numeric>)	Converts a number into an integer.	if int(\$PercentFull) > 80
real(<numeric>)	Converts a number into a real number.	@DiskSpace= (real(\$diskspace)/real(\$total))*100

In the following example, the severity of an alert that monitors disk space usage is set depending on the amount of available disk space.

```
if (int($PercentFull) > 80 && int($PercentFull) <=85)
{
    @Severity=2
}
else if (int($PercentFull) > 85 && int($PercentFull) <=90)
{
    @Severity=3
}
else if (int($PercentFull) > 90 && int($PercentFull) <=95)
{
    @Severity=4
}
else if (int($PercentFull) > 95)
{
    @Severity=5
}
```

The percentage of disk space is not always provided in the event stream. The percentage of disk space can be calculated in the rules file as follows:

```
if (int($total) > 0)
{
    @DiskSpace=(100*int($diskspace)/int($total))
}
```

This can also be calculated using the real function:

```
if (int($total) > 0)
{
    @DiskSpace=(real($diskspace)/real($total))*100
}
```

The previous example can then be used to set the severity of the alert.

Date and Time Functions

You can use date and time functions to obtain the current time or to perform date and time conversions. Times are specified in Coordinated Universal Time (UTC)—the number of elapsed seconds since 1 January 1970. [Table E-12](#) describes the date and time functions supported in rules files.

Table E-12 Date and Time Functions

Function	Description	Example
<code>getdate()</code>	Takes no arguments and returns the current date as a UTC date.	<code>\$tempdate = getdate()</code>

Table E-12 Date and Time Functions (continued)

Function	Description	Example
datetotime(<string>, conversion_specification)	Converts a text string into UTC time using the C library function strptime() . See the man page for strptime for more information.	\$Date = datetotime("Tue Dec 19 18:33:11 GMT 2000", "%a %b %e %T %Z %Y")
timetodate(UTC, conversion_specification)	Converts a UTC time value into a text string using the C library function strftime() . See the man page for strftime for more information.	@StateChange = timetodate (\$StateChange, "%T, %D")

Host and Process Utility Functions

You can use utility functions to obtain information about the environment in which the Cisco Info Mediator is running. Table E-13 describes the host and process functions supported in rules files.

Table E-13 Host and Process Utility Functions

Function	Description	Example
getenv(<string>)	Returns the value of a specified environment variable.	\$My_OMNIHOME = getenv("OMNIHOME")
getpid()	Returns the process ID of the running Cisco Info Mediator.	\$My_PID = getpid()
hostname()	Returns the name of the host running the Cisco Info Mediator.	\$My_Hostname = hostname()

Lookup Table Operations

Lookup tables provide a way to add extra information in an event. A table is made up of a list of keys and values. It is defined by using the **table** keyword and accessed using the **lookup** keyword. You can create a lookup table directly in the rules file or in a separate file.

The **lookup** keyword evaluates the expression in the keys of the named table and returns the associated value. If the key is not found, an empty string is returned. The **lookup** function has the following syntax:

```
lookup(<expression>, <tablename>)
```

Table definitions must appear at the start of a rules file, before any processing statements. You can define multiple tables in a rules file.

You must restart the Cisco Info Mediator before any changes made to a lookup table take effect.

Defining Lookup Tables in the Rules File

You can create a lookup table directly in the rules file using the following format:

```
table tablename={{ "<key>", "<value>" }, { "<key>", "<value>" } ... }
```

For example, to create a table that matches a node name to the department the node is in:

```
table dept={{ "node1", "Technical" }, { "node2", "Finance" }}
```

You can access this lookup table in the rules file as follows:

```
@ExtraChar=lookup(@Node,dept)
```

This example uses the **@Node** field as the key. If the value of the **@Node** field matches a key in the table, **@ExtraChar** is set to the corresponding value.

A lookup table can also have multiple columns. For example:

```
table example_table =
{ "key1", "value1", "value2", "value3" },
{ "key2", "val1", "val2", "val3" }
```

You can obtain values from a multiple value lookup table as follows:

```
[@Summary, @AlertKey, $error_code] = lookup("key1", "example_table")
```

Defining Lookup Tables in a Separate File

Rather than including the lookup table directly in the rules file, you can create the table in a separate file. If you are specifying a single value, the file must be in the format:

```
key[TAB]value
key[TAB]value
```

To create a table in which the node name is matched to the department the node is in, use the following format:

```
node1[TAB]Technical
node2[TAB]Finance
```

Define the table file, preceding any processing statements, by specifying the full path to the file as follows:

```
table dept="/opt/Omnibus/probes/solaris2/Dept"
```

You can then use this lookup table in the rules file as follows:

```
@ExtraChar=lookup(@Node,dept)
```

For multiple values, the format is:

```
key1[TAB]<value1>[TAB]<value2>[TAB]<value3>
```

```
key2[TAB]<val2>[TAB]<val3>[TAB]<val3>
```

The number of values in each row of the lookup table must match the number of values obtained from the **lookup** command. If they do not match, an error is generated.

The following example is incorrect, because the first entry has three values while the second only has two values:

```
table example_table =
{ "key1", "value1", "value2", "value3" },
{ "key2", "val1", "val2" }
```

The following example is also incorrect, because the lookup table holds three values, but only two fields are being set by the **lookup** command:

```
table example_table =
{"key1", "value1", "value2", "value3"},
{"key2", "val1", "val2", "val3"}}
[@Summary, @AlertKey] = lookup("key1", "example_table")
```

Specifying Default Table Values

You can specify a default option to handle an event that does not match any of the key values in a table. The default statement must follow the specific table definition.

The following is an example for a table in the rules file:

```
table example_table =
{"key1", "value1", "value2", "value3"},
{"key2", "val1", "val2", "val3"}}
default = {"defval1", "defval2", "defval3"}
```

The following is an example for a table in a separate file:

```
table dept="/opt/Omnibus/probes/solaris2/Dept"
default="UNKNOWN"
```

Update on Deduplication Function

The deduplication process is managed by the Cisco Info Server, but it can be configured in the Cisco Info Mediator rules file. In each rules file you can specify which fields of an alert are to be updated if the alert is deduplicated using the update function. This allows deduplication rules to be set on a per-alert basis.

The update function has the following syntax options:

update (<fieldname>)

or

update (<fieldname>, [TRUE | FALSE])

If set to **TRUE**, update on deduplication is enabled. If set to **FALSE**, update on deduplication is disabled. The default is **TRUE**.

For example, to ensure the **Severity** field is updated on deduplication, you can add the following to the rules file:

```
update (@Severity)
```

You can also disable update on deduplication for a specific field. For example:

```
update (@Tally, FALSE)
```



Note

Update on deduplication can be set in either the SQL or the rules file, but not in both. If you set update on deduplication in the SQL file, it overrides the rules file setting.

Details Function

Details are extra elements created by a Cisco Info Mediator to display alert information that is not stored in a field of the **alerts.status** table. Alerts do not have detail information unless it is added.

Detail elements are stored in the Cisco Info Server details table, **alerts.details**. You can view details by double-clicking an alert in the Event List.

You can add information to the details table using the **details** function. The detail information is added when an alert is inserted, but not if it is deduplicated.

The following example adds the elements **\$a** and **\$b** to the **alerts.details** table:

```
details($a,$b)
```

The following example adds all of the alert information to the **alerts.details** table:

```
details ($*)
```



Note

You should only use (\$*) when you are debugging or writing rules files. If you use (\$*), events forwarded to the Cisco Info Server in this way are not processed in the normal way. If you use (\$*) for long periods of time, the Cisco Info Server tables will become too large and the performance of the Cisco Info Server will suffer.

In the following example, **\$Summary** is compared to the strings **Incoming** and **Backup**. If there is no match, **\$Summary** is set to the string, and all of the information for the event is added to the details table:

```
if (match($Summary, "Incoming"))
{
    @Summary = "Received a call"
}
else if(match($Summary, "Backup"))
{
    @Summary = "Attempting to back up"
}
else
{
    @Summary = "Please see details"
}
details($*)
}
```

Message Logging Functions

You can use the log function to log messages during rules processing. You can also set a log level using the **setlog** function, and only messages equal to or above that level are logged. There are five log levels: **DEBUG**, **INFO**, **WARNING**, **ERROR**, and **FATAL**, in order of increasing severity. For example, if the log level is set to **WARNING**, only **WARNING**, **ERROR**, and **FATAL** messages are logged, however, if the logging is set to **ERROR**, only **ERROR** and **FATAL** messages are logged.

Log Function

The **log** function sends a message to the log file.

The syntax of the **log** function is:

```
log([DEBUG | INFO | WARNING | ERROR | FATAL], "<string>")
```

**Note**

When a **FATAL** message is logged, the Cisco Info Mediator terminates.

Setlog Function

The **setlog** function sets the minimum level at which messages are logged during rules processing. By default, the level for logging is **WARNING** and above.

The syntax of this function is:

```
setlog([DEBUG | INFO | WARNING | ERROR | FATAL])
```

Message Logging Example

The following is a sequence of log functions executed in the rules file:

```
setlog(WARNING)
log(DEBUG, "A debug message")
log(WARNING, "A warning message")
setlog(ERROR)
log(WARNING, "Another warning message")
log(ERROR, "An error message")
```

This produces log output of:

A warning message.

An error message.

The **DEBUG** level message is not logged, because the logging setting is set higher than **DEBUG**. The second **WARNING** level message is not logged, because the preceding **setlog** function has set the log level higher than **WARNING**.

Service Function

Use the **service** function to define the status of a service before alerts are forwarded to the Cisco Info Server. The status changes the color of the alert when it is displayed in the Event List and Service windows.

The syntax of the **service** function is:

```
service(<service_identifier>, <service_status>)
```

The <service_identifier> identifies the monitored service, for example, **\$host**.

[Table E-14](#) lists the service status levels.

Table E-14 Service Function Status Levels

Service Status Level	Description
BAD	The service level agreement is not being met.
MARGINAL	There are some problems with the service.
GOOD	There are no problems with the service.
No Level Defined	The status of the service is unknown.

Service Function Example

If you want a Ping Info Mediator to return a service status for each host it monitors, you can use the service function in the rules file to assign a service status to each alert. The following example shows an extract from a rules file that assigns a service status to each alert based on the value of the **status** element.

```
switch ($status)
{
case "unreachable":
    @Severity = "5"
    @Summary = @Node + " is not reachable"
    @Type = 1
    service($host, bad)    # Service Entry
case "alive":
    @Severity = "3"
    @Summary = @Node + " is now alive"
    @Type = 2
    service($host, good)   # Service Entry
case "noaddress":
    @Severity = "2"
    @Summary = @Node + " has no address"
    service($host, marginal) # Service Entry
case "removed":
    @Severity = "5"
    @Summary = @Node + " has been removed"
    service($host, marginal) # Service Entry
case "slow":
    @Severity = "2"
    @Summary = @Node + " has not responded within trip time"
    service($host, marginal) # Service Entry
case "newhost":
    @Severity = "1"
    @Summary = @Node + " is a new host"
    service($host, good)    # Service Entry
case "responded":
    @Severity = "0"
    @Summary = @Node + " has responded"
    service($host, good)    # Service Entry
default:
    @Summary = "Ping Info Mediator Error details: " + $*
    @Severity = "3"
    service($host, marginal) # Service Entry
}
```

Monitoring Cisco Info Mediator Loads

To monitor load, it is necessary to obtain time measurements and calculate the number of events processed over time. The **updateload** function takes a time measurement each time it is called and the **getload** function returns the load as events per second.

Each time the **updateload** function is executed, the current time stamp, recorded in seconds and microseconds, is added to the beginning of a series of time stamps. The remaining time stamps record the difference in time from the previous time stamp. For example, to take a time measurement and update a property called **load** with a new time stamp:

```
%load = updateload(%load)
```



Note

Depending on the operating system, differing levels of granularity may be reported in time stamps.

You can specify a maximum time window for which samples are kept, and a maximum number of samples. By default, the time window is *one second* and the maximum number of samples is *fifty*. You can specify the number of seconds for which load samples are kept and the maximum number of samples in the format:

```
<time window in seconds>.<max number of samples>
```

For example, to set or reset these values for the **load** property:

```
%load = "2.40"
```

When the number of seconds in the time window is exceeded, any samples outside of that time window are removed. When the number of samples reaches the limit, the oldest measurement is removed.

The **getload** function calculates the current load, returned as events per second. For example, to calculate the current load and assign it to a temporary element called **current_load**:

```
$current_load = getload(%load)
```

In the following example, the **load** property is initialized the first time the rules file is executed with a sample time window of three seconds and a maximum of ten samples. For each following execution, the current load is logged to the Cisco Info Mediator log file.

```
if (match(%load, "")) {
    %load="3.10"
}
%load = updateload(%load)
$current_load = getload(%load)
log(DEBUG, "Events per second is " + $current_load)
```

Testing Rules Files

You can test the syntax of a rules file using the Syntax Info Mediator: **nco_p_syntax**. This is more efficient than actually running the Cisco Info Mediator to test whether the syntax of the rules file is correct.

To run the Syntax Info Mediator, enter:

```
nco_p_syntax -rulesfile /test/rulesfile
```

In this command, **/test/rulesfile** is the rules file. The Syntax Info Mediator always runs in debug mode. It connects to the Cisco Info Server, tests the rules file, displays any errors to the screen, and then exits. If no errors are displayed, the syntax of the rules file is correct.

Debugging Rules Files

When making changes to the rules file, adding new rules, or creating lookup tables, it is useful to test the Cisco Info Mediator by executing it in debug mode. This shows exactly how an event is being parsed by the Cisco Info Mediator and any possible problems with the rules file.

You can set the Cisco Info Mediator to run in debug mode on the command line or in the properties file. To enable debug mode on the command line, enter the command:

```
$OMNIHOME/probes/arch/<Info Mediator name> -messagelevel DEBUG -messagelog STDOUT
```

Alternatively, you can enter the following in the properties file:

```
MessageLevel: "DEBUG"
```

```
MessageLog: "STDOUT"
```

If you omit the **MessageLog** property or command line option, the debug information is sent to the Cisco Info Mediator log file in the **\$OMNIHOME/log** directory rather than to the screen.

Rules File Examples

The following sections show examples of typical rules file segments.

Enhancing the Summary Field

This example rule tests if the **\$strap-type** element is **"Link-Up"**. If it is, the **@Summary** field is populated with a string made up of **"Link up on "**, the name of the node from the record being generated, **" Port "**, and the value of the **\$ifIndex** element:

```
if (match($strap-type, "Link-Up"))
{
    @Summary = "Link up on " + @Node + " Port " + $ifIndex
}
```

Populating Multiple Fields

This example rule is similar to the previous rule except the **@AlertKey** and **@Severity** fields are also populated:

```
if (match($trap-type, "Link-Up"))
{
    @Summary = "Link up on " + @Node + " Port " + $ifIndex
    @AlertKey = $ifIndex
    @Severity = 4
}
```

Nested IFs

This example rule first tests if the trap has come from an Acme manager and then tests if it is a **"Link-Up"**. If both conditions are true, the **@Summary** field is populated with extra information:

```
if (match($enterprise, "Acme"))
{
    if (match($trap-type, "Link-Up"))
    {
        @Summary= "Acme Link Up on " + @Node + " Port " + $ifIndex + " Reason: " + $ifLocReason
    }
}
```

Regular Expression Match

This example rule tests for a line starting with **"Acme Configuration:"** followed by a single digit:

```
if (regmatch($enterprise, "^Acme Configuration:[0-9]"))
{
    @Summary="Generic configuration change for "+@Node
}
```

Regular Expression Extract

This example rule tests for a line starting with **"Acme Configuration:"** followed by a single digit. Then it extracts that single digit and places it in the **@Summary** field:

```
if (regmatch($enterprise, "^Acme Configuration:[0-9]"))
{
    @Summary="Acme error "+extract($enterprise, "^Acme Configuration:
([0-9])")+ "on "+@Node
}
```

Numeric Comparisons

This example rule tests the value of an element called **\$freespace** as a numeric value by converting it to an integer and performing a numeric comparison:

```
if (int($freespace) < 1024)
{
    @Summary="Less than 1024K free on drive array"
}
```

Simple Numeric Expressions

This example rule creates an element called **\$tmpval**. The value of **\$tmpval** is derived from the **\$temperature** element which is converted to an integer and then has **20** subtracted from it. The string element **\$tmpval** contains the result of this calculation:

```
$tmpval=int($temperature)-20
```

Strings and Numerics in One Expression

This example rule creates an element called **\$Kilobytes**. The value of **\$Kilobytes** is derived from the **\$DiskSize** element, which divided by **1024** before being cast back into a string type with the letter **K** appended:

```
$Kilobytes = string(int($DiskSize)/1024) + "K"
```

Using Load Functions to Monitor Nodes

This example shows how to measure load for each node that is generating events. If a node is producing more than five events per second, a warning is written to the Cisco Info Mediator log file. If more than 80 events per second are generated for all nodes being monitored by the Cisco Info Mediator, events are sent to an alternate Cisco Info Server and a warning is written to the Cisco Info Mediator log file.

```
# set the following in the probes properties file: Server: "HIGHLOAD:LOWLOAD"
# declare an array, loads, at the top of the rules file array loads;

# initialize array items with the number of seconds samples may span and
# number of samples to maintain.

if (match("", loads[@Node])){
    loads[@Node] = "2.50"
}
if (match("", %general_load)){
    %general_load="2.50"
}
loads[@Node] = updateload(loads[@Node])
%general_load=updateload(%general_load)
if (int(getload(loads[@Node])) > 5){
    log(WARN, $Node + " is creating more than 5 events per second")
}
if (int(getload(%general_load)) > 80){
    log(WARN, "Info Mediator is creating more than 80 events per second - switching to
HIGHLOAD")
    setobjectserver("HIGHLOAD")
}
```

Troubleshooting Cisco Info Mediators

This section describes some of the common problems experienced by Cisco Info Center users and explains possible causes and solutions.

This troubleshooting information is divided into two sections:

- [Common Problem Causes](#)
- [What To Do If](#), page E-31.

Table E-15 Host and Process Utility Functions

Section	Description
Common Problem Causes	This section contains a list of common problem causes. If you are unsure what your problem is, you should start by reading this section and following the instructions. If you cannot solve your problem by following the instructions in this part, move on to What To Do If , page E-31.
What To Do If	This section describes common symptoms caused by Cisco Info Mediator problems and step-by-step instructions to help you locate and solve the problem. If none of the headings in this section match the symptoms of your problem, read through the lists of instructions and make sure you have tried all of the most likely solutions listed there.

If you have tried all of the suggested problem solutions and your Cisco Info Mediator still does not work, see your support contract and contact the help desk.

Common Problem Causes

The most common causes of Cisco Info Mediator problems are:

- incorrectly set **OMNIHOME** and **NETCOOL_LICENSE_FILE** variables
- errors in the rules file, particularly in extract statements
- configuration errors in the properties file.

For information about setting the **OMNIHOME** and **NETCOOL_LICENSE_FILE** variables, see the *Cisco Info Center Installation and Configuration Guide*, 3.6.

For information about solving rules file problems, see [Testing Rules Files](#), page E-27 and [Debugging Rules Files](#), page E-27.

For information about Cisco Info Mediator properties, see [Common Cisco Info Mediator Properties and Command Line Options](#), page E-1. Ensure all of the properties are set correctly in the Cisco Info Mediator properties file. For example, ensure the **Server** property contains the correct Cisco Info Server name and the **RulesFile** property contains the correct rules file name.

If you cannot solve the problem, read through the next section and make sure you have tried all of the most likely solutions listed there.

What To Do If

The headings in this section describe the most common symptoms of Cisco Info Mediator problems. Find the heading that most closely describes your problem, then follow the instructions until you locate the cause and solve the problem.

Table E-16 Cisco Info Mediator Problems

Problem	See...
The Cisco Info Mediator will not start.	The Cisco Info Mediator Will Not Start.
The Cisco Info Mediator is not sending alerts to the Cisco Info Server.	The Cisco Info Mediator Is Not Sending Alerts to the Cisco Info Server, page E-33.
The Cisco Info Mediator is losing events.	The Cisco Info Mediator Is Losing Events, page E-34.
The Cisco Info Mediator is consuming too much CPU time.	The Cisco Info Mediator Is Consuming Too Much CPU Time, page E-35.
The Event List fields are not being populated properly.	The Event List Fields Are Not Being Populated Properly, page E-35.

If none of the headings match the symptoms of your problem, read through the lists of instructions and make sure you have tried all of the most likely solutions listed there. If you have tried all of the suggested problem solutions and your Cisco Info Mediator still does not work, see your support contract and contact the help desk.

The Cisco Info Mediator Will Not Start

If the Cisco Info Mediator does not start:

-
- Step 1** Run the Cisco Info Mediator in debug mode by entering:
- ```
J<Info Mediator name> -server <servername> -messagelevel DEBUG &
```
- where *<Info Mediator name>* is the name of the Cisco Info Mediator that is not working and *<servername>* is the name of the server the Cisco Info Mediator is attempting to send alerts. Any errors are recorded in the `$OMNIHOME/log/<Info Mediatorname>.log` debug log file. Check the errors recorded in the debug log file to locate the problem.
- For more information about debugging rules files, see [Debugging Rules Files, page E-27](#).
- Step 2** Ensure the Cisco Info Server is running by trying to connect using `nco_sql`.
- If you can log in successfully, the Cisco Info Server is running. If the Cisco Info Server is not running, this is likely to be the cause of the problem. For more information about using the Cisco Info Server and `nco_sql`, see [Direct Access Using the SQL Interactive Interface \(nco\\_sql\), page 2-4](#).
- Step 3** Ensure the Cisco Info Server has been configured correctly in the Server Editor (`nco_xigen`) and the interfaces information has been distributed to the Cisco Info Server and Cisco Info Mediator hosts, as described in the *Cisco Info Center Installation and Configuration Guide, 3.6*.

**Step 4** Ensure there are no other Cisco Info Mediators running with the same configuration using the command:  
**`/usr/bin/ps -ef | grep nco`**

A list of Cisco Info Center processes is displayed. Ensure none of the processes correspond to the same type of Cisco Info Mediator. Cisco Info Center does not allow you to run two identical Cisco Info Mediator configurations because this would duplicate all of the events forwarded to the Cisco Info Server.

**Step 5** Ensure you have enough licenses available to start another Cisco Info Mediator by entering:  
**`$NCLICENSE/bin/nc_print_license`**

If you do not have enough licenses to run another Cisco Info Mediator and you cannot stop any of the other Cisco Info Mediators, contact the help desk to request another license.

**Step 6** Ensure you are using the correct Cisco Info Mediator for the current version of the target software.

**Step 7** Ensure no syntax errors exist in the rules file.

See [Testing Rules Files, page E-27](#) and [Debugging Rules Files, page E-27](#) for more information about how to do this.

**Step 8** Ensure your system has not run out of system resources and can launch more processes. You can do this using **`df -k`** or **`top`**.

See the **`df`** and **`top man`** pages for more information about using these commands.

**Step 9** Check if the **`$OMNIHOME/var/<Info Mediatorname>.saf`** store and forward file exists. If it does, ensure it has not become too large.

If your disk is full, the Cisco Info Mediators and Cisco Info Servers will not be able to work properly.



**Note** Store and forward is not designed to handle very large numbers of events. Left unattended, a store and forward file will continue to grow until it runs out of disk space. See [Common Cisco Info Mediator Properties and Command Line Options, page E-1](#) for information about setting the **`MaxSAFFileSize`** property.

**Step 10** Ensure the store and forward file has not been corrupted.

If the store and forward file has been corrupted, there should be an error message in the **`$OMNIHOME/log/<Info Mediatorname>.log`** log file. If the file is corrupted, delete it and restart the Cisco Info Mediator.

**Step 11** Ensure the Cisco Info Mediator binary you are trying to run is the correct one for the current architecture by entering:

**`$OMNIHOME/bin/arch/<Info Mediatorname> -version`**

**Step 12** Ensure the Cisco Info Mediator version matches your system architecture.

## If You Are Running the Cisco Info Mediator on a Remote Host

**Step 1** Ensure the **`OMNIHOME`** and **`NETCOOL_LICENSE_FILE`** variables are set correctly.

See the *Cisco Info Center Installation and Configuration Guide, 3.6* for more information about how to do this.

- Step 2** Ensure the Cisco Info Mediator host can connect to the Cisco Info Server host using the **ping** command. Try to **ping** the Cisco Info Server host machine using the host name and the IP address. See the **ping man** page for more information about how to do this.
- If you cannot connect to the Cisco Info Server host using the **ping** command, there is a problem with the connection between your Cisco Info Mediator host and your Cisco Info Server host.
- Step 3** Ensure the Cisco Info Server has been configured correctly in the Server Editor (**nco\_xigen**) and the interfaces information has been distributed to the Cisco Info Server and Cisco Info Mediator hosts. See the *Cisco Info Center Installation and Configuration Guide, 3.6* for more information.
- Step 4** Check if a firewall exists between the Cisco Info Mediator host and the Cisco Info Server host.
- Step 5** If there is, ensure the firewall allows traffic between the Cisco Info Mediator and the Cisco Info Server.
- 

## The Cisco Info Mediator Is Not Sending Alerts to the Cisco Info Server

If the Cisco Info Mediator is not sending alerts to the Cisco Info Server:

---

- Step 1** Ensure the Cisco Info Mediator is running by entering:
- ```
ps -ef | grep nco
```
- A list of Cisco Info Center processes is displayed. If the Cisco Info Mediator is not running, start the Cisco Info Mediator from the command line.
- Step 2** Ensure no other Cisco Info Mediators are running with the same configuration by entering:
- ```
ps -ef | grep nco
```
- A list of Cisco Info Center processes is displayed. Ensure none of the processes correspond to the same type of Cisco Info Mediator. Cisco Info Center does not allow you to run two identical Cisco Info Mediator configurations because this would duplicate all of the events forwarded to the Cisco Info Server.
- Step 3** Read the Cisco Info Mediator properties file and ensure all properties are set correctly. For example, ensure the **Server** property contains the correct Cisco Info Server name and the **RulesFile** property contains the correct rules file name.
- Step 4** Check whether the Cisco Info Mediator event source has events to send to the Cisco Info Server.
- Step 5** Ensure the Cisco Info Server you are logged in to is the same Cisco Info Server the Cisco Info Mediator is forwarding events.
- Step 6** Ensure the Cisco Info Mediator host can connect to the Cisco Info Server host using the **ping** command. Try to **ping** the Cisco Info Server host machine using the host name and the IP address. See the **ping man** page for more information about how to do this.
- If you cannot connect to the Cisco Info Server host using the **ping** command, there is a problem with the connection between your Cisco Info Mediator host and your Cisco Info Server host.
- Step 7** Ensure the event source to the Cisco Info Mediator is working correctly. See the documentation supplied with your Element Manager for more information about how to do this.
- Step 8** Ensure you are using the correct Cisco Info Mediator.

- Step 9** Ensure the Cisco Info Mediator is not running in store and forward mode.
- To do this, check the `$OMNIHOME/var/<Info Mediatorname>.saf` and `$OMNIHOME/var/<Info Mediatorname>.reco` files to see if they are growing. If they are, disable store and forward in the Cisco Info Mediator's properties file. See [Properties File, page 5-6](#) for more information.
- Step 10** Ensure your system has not run out of system resources and can launch more processes.
- You can do this using `df -k` or `top` commands. See the `df` and `top man` pages for more information about using these commands.
- 

### If You Are Running the Cisco Info Mediator on a Remote Host

- Step 1** Ensure the Cisco Info Mediator host can connect to the Cisco Info Server host using the `ping` command.
- Try to `ping` the Cisco Info Server host machine using the host name and the IP address. See the `ping man` page for more information about how to do this.
- If you cannot connect to the Cisco Info Server host using the `ping` command, there is a problem with the connection between your Cisco Info Mediator host and your Cisco Info Server host.
- Step 2** Ensure the Cisco Info Server has been configured correctly through the Server Editor (`nco_xigen`) and the interfaces information has been distributed to the Cisco Info Server and Cisco Info Mediator hosts.
- See the *Cisco Info Center Installation and Configuration Guide, 3.6* for more information.
- Step 3** See if a firewall exists between the Cisco Info Mediator host and the Cisco Info Server host.
- Step 4** If there is, ensure the firewall allows traffic between the Cisco Info Mediator and the Cisco Info Server.
- 

### The Cisco Info Mediator Is Losing Events

If not all events are being forwarded to the Cisco Info Server:

- Step 1** Run the Cisco Info Mediator in debug mode by entering:
- ```
J<Info Mediatorname> -server <servername> -messagelevel DEBUG &
```
- In this command, `<Info Mediatorname>` is the name of the Cisco Info Mediator that is not working and `<servername>` is the name of the Cisco Info Server the Cisco Info Mediator is attempting to send alerts. Any errors are recorded in the debug rules file. Check the errors recorded in the debug log file to locate the problem.
- For more information about debugging rules files, see [Debugging Rules Files, page E-27](#).
- Step 2** Ensure the event source to Cisco Info Mediator is working correctly.
- See the documentation supplied with your Element Manager for more information about how to do this.
- Step 3** Ensure the Cisco Info Mediator event source has events to send to the Cisco Info Server.
- Step 4** Ensure all of the properties in the properties file are set correctly.
- For example, ensure the `Server` property contains the correct Cisco Info Server name and the `RulesFile` property contains the correct rules file name.
-

The Cisco Info Mediator Is Consuming Too Much CPU Time

If the Cisco Info Mediator is consuming too much CPU time:

Step 1 Run the Cisco Info Mediator in debug mode by entering:

```
J<Info Mediatorname> -server <servername> -messagelevel DEBUG &
```

where *<Info Mediatorname>* is the name of the Cisco Info Mediator that is not working and *<servername>* is the name of the Cisco Info Server the Cisco Info Mediator is attempting to send alerts to. Any errors are recorded in the debug log file. Check the errors recorded in the debug log file to locate the problem.

For more information about debugging rules files, see [Debugging Rules Files, page E-27](#).

Step 2 Ensure the Cisco Info Mediator can connect to the event source.

Step 3 See if the `$OMNIHOME/var/<Info Mediatorname>.saf` store and forward file exists.

Step 4 If it does, ensure it has not become too large.

If your disk is full, the Cisco Info Mediators and Cisco Info Servers will not be able to work properly.



Note

Store and forward is not designed to handle very large numbers of alerts. Left unattended, a store and forward file continues to grow until it runs out of disk space. See [Common Cisco Info Mediator Properties and Command Line Options, page E-1](#) for information about setting the `MaxSAFFileSize` property.

Step 5 Ensure the store and forward file has not been corrupted.

If it has been corrupted, an error message should exist in the `$OMNIHOME/log/<Info Mediatorname>.log` log file.

Step 6 If the file is corrupted, delete it, then restart the Cisco Info Mediator.

The Event List Fields Are Not Being Populated Properly

If the Cisco Info Mediator is detecting events and forwarding them to the Cisco Info Server but the Event List fields are not being populated correctly:

Step 1 Run the Cisco Info Mediator in debug mode by entering:

```
J<Info Mediatorname> -server <servername> -messagelevel DEBUG &
```

In this command, *<Info Mediatorname>* is the name of the Cisco Info Mediator that is not working and *<servername>* is the name of the Cisco Info Server the Cisco Info Mediator is attempting to send alerts. Any errors are recorded in the debug rules file. Check the errors recorded in the debug log file to locate the problem.

For more information about debugging rules files, see [Debugging Rules Files, page E-27](#).

Step 2 Ensure the fields not being populated properly are being correctly mapped to elements in the rules file.

See [Cisco Info Mediator Rules File Processing, page E-6](#) for more information about configuring rules files.

- Step 3** Ensure it is not a GUI problem by listing the table fields using Cisco Info Server SQL. See [Chapter 2, “Cisco Info Server SQL,”](#) for more information about using the SQL interactive interface to view the table information.

Cisco Info Mediator Information and Error Messages

This section lists all messages not Cisco Info Mediator-specific, including ProbeWatch and TSMWatch messages.

See the individual Cisco Info Mediator guides for information about Cisco Info Mediator-specific messages.

Generic Error Messages

Cisco Info Mediators can generate the following types of messages:

- Fatal
- Error
- Warning
- Information
- Debug.

Fatal Level Messages

The Cisco Info Mediator automatically terminates when a fatal message is issued. See [Table E-17](#) for a list and description of the fatal messages.

Table E-17 Fatal Level Cisco Info Mediator Messages

Message	Description	Action
No matching field found for <field name>. Field <field name> not found in the status table.	The rules file being used refers to a field in the format @ fieldname which does not exist in the status table.	Ensure the rules file and correct the problem.
Connection to Info Mediator marked DEAD - aborting...	The connection to the Cisco Info Server ceased (and store and forward is not enabled in the Cisco Info Mediator).	Ensure the Cisco Info Server is available.
Failed to set interfaces file location. Failed to access OMNIHOME directory: <directory name>.	The Cisco Info Mediator was unable to locate the interfaces file.	Ensure the OMNIHOME environment variable is set to the correct destination.

Table E-17 Fatal Level Cisco Info Mediator Messages (continued)

Message	Description	Action
Unknown data type returned from the Info Mediator.	The Cisco Info Server has returned unknown data.	See your support contract for information about contacting the help desk.
Failed to initialize. Failed to create property. Failed to set property. Failed to define argument. Failed to process arguments. Session create failed - aborting.	Internal errors.	Note the message and contact the help desk. See your support contract for information about contacting the help desk.
Failed to connect - aborting	The Cisco Info Server is not available.	Ensure the Cisco Info Server is running, the interfaces file on the system where the Cisco Info Mediator is installed has an entry for it, and there is no networking problem between the two systems.
Failed to read rules - aborting	A property or command line option is pointing to a non-existent rules file.	Check the command line options and properties file and set them to refer to the correct rules file.

Error Level Messages

The Cisco Info Mediator is likely to terminate when an error message, as described in [Table E-18](#), is issued.

Table E-18 Error Level Cisco Info Mediator Messages

Message	Description	Action
Option -propsfile used without argument	The -propsfile command line option must be followed by a parameter specifying where to find the properties file.	Check the command line you are using to start the Cisco Info Mediator and supply a correct parameter.
Unknown option: <option name>	An option was used on the command line to start the Cisco Info Mediator, however, is not supported by the Cisco Info Mediator.	Check the Cisco Info Mediator documentation and the contents of the command line.
Option <option name> used without argument	The option used expects a parameter which has not been supplied.	Check the Cisco Info Mediator documentation and the contents of the command line.

Table E-18 Error Level Cisco Info Mediator Messages (continued)

Message	Description	Action
Property <property name> for option <option name> does not exist Can't set generic property <property name> using the command line	An option in the Cisco Info Mediator is not mapped correctly to a property.	Check the properties file for the named property and see the Cisco Info Mediator documentation for supported properties.
No extraction data for <regexp> - missing ()'s? Regex doesn't match for <string>	A regular expression in the rules file being used in the extract() function may be missing parentheses. The string data used to extract may not match the regular expression. Extract() is unable to extract data.	Check the rules file and correct the problem.
Failed to open Properties file: <properties file name>	The Cisco Info Mediator is unable to open the properties file.	Check the properties file or command line to ensure the properties file is in the specified location.
Properties file: <error description> at line <line no>	There is an error in the format of the properties file.	Check the properties file at the specified line number and correct the problem.
Unknown property <property name> - ignored	A property specified in the properties file does not exist in the Cisco Info Mediator.	Check the properties file for the named property and see the Cisco Info Mediator documentation for supported properties.
Regular Expression Error: <regexp>	A regular expression is incorrectly formed in the rules file.	Check the rules file for the regular expression and correct the entry.
Failed to open Rules file: <rules file name> The rules file for the Info Mediator is not available or incorrectly specified.	The Cisco Info Mediator is unable to open the rules file.	Check the properties file or command line to ensure the rules file is in the specified location.
Rules file: <error description> at line <line no>	There is an error in the rules file format or syntax.	Check the rules file at the specified line number and correct the problem.
Failed to open file: <file name>	A file referred to in the rules file (for example, with the table function) does not exist.	Check the rules file and ensure the file is available.
Unexpected value during results processing. Results processing failed Unexpected return from results processing.	There is a problem with the Cisco Info Server.	See your support contract for information about contacting the help desk.

Table E-18 Error Level Cisco Info Mediator Messages (continued)

Message	Description	Action
OS Error: <error message> Server <server name>: <error message> Procedure <procedure name>: <error message>	There is an error in the Sybase connection. There should be a subsequent message from the Cisco Info Mediator which details the effect of this error.	See your support contract for information about contacting the help desk.
Unknown message level <message level string> - using WARNING level	The properties file or command line specified a message level which is not supported.	Check the properties file or command line and use a supported message level (debug, info, warning, error, fatal).
Could not set <fieldname> field	The Cisco Info Mediator was unable to set a field value. This may be because the Cisco Info Server tables have been modified so that default fields are no longer present.	Ensure the Cisco Info Server is available.
Could not send alert	The Cisco Info Mediator was unable to send an alert (usually an internal alert) to the Cisco Info Server.	Ensure the Cisco Info Server is available.
Error Setting SIGQUIT Handler Error Setting SIGINT Handler Error Setting SIGTERM Handler	The Cisco Info Mediator was unable to set up a signal handler for either a QUIT , INT , or TERM .	See your support contract for information about contacting the help desk.
PropGetValue failed	A required property has not been set.	Check the properties file.
CreateAndSet failed CreateAndSet failed for attr: <element name>	The Cisco Info Mediator is unable to create an element.	See your support contract for information about contacting the help desk.
SessionProcess failed	The Cisco Info Mediator was unable to process the alert against the rules file.	See your support contract for information about contacting the help desk.
Failed to open message log: <file name>	The Cisco Info Mediator is unable to open the specified logging file.	Check the command line or properties file and correct the problem.
SendAlert failed	The Cisco Info Mediator was unable to send an alert to the Cisco Info Server.	Ensure the Cisco Info Server is available.

Warning Level Messages

These messages, shown in [Table E-19](#), are issued as warnings, however, they should not cause the Cisco Info Mediator to terminate.

Table E-19 Warning Level Cisco Info Mediator Messages

Message	Description	Action
Unknown property '<property name>' - ignored	A property specified in the properties file does not exist in the Cisco Info Mediator.	Check the properties file for the named property and see the Cisco Info Mediator documentation for supported properties.
Regular Expression Error: <regexp>	A regular expression is incorrectly formed in the rules file.	Check the rules file for the regular expression and correct the entry.
Failed to open Rules file: <rules file name> The rules file for the Info Mediator is not available or incorrectly specified.	The Cisco Info Mediator is unable to open the rules file.	Check the properties file or command line to ensure the rules file is in the specified location.
Rules file: '<error description>' at line <line no>	There is an error in the rules file format or syntax.	Check the rules file at the specified line number and correct the problem.
Failed to open file: <file name>	A file referred to in the rules file (for example, with the table function) does not exist.	Check the rules file and ensure the file is available.
Unexpected value during results processing Results processing failed Unexpected return from results processing	There is a problem with the Cisco Info Server.	See your support contract for information about contacting the help desk.
OS Error: <error message> Server '<server name>': <error message> Procedure '<procedure name>': <error message>	There is an error in the Sybase connection. There should be a subsequent message from the Cisco Info Mediator which details the effect of this error.	See your support contract for information about contacting the help desk.
Unknown message level '<message level string>' - using WARNING level	The properties file or command line specified a message level which is not supported.	Check the properties file or command line and use a supported message level (debug, info, warning, error, fatal).

Table E-19 Warning Level Cisco Info Mediator Messages (continued)

Message	Description	Action
Could not set <fieldname> field	The Cisco Info Mediator was unable to set a field value. This may be because the Cisco Info Server tables have been modified so that default fields are no longer present.	Ensure the Cisco Info Server is available.
Could not send alert	The Cisco Info Mediator was unable to send an alert (usually an internal alert) to the Cisco Info Server.	Ensure the Cisco Info Server is available.
Error Setting SIGQUIT Handler Error Setting SIGINT Handler Error Setting SIGTERM Handler	The Cisco Info Mediator was unable to set up a signal handler for either a QUIT, INT, or TERM.	See your support contract for information about contacting the help desk.
PropGetValue failed	A required property has not been set.	Check the properties file.
CreateAndSet failed CreateAndSet failed for attr: <element name>	The Cisco Info Mediator is unable to create an element.	See your support contract for information about contacting the help desk.
SessionProcess failed	The Cisco Info Mediator was unable to process the alert against the rules file.	See your support contract for information about contacting the help desk.
Failed to open message log: <file name>	The Cisco Info Mediator is unable to open the specified logging file.	Check the command line or properties file and correct the problem.
SendAlert failed	The Cisco Info Mediator was unable to send an alert to the Cisco Info Server.	Ensure the Cisco Info Server is available.
New value for field <field name> truncated to <number> characters	A string being copied into an alert field has had to be truncated to fit the field.	Check the rules file.
Failed to set SYBASE in environment	The Cisco Info Mediator was unable to override the SYBASE environment variable.	N/A

Table E-19 Warning Level Cisco Info Mediator Messages (continued)

Message	Description	Action
Type mismatch for property '<property name>' - new value ignored	A property has been set with the wrong data type.	Check the properties file or command line to ensure that the property is correctly set.
Results processing failed Failed to retrieve connection status - attempting to continue Failed to install Client Message Callback Failed to install Server Message Callback	There is a problem with the Cisco Info Server. The Cisco Info Mediator will try to continue.	N/A

Information Level Message

This message is for information purposes.

Table E-20 Information Level Cisco Info Mediator Message

Message	Description
Using stderr for logging.	The Cisco Info Mediator was unable to open a log file, so it is writing messages to stderr .

Debug

The debug messages, listed and described in [Table E-21](#), give detailed error and information messages on the internal functioning of the Cisco Info Mediator. These messages are aimed at Cisco Info Mediator developers, however, they are listed here for completeness.

Table E-21 Debug Level Cisco Info Mediator Messages

Message	Description
New value for field <field name> is <value>	A field value has been set.
OpInitialise() called more than once	Multiple calls have been made to the C Cisco Info Mediator API function OpInitialise() , which can only be called once.
A value for <string> doesn't exist in lookup table <table name>	A value requested from a lookup table is not available. The function in the rules file will return an empty string.

Table E-21 Debug Level Cisco Info Mediator Messages (continued)

Message	Description
Failed to allocate memory (Requested size was <number> bytes) Attempted to free NULL pointer Attempted to realloc NULL pointer Failed to reallocate memory block at address <hex address> (Requested size was <number> bytes) Attempted to duplicate NULL string Failed to duplicate string	An error or problem has occurred in the memory allocation or string handling components of the Cisco Info Mediator library. The library will cope with the problem.
Failed to allocate command structure Failed to set command query Failed to send command Failed to allocate context structure Failed to initialize Sybase internals: <number> Failed to set username Failed to set password Failed to set appname Failed to set hostname Failed to connect Failed to fetch number of columns No columns in result set Failed to describe column Failed to bind column Got a row fail - continuing	A problem or error has occurred at the Sybase connection or Cisco Info Server connection level.
Failed to flush alerts before EXIT	Problem during session destruction before EXIT .
Problem during disconnect before EXIT	Problem during shutdown before EXIT .

ProbeWatch and TSMWatch Messages

In certain situations, a Cisco Info Mediator or TSM generates events of their own. These events can provide information (such as startup or shutdown messages) or identify problems. This section describes the elements common to all ProbeWatch and TSMWatch messages.

ProbeWatch and TSMWatch messages are processed in the rules file and converted into alerts, like other events. [Table E-22](#) shows the elements common to ProbeWatch and TSMWatch events.

Table E-22 Common ProbeWatch and TSMWatch Elements

Element	Description
Summary	Summary string, described in the following tables.
Node	Name of the node on which the Cisco Info Mediator or TSM is running.
Agent	Name of the Cisco Info Mediator or TSM.
Manager	ProbeWatch or TSMWatch.

Table E-23 describes summary strings common to all Cisco Info Mediators and TSMs.

Table E-23 Common ProbeWatch and TSMWatch Summary Strings

ProbeWatch/TSMWatch Message	Description	Action
Running...	The Cisco Info Mediator or TSM has started running.	N/A
Unable to get events...	The Cisco Info Mediator or TSM was unable to start up.	See your support contract for information about contacting the help desk.
Going down...	The Cisco Info Mediator or TSM is shutting down.	N/A

See the *Cisco Info Mediator Reference, 3.6* for additional summary strings for each Cisco Info Mediator. TSMWatch messages are in the same format as ProbeWatch messages. Table E-24 describes summary strings common to all TSMs.

Table E-24 Common TSMWatch Summary Strings

TSMWatch Message	Description
Connection Attempted...	Messages relating to the establishment of a TCP/IP connection.
Connection Succeeded...	
Connection Failed...	
Connection Timed out...	
Connection Lost...	
Disconnection Attempted...	Messages relating to relinquishing a TCP/IP connection.
Disconnection Succeeded...	
Disconnection Failed...	
Wakeup Attempted...	Messages relating to wake up functionality.
Wakeup Succeeded...	
Wakeup Failed...	

Table E-24 Common TSMWatch Summary Strings (continued)

TSMWatch Message	Description
Login Attempted... Login Succeeded... Login Timed out... Login Failed...	Messages relating to logging in to a host.
Logout Attempted... Logout Succeeded... Logout Timed out... Logout Failed...	Messages relating to logging out from a host.
Heartbeat Sent... Heartbeat Received... Heartbeat Timed out...	Messages relating to sending/receiving heartbeat messages to/from the host.
Resynchronization Attempted... Resynchronization Succeeded... Resynchronization Failed...	Messages relating to synchronizing current alerts between the switch and Cisco Info Center.

