

CA Spectrum®

Watches User Guide

Release 9.4



This Documentation, which includes embedded help systems and electronically distributed materials, (hereinafter referred to as the "Documentation") is for your informational purposes only and is subject to change or withdrawal by CA at any time.

This Documentation may not be copied, transferred, reproduced, disclosed, modified or duplicated, in whole or in part, without the prior written consent of CA. This Documentation is confidential and proprietary information of CA and may not be disclosed by you or used for any purpose other than as may be permitted in (i) a separate agreement between you and CA governing your use of the CA software to which the Documentation relates; or (ii) a separate confidentiality agreement between you and CA.

Notwithstanding the foregoing, if you are a licensed user of the software product(s) addressed in the Documentation, you may print or otherwise make available a reasonable number of copies of the Documentation for internal use by you and your employees in connection with that software, provided that all CA copyright notices and legends are affixed to each reproduced copy.

The right to print or otherwise make available copies of the Documentation is limited to the period during which the applicable license for such software remains in full force and effect. Should the license terminate for any reason, it is your responsibility to certify in writing to CA that all copies and partial copies of the Documentation have been returned to CA or destroyed.

TO THE EXTENT PERMITTED BY APPLICABLE LAW, CA PROVIDES THIS DOCUMENTATION "AS IS" WITHOUT WARRANTY OF ANY KIND, INCLUDING WITHOUT LIMITATION, ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NONINFRINGEMENT. IN NO EVENT WILL CA BE LIABLE TO YOU OR ANY THIRD PARTY FOR ANY LOSS OR DAMAGE, DIRECT OR INDIRECT, FROM THE USE OF THIS DOCUMENTATION, INCLUDING WITHOUT LIMITATION, LOST PROFITS, LOST INVESTMENT, BUSINESS INTERRUPTION, GOODWILL, OR LOST DATA, EVEN IF CA IS EXPRESSLY ADVISED IN ADVANCE OF THE POSSIBILITY OF SUCH LOSS OR DAMAGE.

The use of any software product referenced in the Documentation is governed by the applicable license agreement and such license agreement is not modified in any way by the terms of this notice.

The manufacturer of this Documentation is CA.

Provided with "Restricted Rights." Use, duplication or disclosure by the United States Government is subject to the restrictions set forth in FAR Sections 12.212, 52.227-14, and 52.227-19(c)(1) - (2) and DFARS Section 252.227-7014(b)(3), as applicable, or their successors.

Copyright © 2014 CA. All rights reserved. All trademarks, trade names, service marks, and logos referenced herein belong to their respective companies.

CA Technologies Product References

This guide references CA Spectrum®.

Contact CA Technologies

Contact CA Support

For your convenience, CA Technologies provides one site where you can access the information that you need for your Home Office, Small Business, and Enterprise CA Technologies products. At <http://ca.com/support>, you can access the following resources:

- Online and telephone contact information for technical assistance and customer services
- Information about user communities and forums
- Product and documentation downloads
- CA Support policies and guidelines
- Other helpful resources appropriate for your product

Providing Feedback About Product Documentation

If you have comments or questions about CA Technologies product documentation, you can send a message to techpubs@ca.com.

To provide feedback about CA Technologies product documentation, complete our short customer survey which is available on the CA Support website at <http://ca.com/docs>.

Contents

Chapter 1: Working with Watches	7
About Watches.....	7
Thresholds And Watches Subview	8
Create a Watch.....	9
Create and Edit an Alarm Cause Description	13
Copy an Alarm	13
Alarm Script Example	14
Activate or Deactivate a Watch.....	16
Watch Activation and Deactivation from the Command Line	17
Edit a Watch	17
Inheritable Watch Edits.....	18
Copy a Watch	18
Delete a Watch.....	18
Display Watch Information	19
Generate a Report on Multiple Watches	19
 Chapter 2: Event Management	 21
Events.....	21
Configure Events	21
Event Codes.....	23
Event Format Files.....	24
Event Variables.....	24
 Chapter 3: Watch Expression Formula Rules	 27
Watch Expressions	27
Primitives	28
Constants	34
Casting Operators	34
Data Type of a Literal Number	35
Attributes	35
Instance Identifiers.....	36
Watch Definition Attributes.....	37
Watch Destination Attributes	38
Threshold Reference and Reset Compatibility	38

Appendix A: Watch Type Examples 39

First Watch Scenario	39
Second Watch Scenario	41
Evaluate-On-Change Watches.....	43
Polled Threshold Watches.....	44
First On-Demand Watch Scenario	45
Second On-Demand Watch Scenario	46
Third On-Demand Watch Scenario.....	48
Fourth On-Demand Watch Scenario	50
First Usability and Testing Watch Scenario	51
Second Usability and Testing Watch Scenario	52
Third Usability and Testing Watch Scenario.....	53
Fourth Usability and Testing Watch Scenario	54

Index 57

Chapter 1: Working with Watches

This section contains the following topics:

[About Watches](#) (see page 7)

[Thresholds And Watches Subview](#) (see page 8)

[Create a Watch](#) (see page 9)

[Activate or Deactivate a Watch](#) (see page 16)

[Edit a Watch](#) (see page 17)

About Watches

A *watch* is a mechanism for adding thresholds for model attributes. Watches let you monitor network elements, such as routers, with a high level of detail. They also provide current data that can be used with other CA Spectrum tools in network analysis.

Watch administration comprises two main components: an intelligent circuit in the SpectroSERVER that performs monitoring functions, and a OneClick administration interface. Watch administration is available under the Thresholds And Watches view on the Information tab for a model.

You can dynamically apply watches on any type of attribute and can monitor attributes against thresholds to generate events and alarms. You can also copy the properties of a watch to another watch. You can thus retain the first watch information while adding new information to the second watch.

You can set watches on the internal and external attributes of any model type. You can also set multiple watches for an attribute. For example, you can set two packet rate threshold watches on a device. One threshold generates a yellow alarm when the value exceeds 10,000, and another threshold generates a red alarm for values above 15,000.

Note: We recommend familiarizing yourself with CA Spectrum administration before setting up watches. For more information, see the *Database Management Guide*, the *Administrator Guide*, and the *SpectroSERVER Performance Administration Guide*. Familiarity with the OneClick interface is also helpful.

The watch feature lets you perform the following tasks:

- Dynamically apply watches on any type of attribute.
- Monitor attributes against thresholds and generate events and alarms.

- Execute scripts when watches are violated or when they are reset.
- Generate reports about watches.

Note: Watches that you create in one release of CA Spectrum can be migrated to later releases.

Thresholds And Watches Subview

You can create, configure, and administer watches in OneClick. View and configure watches from a table in the Thresholds And Watches subview.

Note: You can access the Thresholds and Watches subview from the Information tab for a model.

The Watches table displays information for each watch defined on that model. The Watch Status column displays the watch condition with color codes as follows:

Gray

Indicates that the watch is inactive. The watch is not currently running because it has not been activated.

Blue

Indicates the initial state of the watch. The watch is activated but has yet to run for the first time.

Green

Indicates that the watch is active and running without any violation.

Yellow

Indicates that the watch threshold is violated.

Red

Indicates that the watch failed to evaluate. The text explains the reason.

The toolbar buttons let you do the following:

- Activate
- Deactivate
- Create
- Edit
- Copy
- Delete
- Display watch information

- Print watch information
- Export the Watches table

Create a Watch


You can create watches to monitor selected attributes of network models.

Follow these steps:

1. Click the Information tab for a model, and expand the Thresholds And Watches subview.

2. Expand the Watches subview.

The Watches table appears.

3. Click  (Create a new watch) on the toolbar in the Watches subview.

The Create Watch dialog opens.

Note: The Expression tab is selected by default.

4. Enter a name for the new watch and select a data type from the Data Type list.

Note: This data type is the watch destination attribute. The watch expression must evaluate to this data type.

5. Build the expression for the watch by combining operators, functions, and model type attributes:

- Click a button to the left of the expression area to insert an operator or function into the watch expression at the cursor.
- Click the Attributes button and select the required attribute to insert an attribute.

Note: Set the PollingStatus attribute to True to activate a watch.

6. Click the Properties tab and specify the following settings:

Active By Default

Specifies whether the watch is active by default for all models that inherit the watch. If this option is not set, activate the watch manually for the desired models.

Important! Setting Active By Default for a polled watch can adversely affect SpectroSERVER performance.

Evaluate On Demand

Evaluates the watch expression only when the watch attribute is read.

Evaluate On Change

Evaluates the watch expression if any attribute in the watch expression changes. The attributes must have either the Memory or Database flag set.

Evaluate By Polling

Evaluates the watch after each poll interval.

Poll Interval

Specifies the polling frequency in seconds. This field is enabled only if you select Evaluate By Polling.

Note: A watch does not become active if the poll interval is set to 0.

Create watch on model type *model type*

Specifies the model type where the watch is created. Click Browse to select a different parent model type.

Note: The watch is created on the model type of the selected model by default. However, you can select a different parent model type for the watch to let other derived model types inherit the watch.

Make Inheritable

Makes the watch available for models of all model types that are derived from the selected model type.

7. Click the Threshold tab and select the Attach a Threshold check box if you want to attach a threshold value to the watch.

The Comparison, Notification, and Script options are enabled.

8. Specify the following Comparison options. The Comparison options specify the threshold settings that specify the boundaries or limits to regulate watch notifications:

Threshold violated if value

Specifies the operator for threshold violations. For example, select "greater than". The threshold is considered violated if the sample returns a value that exceeds the value in the threshold attribute. For the threshold violation value, specify a constant or click Attributes to select an attribute.

Threshold reset if value

Determines when a threshold status of Violated is reset to Normal. For each subsequent sample after a threshold violation, this reset value is compared to the watch expression using the comparison primitive in the opposite direction. That is, if the comparison primitive is \geq (greater than or equal to), the status is reset to Normal as soon as a sample returns a value that falls below this reset value. The reset value is not applicable when the Threshold Comparison is set to $=$ or \neq because the watch is considered violated if the value is either greater than or less than the watch expression, irrespective of the direction. In such a situation, this option is disabled; otherwise, this entry is mandatory. For the threshold reset value, specify a constant or click the Attributes button to select an attribute.

9. Specify the following Notification options. The Notification options let you specify the type of notification that is sent when the threshold is violated.

No Notification

Prevents notification generation when a threshold is violated.

Generate Event(s)

Specifies the events to be generated by the CA Spectrum event management system when the watch is violated or reset. Selecting this option enables fields for event codes as follows:

- when threshold is violated—Specifies the event that is generated when the threshold is violated.
- when threshold is reset—Specifies the event that is generated when the threshold is reset.
- when deactivated/all instances are reset—Specifies the event that is generated when all instances of this watch are reset or when a violated watch is deactivated.

You can further configure these events to generate or clear alarms and to participate in Event Rules and Procedures. Events are not required for all fields; events are only generated for those fields that contain valid event codes.

Generate Alarm

Specifies an alarm to be generated directly if the watch threshold is violated. A generic event (0x480004) is also generated and associated with the alarm. If the watch resets and the alarm is cleared, a generic reset event (0x480005) is generated. Selecting this option enables the following fields:

- Severity—Specifies the alarm severity as Minor, Major, or Critical.
- Description—Specifies the alarm cause description. A generic description is provided by default. The Browse button lets you select a different cause description or create a new one. Newly created descriptions are stored on the OneClick server in the directory `$INSTALL/custom/Events/CsPCause`. If you have multiple OneClick servers, copy the files manually.

- User clearable—When set, lets you manually clear the watch alarm.
- Reset watch upon user clearing alarm—Resets the watch if the alarm is manually cleared even if the watch has not reached its reset value. The alarm can be generated again on a subsequent violation. This option is only available if User Clearable is selected.

10. Specify the following Script options. The Script options let you specify a script to execute if the watch threshold is violated or cleared.

Execute a Script when threshold

Enables script execution. Select the threshold condition that triggers script execution.

Execute for each instance

If enabled and the watch expression contains list attributes, the script is executed for each instance that meets the threshold condition.

Script File

Specifies the script file on the SpectroSERVER to execute. If no directory path is specified, the default directory, `<$SPECROOT>/SS-Tools/SwScript`, is used. You can change the default directory by setting `sw_script_path` in the file `<$SPECROOT>/SS/.watchrc`.

Note: Scripts are executed by the user who starts the SpectroSERVER. Therefore, that user requires the privileges to access and execute the scripts. If a permission problem is detected during watch creation or modification, an error message appears. An event is generated if privileges are changed later and the problem is detected in an attempt to execute the script.

11. Click the Landscapes button to specify the destination landscapes for the watch. In the OneClick environment, you can distribute watches to multiple landscapes in a distributed server environment. New watches are created in all landscapes by default.

A dialog opens with a list of landscapes.

12. Move the landscapes in which you do not want to create the watch to the right.

Note: If a landscape is not available when the watch is created, you can add the watch to that landscape later. Use the [Edit Watch dialog](#) (see page 17) and click Landscapes.

13. Click OK.

The watch is created.




More information:

[Attributes](#) (see page 35)

Create and Edit an Alarm Cause Description

You can create and edit cause descriptions for alarms that are generated for threshold violations. A notification of a threshold violation is delivered through the CA Spectrum event and alarm facilities. These features support notification delivery by telephone, pager, or by email using an external script interface.



Follow these steps:

1. Select a watch and click  (edit).
The watch opens in the Edit Watch dialog.
2. Select the Threshold tab, and select Generate Alarm.
The alarm options are displayed.
3. Click Browse.
The Select Alarm Cause Code dialog opens. The alarm details appear at the bottom.
4. Click  (create) to create a new alarm cause or select one from the list and click  (edit) to edit it.
A dialog opens to let you create or edit the alarm cause.
Note: The Cause Code field is read-only.
5. Enter information in the fields as required, and click OK.
The alarm cause description is created or modified.

Copy an Alarm

Use the Copy feature to create multiple alarm cause descriptions that differ only minimally from each other.

Follow these steps:

1. Select a watch in the Watches table and click  (Edit).
The watch opens in the Edit Watch dialog.
2. Select Generate Alarm on the Threshold tab and click Browse.
The Select Alarm Cause Code dialog opens.
3. Select an alarm cause description from the list and click  (Copy).
The Copy Alarm Cause dialog opens.

4. Enter the cause code.
5. (Optional) Edit other alarm properties.
6. Click OK.

The new alarm cause description is created.

Alarm Script Example

The following sample is an example of a UNIX script file that you can create as a notification. You can designate scripts for execution whenever an alarm is set or cleared in conjunction with a threshold watch. This example script is named *sw_alm_script* and is included in the SS-Tools/SwScript directory when you install CA Spectrum.

```
#!/bin/sh
#####
This is an example script that can used by users to perform important tasks upon
threshold violation of a watch (Watch Status is #VIOLATED) or when an already violated
watch becomes normal (Watch Status #is NORMAL).
#
# In each of the above cases, the user-specified script is executed (specified during
watch creation) and provides twenty arguments. The arguments in ascending order are:
#
# 1) DATE                - The date on which the watch was violated.
# 2) TIME                - The time at which the watch was violated.
# 3) MTYPE_NAME          - The model type of the watch.
# 4) MODEL_HANDLE        - The modelhandle of the model.
# 5) MODEL_NAME          - The model of the watch.
# 6) INSTANCE            - The instance (port, board, etc) for which the watch
is either violated or reset.
# 7) ALARM_ID            - The ID of the alarm.
# 8) CONDITION            - The CONDITION of the alarm.
# 9) CAUSE_CODE          - The cause code of the alarm.
# 10) REPAIR_PERSON      - The repair person assigned to troubleshoot.

# 11) ALARM_STATUS        - The status of the alarm.
# 12) SCRIPT_TYPE        - This is set to "VIOLATED" if the script is getting
executed upon violation and it is set to # "NORMAL" if the script is getting executed
upon reset.
# 13) WATCH_NAME        - The name of the watch for which the script is executed.
# 14) WATCH_CREATOR      - The name of the creator of the watch.
# 15) WATCH_SRC          - The formula for the watch expression.
# 16) WATCH_SRC_VAL      - The value of the watch expression formula.
# 17) WATCH_REF          - The formula for threshold reference.
# 18) WATCH_REF_VAL      - The value of the threshold reference.
# 19) WATCH_RES          - The formula for threshold reset.
# 20) WATCH_RES_VAL      - The value of the threshold reset.
#####
```

```
## Save the argument values into variables that could be used later.
## Remember that DATE is the first argument and WATCH_RES_VAL is the last argument.
# IMPORTANT NOTE:
## Pay particular attention to the variable, "SCRIPT_TYPE". If the watch has a violated
status, SCRIPT_TYPE is set to "VIOLATED". If a watch has normal status, SCRIPT_TYPE
is set to "NORMAL". You can be use this to decide what to do when the watch is violated
and when it is normal.
while [ "$#" -ne 0 ]
do
case "$#"
in
20) DATE=`echo "$1"`;;
19) TIME=`echo "$1"`;;
18) MTYPE_NAME=`echo "$1"`;;
17) MODEL_HANDLE=`echo "$1"`;;
16) MODEL_NAME=`echo "$1"`;;
15) INSTANCE=`echo "$1"`;;
14) ALARM_ID=`echo "$1"`;;
13) CONDITION=`echo "$1"`;;
12) CAUSE_CODE=`echo "$1"`;;
11) REPAIR_PERSON=`echo "$1"`;;
10) ALARM_STATUS=`echo "$1"`;;
9) SCRIPT_TYPE=`echo "$1"`;;
8) WATCH_NAME=`echo "$1"`;;
7) WATCH_CREATOR=`echo "$1"`;;
6) WATCH_SRC=`echo "$1"`;;
5) WATCH_SRC_VAL=`echo "$1"`;;
4) WATCH_REF=`echo "$1"`;;
3) WATCH_REF_VAL=`echo "$1"`;;
2) WATCH_RES=`echo "$1"`;;
1) WATCH_RES_VAL=`echo "$1"`;;
    esac
    shift
done
```



```
## Compose a message that can be used, for example, to send mail.
## The composed message can be used for any other purpose as well.
message()
{
echo "Watch Status Notification"
    echo ""
    echo "Watch Status is $1"
    echo ""

echo "Date:                $DATE"
echo "Time:                $TIME"
echo "MType Name:          $MTYPE_NAME"
echo "Model Handle:        $MODEL_HANDLE"
echo "Model Name:          $MODEL_NAME"
echo "Instance:            $INSTANCE"
echo "Alarm ID:             $ALARM_ID"
echo "Condition:           $CONDITION"
echo "Cause Code:           $CAUSE_CODE"
echo "Repair Person:        $REPAIR_PERSON"
echo "Alarm Status:         $ALARM_STATUS"
echo "Watch Name:           $WATCH_NAME"
echo "Watch Creator:        $WATCH_CREATOR"
echo "Watch Expression:     $WATCH_SRC"
echo "Watch Expression Value: $WATCH_SRC_VAL"
echo "Watch Reference:      $WATCH_REF"
echo "Watch Reference Value: $WATCH_REF_VAL"
echo "Watch Reset:          $WATCH_RES"
echo "Watch Reset Value:    $WATCH_RES_VAL"}
## Mail the composed message to the interested users.
## <USER LIST> is the list of mail recipients.
message "$SCRIPT_TYPE" | mail <USER LIST>
```

Activate or Deactivate a Watch

A watch is active or inactive by default, depending on how it was created.

No limits apply to the number or form of the watches that you can add to the system. This flexibility also means that careless use can deplete system resources and result in performance degradation. Even though a single watch consumes minimal platform resources and generates little network traffic, we recommend deactivating or deleting unnecessary watches. Also exercise caution when setting up a watch that references a large list of device attributes.

To activate or deactivate a watch for a single model, select the watch from the Watches table and click  (activate watch) or  (deactivate watch). You can also select multiple watches to activate or deactivate at once.

You can activate or deactivate multiple watches on multiple models using the Attribute Editor. When you select a watch attribute and move it to the right panel of the Attribute Editor, the Activate Watch and Deactivate Watch options are enabled.

Note: For more information about the Attribute Editor, see the *Modeling and Managing Your IT Infrastructure Administrator Guide*.

Watch Activation and Deactivation from the Command Line

In addition to the OneClick Console, you can activate or deactivate a watch and view data profiles from the command line.

The following command lists the applicable watch data for the model type:

```
show watch [mh=model_handle] [lh=landscape_handle]
```

The following command updates the watch status:


```
update action=action_code [watch=watch_id] [mh=modelhandle]
```

Note: The [watch=watch_id] argument is only applicable to activate (0x00480003) and deactivate (0x00480004) actions.

Edit a Watch

You can edit a watch to change its expression, properties, and thresholds.

Follow these steps:

1. Click the Information tab for a model, and expand the Thresholds And Watches subview.
2. Expand the Watches subview.
3. Select the watch in the Watches table and click  (Edit).
The watch opens in the Edit Watch dialog.
4. Change the watch expression, properties, and thresholds as required, and click OK.

Note: For more information about the watch expression, properties, and thresholds, see [Create a Watch](#) (see page 9).

The watch is edited.

Inheritable Watch Edits

When you edit an inheritable watch, the watch definition on the selected model type is modified by default. You can select a different parent model type for the watch. Use the Browse button in the Model Type section of the Properties tab.


If a watch definition is modified for a model type that was derived from the watch's originating model type, the changes only take effect for models of that type and its derived types. The model type where the watch originated is not changed.

For example, a watch is originally created on Gen_IF_Port and is inheritable. Serial_IF_Port is derived from Gen_IF_Port, so Serial_IF_Port models inherit the watch. If the watch definition is modified on the Gen_IF_Port model type, the changes are propagated to Serial_IF_Port models. However, if the watch definition is modified on Serial_IF_Port, the new version overrides the definition on Gen_IF_Port. Only Serial_IF_Port models are affected, while Gen_IF_Port models retain the original watch definition. As a result of the inheritance rules, you can redefine or override inheritable watches differently for derived model types.

Copy a Watch

Use the Copy feature to create multiple watches that differ only minimally from each other.


Follow these steps:

1. Select a watch in the Watches table, and click  (Copy Watch).
The watch opens.
2. Enter a name for the new watch, change any other information as needed, and click OK.
The watch is copied and a new watch is created.

Delete a Watch

You can delete a watch if you no longer require the associated threshold monitoring.

Follow these steps:

1. Select the watch in the Watches table and click  (Delete).
If the watch exists in multiple landscapes, a dialog opens. The left side displays the landscapes where the watch exists and from which it will be deleted.


2. (Optional) To preserve the watch on a landscape, move that landscape to the right.
3. Click OK.

The watch is deleted from the landscapes that are listed on the left.

Display Watch Information

You can display watch information to see a report about the watch. You can also print and export the information to a text file or HTML file.

Follow these steps:

1. Select a watch in the Watches table and click  (information).
The Watch Report dialog displays the watch information.
2. (Optional) Click Print to print the watch information.
3. (Optional) Click Export to save the information to a text or HTML file.
The watch is printed or exported.

Generate a Report on Multiple Watches

You can generate a report about multiple watches from the OneClick Administration page.

Follow these steps:

1. Open the OneClick home page in a browser, and click the Administration tab.
The Administration Pages opens.
2. Click the Watch Reports link.
The Generate Watch Report dialog opens. The Select Landscape list and the Select Model Type(s) list are displayed.
3. Select the landscape and model types for which you want to generate a report, and click Generate Report.
The report is generated.

Chapter 2: Event Management

This section contains the following topics:

[Events](#) (see page 21)

[Configure Events](#) (see page 21)

Events

To illustrate CA Spectrum advanced event processing, consider the following scenario. For testing purposes, you want to temporarily allow a high child count. You want to generate an alarm only if the high child count condition persists for more than one minute, and to clear any such alarms when the watch is reset. Using an EventPair Event Rule, CA Spectrum can perform the following tasks:

- Generate an initial event when the ChildLimit watch is violated.
- Suppress the alarm unless that initial event is *not* followed by a reset event within 60 seconds.
- Generate another event when all instances of this watch are either reset or deactivated.

Note: For more information about event rules and creating events, see the *Event Configuration User Guide*.


Configure Events

The following procedure is an example of event configuration. Building from the example in the previous topic, the first event is "Initial child count too high" (0xffffffff), and the second event is "Reset" (0xffffffffe). If the child count threshold is violated and is not reset within 60 seconds, a "Child count too high" alarm (event/alarm 0xffffffffd) is generated.

Follow these steps:

1. Select Utilities, Event Configuration from the Tools menu.

The Event Configuration page appears with the Navigation, Contents, and Details panels.

2. Click  (Create Event) in the Navigation panel.

The Create Event dialog appears, with Event Code populated with a default value.

3. (Optional) Edit the event code and enter an event message and click OK.

The event is created and its details appear in the Contents and Details panels on the right.

4. Create and save the initial Child count too high event (0xffffffff).

This event triggers the EventPair rule that conditionally generates an alarm and a Reset event (0xffffffffe). The watch uses the Reset event to evaluate the persistence of the child count condition and to clear any alarms that the EventPair rule generates when the watch is reset.

5. Create and save an event "0xffffffffd" and a corresponding alarm that is used to generate a notification if the child count is persistently high.
6. Create and save an Event Pair rule for the initial "Child count too high" event (0xffffffff) that generates the "0xffffffffd" event and alarm if the Reset event (0xffffffffe) is not received within 60 seconds.

Important! If the SpectroSERVER is updated while an event is being processed using an Event Rule, the processing is not completed. All event rules that are processing when this functionality is invoked are flushed.

7. Click File, Save All.

The new events are saved.

8. Open a watch for editing.

9. On the Threshold tab, select Generate Event(s) under Notification.

The event code fields appear.

10. Enter the initial "Child count too high" event (0xffffffff) that you previously created in the 'when threshold is violated' field.

11. Enter the alarm-clearing "Reset" event (0xffffffffe) that you previously created in the 'when threshold is reset' field, and click OK.

The event is associated with the watch.

If the LAN_802_3 container contains 5 or more models, the initial "Child count too high" event is generated, and the event rule is evaluated. An alarm is generated only if the reset event is not received within 60 seconds of the initial event. If the watch is reset, any "Child count too high" alarms that the watch generates are cleared.

The EventDisp entries (if needed) for this example are as follows:

```
# make sure child count too high (0xffffffff) is
# reset (0xffffffffe) within 60 seconds
# otherwise generate alarming event (0xffffffffd)0xffffffff E 50 R Aprisma.EventPair,
0xffffffffe, 0xffffffffd, 60

# generate an alarm on persistent child count too high0xffffffffd E 50 A 1,0xffffffffd

# if child count drops below threshold, clear any existing alarms0xffffffffe E 50 C
0xffffffffd
```

Event Codes

Event codes are generated in the CA Spectrum Event Log for each event.

Note: An event is not generated for watch creation.

The following list contains event codes and their descriptions:

0x0048000a

Watch could not be activated for a model

0x00480000

Modification of a watch

0x00480001

Destruction of a watch

0x00480002

Activation of a watch

0x00480003

Deactivation of a watch

0x00480004

Violation of a watch threshold

0x00480005

Resetting of threshold status for a watch (with variable reset value)

0x00480006

Violation of an instance of a watch

0x00480007

Resetting of threshold status for an instance of a watch (with variable reset value)

0x00480010

Bad probable cause message

0x00480012

Logging interval conflict

0x00480013

Error executing a script (for threshold watches only)

0x00480014

Resetting a watch threshold for a model, with reason

Event Format Files

You can use the existing event format files Event00480004 and Event00480005 (located in `<$SPECROOT>/SG-Support/CsEvFormat`) as templates to create your own watch-related event format files.

Note: For more information, see the *Event Configuration User Guide*.

Event Variables

The events for watch threshold violation (0x00480004) and reset (0x00480005) have variable bindings. This applies both to watches that generate alarms directly and watches that generate events. The clear event has a single variable binding, ID #2, containing the watch name.

The following table shows the binding variables and their contents:

Variable Binding ID	Contents
2	Watch name
4	Threshold reference value
5	Threshold reset value (only for reset events)
6	Comparison string value
7	Calculated watch value
8	Instance object identifier (OID) if the watch has instances, otherwise empty
9	Internal; strings to complete the respective event message
10	Internal; strings to complete the respective event message

Variable Binding ID	Contents
11	Internal; strings to complete the respective event message

Chapter 3: Watch Expression Formula Rules

This section contains the following topics:

[Watch Expressions](#) (see page 27)

[Attributes](#) (see page 35)

[Threshold Reference and Reset Compatibility](#) (see page 38)

Watch Expressions

A watch expression defines what is monitored by the watch. The watch expression can be as simple as the name of an attribute or as complex as a formula. A watch expression includes attributes and constant values that are related to each other through mathematical symbols and Boolean operators, which are referred to as primitives.

Expressions are evaluated proceeding from left to right in the following order of precedence for primitives:

Precedence	Primitives
1	.# .
2	()
3	DELTA () COUNTER_DELTA ()
4	!
5	&
6	* /
7	+ -
8	= = != >= <= < >

More information:

[Attributes](#) (see page 35)

Primitives

Primitives are typically used in expressions that have a left side and a right side. For example, in $A+B$, $+$ is the primitive defining the relationship between the left side, A , and the right side, B . Parentheses are used to indicate the order of evaluation for expressions with multiple components.

The available primitives are as follows:

- $+$
- $-$
- $*$
- $/$
- $==$
- $!=$
- $>$
- $<$
- $>=$
- $<=$
- $!$ (not)
- $\&$ (and)
- $,$
- $|$
- $($
- $)$
- TRUE
- FALSE
- TIME
- DELTA
- MIN
- MAX
- INTEGER
- REAL
- UNSIGNED

- COUNTER_DELTA
- UNSIGNED64

You can group primitives by function and applicability. You can apply the following primitives only between text strings or in numerical expressions:

- == equal to
- >= greater than or equal to
- <= less than or equal to
- != not equal to
- > greater than
- < less than

Note: These primitives do not support the attribute types Object ID, IP Addr, and Boolean.

You can apply the following primitives only for numerical expressions:

- + addition
- - subtraction
- * multiplication
- / division

These primitives do not support the attribute types Text String, Object ID, IP Addr, and Boolean. Supported types are evaluated in the following order:

- Real
- Time Ticks, Date, Gauge, Counter
- Enumeration Integer

If elements are from two different levels, they are evaluated as belonging to the higher level. For example, $5 + 5$ is evaluated as an integer, while $5 + 5.1$ is evaluated as a real.

Note: CA Spectrum model types use the attribute type Counter to store unsigned long integers. If you use the unsigned primitive illegally in an expression, for example, "Text String + UNSIGNED(5)", the resulting error message is of type Counter.

You can apply the following primitives only to Boolean expressions:

- ! logical not
- & and
- | or

The following primitives are composed of words instead of symbols:

TIME

Represents the current time as the number of seconds since January 1, 1970, 00:00 Greenwich Mean Time.

DELTA

Calculates the change of an attribute over the intervals of the sampling frequency. For example, a watch on an integer attribute named int1 at 30-second intervals might produce DELTA values as follows:

Interval	Values
0	100, 0
30	1000 900
60	1000 0
90	-1000 -2000
120	2000 3000

The DELTA primitive supports the following attribute types:

- Integer
- Enumeration
- Real
- Short
- Time Ticks
- Date
- Gauge
- Counter

In cases where the value is never expected to go down (such as with Time Ticks and Counter), use COUNTER_DELTA.

You can instance attributes in DELTA expressions as follows:

- DELTA (If_In_Octets.2)
- DELTA (If_Out_Octets.#)
- DELTA (TIME) is also supported

Note: DELTA (TIME) is the difference between now and the last evaluation of the watch; it is useful only for calculations involving internal attributes. To generate a rate calculation involving external attributes, use the device's time, for example, DELTA (XfIDELTA (Sys_Up_Time), where Sys_Up_Time is the time counter on the device.

COUNTER_DELTA

Calculates the change of an attribute whose value is assumed to be an increasing unsigned integer. The value may be reset to zero, but the result of this primitive is always a positive unsigned integer. Negative attribute values are treated as large unsigned values.

The COUNTER_DELTA primitive supports the same attribute types as DELTA, and you can also instance attributes within COUNTER_DELTA expressions as described for DELTA. COUNTER_DELTA (TIME) is also supported.

ATTR

Indicates that the hexadecimal number in parentheses following the primitive is an attribute identifier. This primitive lets an attribute to be uniquely identified when duplicate names exist as shown in the following examples:

```
ATTR (<attr_id>)  
ATTR (<attr_id>).<instance id>  
ATTR (<attr_id>).#
```

The <attr_id> value is "0x" or "0X" followed by between one and eight hexadecimal numerals.

TRUE

Defines the Boolean constant True.

FALSE

Defines the Boolean constant False.

MAX

Identifies the larger of two expressions separated by a comma and enclosed by parentheses, for example, (attribute x+1, attribute y-5). The expressions may consist of any combination of operands, attributes, or primitives that results in a numeric value (non-text, string, octet, bool).

MIN

Identifies the smaller of two expressions separated by a comma within parentheses, for example, (attribute x+1, attribute y-5). The expressions may consist of a combination of operands, attributes, or primitives that results in a numeric value (non-text, string, octet, bool).

You can use the following primitives (not shown on the pop-up selector menu) to indicate the instances of a list attribute that are monitored by a watch.

.

Indicates that the following entry identifies a specific instance of a list attribute.

For example, If_In_Octets.2 specifies the second instance of the attribute If_In_Octets. In the case of an IP address table, the identifier might be an entire address instead of a single digit.

A watch fails if the specified instance of an attribute does not exist even for one of the models for which the watch is activated. In cases where the instance exists for some models but not for others, the watch succeeds for those models where it is present and fails for others.

.#

Indicates that the current Instance Specifier value shown in the Watch Detail View determines which instances are monitored.

For example, if the Instance Specifier value is ALL, the expression If_In_Octets.# applies the watch to all instances of the If_In_Octets attribute. However, if the Instance Specifier value is RANGE (1-3), only instances 1, 2, and 3 are monitored.

In cases where a range of instances is specified, the watch fails even if one of the instances in the range is not present for one of the models. Conversely, the watch is successful for any model where all of the instances in the range are present.

Data Types

The following table shows the data type values that you can assign to the various attribute types.

Note: Although CA Spectrum does not support watches of attributes with the data type OCTET STRING, you can watch such attributes using the TEXT STRING data type.

Expression Result Type	Acceptable Destination Attribute Types
BOOLEAN	Any Numeric Type
TEXT STRING	Text String
INTEGER	Integer, Enumeration, Real
ENUMERATION	Integer, Enumeration, Real
REAL	Integer, Enumeration, Real
DATE	Real, Date, Time Ticks, Counter, Gauge, Counter64
TIME TICKS	Real, Date, Time Ticks, Counter, Gauge, Counter64
COUNTER	Real, Date, Time Ticks, Counter, Gauge, Counter64
GAUGE	Real, Date, Time Ticks, Counter, Gauge, Counter64
OBJECT Id	OBJECT Id
IP ADDRESS	IP ADDR
COUNTER64	COUNTER64, Real

For example, given an integer attribute named Int1, the expression `Int1 = 50.5` (requires a cast) is allowed, while `Int1 = "a string"` is not allowed.

The rules are most flexible for assigning values to Boolean and Text String attributes. Any expression can be evaluated to a 0 and 1 and can be written to a Boolean attribute. For example, given Bool1 as a Boolean attribute, the expression `Bool1 = 500 * 50 + 450` assigns TRUE to Bool1. The expression `Bool1 = (500 * 50 + 450) * 0` assigns FALSE to Bool1. Similarly, any expression can be evaluated to a text string. Given a Text String attribute named str1, the expression `str1 = 500 + 50` assigns the string "550" to str1.

Note: The Text String attribute type accepts text strings in quotation marks, and arithmetic expressions that are not contained in quotation marks, such as `500 + 50`, which results in a string of "550."

Constants

Like attribute values and operators, you can enter constants directly into the expression field. You can enter the following types of constants in expression formulas:

- **UnsignedInteger**, represented by a sequence of one or more digits with a positive value.
- **Real**, represented by a sequence of zero or more digits followed by a dot, followed by a sequence of one or more digits. For example, .7 or 1.7 or 23.24 (but not 7).
- **Signed Integer**, represented by a sequence of one or more digits with either a positive or negative value.
- **Text String**, represented by any sequence of characters that are enclosed by double quotation marks. For example, "a string" or "5.25" or " ".

Casting Operators

Casting forces one data type into another. When you perform casting, you risk losing some portion of the value being cast. Casting is unnecessary when the range of the source data type fits into the range of the destination data type.

For example, you can assign an integer to a real number without casting because $-1.79769 \text{ e}+308 \leq -2,147,483,648$ and $2,147,483,647 \leq 1.79769 \text{ e}+308$. However, you must use casting when assigning an integer to a counter because the ranges do not overlap.

Casting works correctly for the overlapping ranges of the involved data types. For example, casting the integer 5 to a counter behaves as expected. But negative numbers do not fit inside of the range of Counter. As a result, casting -5 to a Counter produces the unsigned (positive) result of 4,294,967,291 because of the way computers represent numbers. Conversely, casting a counter value larger than 2,147,483,647 into an integer produces a negative number.

You can use the following operators to cast the result of an expression in parentheses to a selected data type.

Operator	Result of Casting
UNSIGNED	Normally, 27 is treated as a signed integer; however, "UNSIGNED (27)" is interpreted as the unsigned integer 27. If a negative number appears in the expression following this operator, it is interpreted as unsigned. Example: UNSIGNED (-5) is interpreted as 5. The number, -5 is UNSIGNED with a value of: 4294967291.

Operator	Result of Casting
INTEGER	Any real number in the expression following this operator is rounded up or down to the nearest integer. Examples: INTEGER (2.4) is interpreted as 2 and INTEGER (2.6) is interpreted as 3.
REAL	Example: REAL (3) is interpreted as 3.0.

Data Type of a Literal Number

A numeric literal in a watch expression is compared against a series of ranges to determine its data type. The Type column in the table below is the actual data type of the numeric literal, which is directly interchangeable with the other types that are listed in that column cell.

The following table shows how the data type of a literal number is determined:

Minimum	Maximum	Type
-2,147,483,648	2,147,483,647	Integer, Boolean*, Enumeration
0	4,294,967,295	Counter, Date**, Gauge, TimeTicks
0	18,446,744,073,709,551,615	Counter64
-1.79769 e+308	1.79769 e+308	Real

* You can also use the constants TRUE and FALSE for Boolean data types. Any non-zero value is equivalent to TRUE.

**The number of seconds from Jan 1, 1970 00:00:00 UTC(0).

Attributes

The attributes that you enter in watch expression formulas must always begin with a letter and consist of a combination of letters, digits and underscore characters (_). Enclose attributes with names identical to primitives (TRUE, FALSE, TIME, DELTA, COUNTER_DELTA) in single quotes. You can also specify attributes by attribute ID using the ATTR primitive.

A [watch expression](#) (see page 27) can consist of a single attribute. To create a multi-attribute expression, select an operator from the button palette, then select another attribute and so on.

If a list attribute is specified in the expression, you must include instance information. To specify an instance, append a '.' followed by the instance ID number (for example, **ifInOctets.2**). Or append '#' (for example, **ifInOctets.#**), and select either All or Range in the Instance field. For Range, specify the low and high IDs in the From and To fields. List attributes are indicated with "[]" in the Type column in the Attribute Selector dialog.

In addition to the attributes that you enter as part of watch expressions, you can also dynamically create the attributes that are required to store watch information and destination attributes.

Instance Identifiers

You can use instance identifiers to specify a particular object when you are using list attributes in a watch expression formula.

List attributes require instance IDs to fully identify an object. You can specify a specific instance, a range of instances, or all the possible instances of an attribute in a watch expression. When an OID is built, it uses up to two different sub-OIDs: the standard SpectroSERVER OID mechanism or the Watch Instance Specifier. The SpectroSERVER creates an OID from the OID prefix and the OID reference (optional). When you specify an instance for the watch expression on the Expression page, it is appended to the OID as follows:

OID = OID prefix + OID reference + Watch instance

If you use the Instance Specifier to specify a range, a result for each instance in the range is computed. If the range specified is larger than the actual instances, results are only computed for the actual instances. That is, if you specify a range of 1-10, and there are only 3 instances, only 3 results are computed.

For example, you want to set a watch on **If_In_Octets** for a router. You supply a range of 1-3 as the Instance Specifier. Because the OID prefix that the CA Spectrum database uses for **If_In_Octets** is 1.3.6.1.2.1.2.2.1.10, the following OIDs are assigned:

- OID = 1.3.6.1.2.1.2.2.1.10 + .1
- OID = 1.3.6.1.2.1.2.2.1.10 + .2
- OID = 1.3.6.1.2.1.2.2.1.10 + .3

If you use ALL as the Instance Specifier, all of the instances for the If_In_Octets object are dynamically determined. You can also specify an instance in the watch expression, or you can mix specific instances of an object with RANGE or ALL Instance Specifiers. Using the Instance_of primitive (".") to specify an instance of an object overrides the Instance Specifier for that object.

For example, in the following formula, the Instance_of primitive (followed by a value) is used to specify instance number 3 of the If_In_Errors object, and the .# primitive indicates that the current Instance Specifier should be used for the If_In_Octets object.

```
If_In_Errors.3 / If_In_Octets.#
```

Assume that the current Instance Specifier is ALL, and both If_In_Errors and If_In_Octets are list attributes with the OIDs 1.3.6.1.2.1.2.2.1.14 and 1.3.6.1.2.1.2.2.1.10 respectively. The division operation for each instance would be performed, using the following OIDs:

```
If_In_Errors OID = 1.3.6.1.2.1.2.2.1.14 + .3
```

```
If_In_Octets OID = 1.3.6.1.2.1.2.2.1.10 + .1
```

```
If_In_Errors OID = 1.3.6.1.2.1.2.2.1.14 + .3
```

```
If_In_Octets OID = 1.3.6.1.2.1.2.2.1.10 + .2
```

```
If_In_Errors OID = 1.3.6.1.2.1.2.2.1.14 + .3
```

```
If_In_Octets OID = 1.3.6.1.2.1.2.2.1.10 + .n
```

Notice how the Instance_of primitive overrides the Instance Specifier for the If_In_Errors attribute OID. Instead of each attribute using each possible instance as a result of selecting ALL, the Instance_of primitive allows specification of a particular instance. However, the Instance Specifier still determines the number of instances of a particular watch.

The INSTANCE_ID of the model and the Instance Specifier is applied to each list attribute in the watch expression. If the watch expression contains attributes that use different instances, an error is returned when you attempt to add the watch.

For example, if you enter one formula that contains an attribute that uses board number as an instance and another formula that uses board port as an instance, an error is returned.

Watch Definition Attributes

When you create a watch for a model, an attribute is created to store the watch details. The attribute has a computer-generated name. It is created using the active database Developer ID (registered or default) in the Watch Definition group under the model type to which the model belongs.

Watch Destination Attributes

When you create a watch for a model, a destination attribute is also created with the same name and data type as the watch. This attribute resembles any other model type attribute. When it is read, the watch expression is automatically evaluated. The result of the evaluation is the result of the read operation.

Threshold Reference and Reset Compatibility

For threshold watches, the threshold and reset expressions must be of a type that is compatible with the watch expression.

The following table shows the result types and the corresponding acceptable types:

Result Type	Acceptable Types
BOOLEAN	Boolean
TEXT STRING	Text String
INTEGER	Integer, Enumeration, Real
ENUMERATION	Integer, Enumeration, Real
REAL	Real
DATE	Real, Date, Time Ticks, Counter, Gauge, Counter64
TIME TICKS	Real, Date, Time Ticks, Counter, Gauge, Counter64
COUNTER	Real, Date, Time Ticks, Counter, Gauge, Counter64
GAUGE	Real, Date, Time Ticks, Counter, Gauge, Counter64
OBJECT Id	OBJECT Id
IP ADDRESS	IP_ADDR
COUNTER64	Counter64, Real

Appendix A: Watch Type Examples

This section contains the following topics:

- [First Watch Scenario](#) (see page 39)
- [Second Watch Scenario](#) (see page 41)
- [Evaluate-On-Change Watches](#) (see page 43)
- [Polled Threshold Watches](#) (see page 44)
- [First On-Demand Watch Scenario](#) (see page 45)
- [Second On-Demand Watch Scenario](#) (see page 46)
- [Third On-Demand Watch Scenario](#) (see page 48)
- [Fourth On-Demand Watch Scenario](#) (see page 50)
- [First Usability and Testing Watch Scenario](#) (see page 51)
- [Second Usability and Testing Watch Scenario](#) (see page 52)
- [Third Usability and Testing Watch Scenario](#) (see page 53)
- [Fourth Usability and Testing Watch Scenario](#) (see page 54)

First Watch Scenario

The following parameters establish a watch on Hub_CSI_IRBM. This watch checks the total number of frames transmitted and collisions received every 60 seconds. It generates a minor alarm with the message "Too many collisions" if the threshold value of one million is exceeded. The Threshold Reset Value indicates that the threshold status is reset from Violated to Normal (and the alarm is cleared) when the total number of collisions falls to 500,000.

You can create watches for this scenario as follows:

1. Create an inactive watch of type Counter and name it HubColls_v. Assign the total number of collisions of 1000000 in the expression.
2. Create an inactive watch of type Counter and name it HubColls_r. Assign the total number of collisions of 500000 in the expression.
3. Create an active watch of type Counter and name it HubColls and assign it values as shown in the following watch examples.

Examples: Watches

The first watch consists of the following parameters:

- Name: HubColls_v
- Data Type: Counter

- Expression:
 - Expression: 1000000
- Instance: None
- Properties
 - Default Activation: Inactive
- Threshold: None

The second watch consists of the following parameters:

- Name: HubColls_r
- Data Type: Counter
- Expression:
 - Expression: 500000
- Instance: None
- Properties
 - Default Activation: Inactive
- Threshold: None

The third watch consists of the following parameters:

- Name: HubColls
- Data Type: Counter
- Expression
 - Expression: Hub_Trans_Coll+Hub_Rec_Colls
- Instance: None
- Properties
 - Default Activation: Active
 - Evaluated: By Polling
 - Poll Interval: 00:01:00
- Threshold
 - Attach a Threshold (checked) Threshold Violated if value: > HubColls_v (create a counter watch named HubColls_v Inactive, expression is 1000000.)
 - Threshold reset if value: HubColls_r (create Counter watch named HubColls_r Inactive, expression is 500000.)

- The attributes, HubColls_v and HubColls_r must be created before the creation of the HubColls watch.
- Generate Alarm (checked) Severity: Minor Alarm Desc: CollsExceeded (Create a new alarm named CollsExceeded with the message "Too Many Colls".)

Second Watch Scenario

A network administrator wants to monitor the level of disk utilization on a server, but wants to exclude CD-ROM drives from monitoring. CD-ROM drives typically show 0% utilization if no CD is present, but they show close to 100% utilization if a full CD is in the drive. However, such information is not useful to the administrator.

To check disk utilization, the administrator wants to use RFC2790App, which returns device types as OIDs. Each OID maps to a different device type. Any expression that must check against a storage type in this MIB must use its OID for comparison. (You can compare strings to OIDs).

The following table shows RFC2790App Device Type/OID Mapping.

Device Type	OID
hrStorage	1.3.6.1.2.1.25.2
hrStorageTypes	1.3.6.1.2.1.25.2.1
hrStorageOther	1.3.6.1.2.1.25.2.1.1
hrStorageRam	1.3.6.1.2.1.25.2.1.2
hrStorageVirtualMemory	1.3.6.1.2.1.25.2.1.3
hrStorageFixedDisk	1.3.6.1.2.1.25.2.1.4
hrStorageRemovableDisk	1.3.6.1.2.1.25.2.1.5
hrStorageFloppyDisk	1.3.6.1.2.1.25.2.1.6
hrStorageCompactDisk	1.3.6.1.2.1.25.2.1.7
hrStorageRamDisk	1.3.6.1.2.1.25.2.1.8
hrStorageFlashMemory	1.3.6.1.2.1.25.2.1.9
hrStorageNetworkDisk	1.3.6.1.2.1.25.2.1.10

The administrator must create two new watches on the RFC2790App model. The first watch sets up a Boolean value that is used in the second watch to exclude CD-ROM drives (OID 1.3.6.1.2.1.25.2.1.7) from monitoring. The second watch monitors the disk utilization (expressed as a percentage) of other types of storage devices on the host. In this example, an alarm is sent if utilization exceeds 90%.

Examples: Watches

The first watch consists of the following parameters:

- Name: isNotCDROM
- Data Type: Boolean
- Expression: Expression: hrStorageType.# != 1.3.6.1.2.1.25.2.1.7
- Instance: All
- Properties
 - Default Activation: Active
 - Evaluate: On Demand
- Threshold: None

The second watch consists of the following parameters:

- Name: PctDiskUsed
- Data Type: Real
- Expression
 - Expression: $\text{REAL}(\text{hrStorageUsed.\#}) / \text{REAL}(\text{hrStorageSize.\#}) * 100 * \text{REAL}(\text{isNotCDROM.\#})$
 - Instance: All
- Properties
 - Default Activation: Active
 - Evaluate: By Polling
 - Poll Interval: 30 seconds
- Threshold
 - Threshold violated if value \geq *<set to the acceptable level of disk utilization>*
- Alarm
 - Alarm Severity: Minor
 - Alarm Description: DiskUtilAlarm
 - Alarm is user clearable
 - Watch is not reset upon user clearing of alarm.
- Script: None

Evaluate-On-Change Watches

The following watches are examples of evaluate-on-change (EoC) watches. These watches determine when a view was edited. As a result, the administrator can run a search that shows exactly when LANs were edited and when new models were added to them. Both attributes are evaluated on change, and neither attribute is likely to change frequently.

Examples: Evaluate-on-Change Watches

The first watch consists of the following parameters:

- Name: Child_Count_Watch
- Data Type: Integer
- Expression
 - Expression: Child_Count
 - Instance: None
- Properties
 - Default Activation: Active
 - Evaluate: By EoC
 - Poll Interval: None
- Threshold: None

The second watch consists of the following parameters:

- Name: Edit_Count
- Data Type: Integer
- Expression
 - Expression: EDIT_COUNT
 - Instance: None
- Properties
 - Default Activation: Active
 - Evaluate: By EoC
- Threshold: None

Polled Threshold Watches

In the following example, a Polled Threshold watch is created to generate a minor alarm each time the number of children in a LAN_802_3 exceeds 5. The watch clears the alarm when the count of children falls below 5.

First, create the attribute watch ChildLimit_v (an arbitrary name) and give it an expression of "5". Creating a watch depicts the creation of this watch in detail. Then create a polled threshold watch named Child_Limit.

Examples: Polled Threshold Watches

The first watch consists of the following parameters:

- Name: ChildLimit_v
- Data Type: Counter
- Expression: 5
- Properties: None
- Threshold: None

The second watch consists of the following parameters:

- Name: ChildLimit
- Data Type: Integer
- Expression
 - Expression: Child_Count
- Instance: None
- Properties
 - Default Activation: Active
 - Evaluated: By Polling
 - Poll Interval: 00:01:00
- Threshold
 - Attach a Threshold (checked)
 - Threshold Violated if value: > ChildLimit_v
 - Threshold reset if value: <= ChildLimit_v
 - Generate Alarm (checked)
 - Severity: Minor
 - Alarm Description: ExceedChildLimit (Create a new alarm named ExceedChildLimit with the message "Only 5 children allowed.")

First On-Demand Watch Scenario

On-Demand watches are evaluated whenever information is requested from them.

The following examples of simple On-Demand watches work with a polled Threshold watch that generates an alarm when a Cisco router (model type: Rtr_CiscoIGS) starts running out of memory. The attribute, freeMem, is less than 1500 and clears when it exceeds 2500.

Examples: Simple On-Demand Watches Working With a Polled Threshold Watch

The first watch consists of the following parameters:

- Name: MemLow
- Data Type: Integer
- Expression
 - Expression: 1500
 - Instance: None
- Properties
 - Default Activation: Inactive
 - Evaluated: On Demand
- Threshold: None

The second watch consists of the following parameters:

- Name: MemHigh
- Data Type: Integer
- Expression
 - Expression: 2500
 - Instance: None
- Properties
 - Default Activation: Inactive
 - Evaluated: On Demand
- Threshold: None

The third watch consists of the following parameters:

- Name: Mem_Good
- Data Type: Integer

- Expression
 - Expression: freeMem
 - Instance: None
- Properties
 - Default Activation: Active
 - Evaluate: By Polling
 - Poll Interval: 1 minute
- Threshold
 - Threshold violated if value: <= MemLow
 - Threshold reset if value: > MemHigh
- Alarm
 - Alarm Severity: Major
 - Alarm Description: Rtr+Memory
- Script: None

Second On-Demand Watch Scenario

On-Demand watches are evaluated whenever information is requested from them.

The following watches are designed to show the network administrator colored alarm conditions as the performance of a Cisco router (model type: Rtr_Cisco) progressively degrades. A minor alarm has a value of 50 -59; a major alarm has a value of 60 - 69, and the value of a critical alarm exceeds 70.

Examples: On-Demand Watches

The first watch consists of the following parameters:

- Name: True_Ref
- Data Type: Boolean
- Expression
 - Expression: 1
 - Instance: None
- Properties
 - Default Activation: Inactive
 - Evaluate: On Demand
- Threshold: None

The second watch consists of the following parameters:

- Name: Minor_Busy
- Data Type: Boolean
- Expression
 - Expression: ((busyPer >= 50) & (busyPer <60))
 - Instance: None
- Properties
 - Default Activation: Active
 - Evaluate: By Polling
 - Poll Interval: 1 minute
- Threshold
 - Threshold violated if value: > True_Ref
 - Threshold reset if value: <= True_Ref
- Alarm:
 - Alarm Severity: Minor
 - Alarm Description: RtrBusy1
 - Script: None

The third watch consists of the following parameters:

- Name: Major_Busy
- Data Type: Boolean
- Expression
 - Expression: ((busyPer >= 60) & (busyPer < 70))
 - Instance: None
- Properties
 - Default Activation: Active
 - Evaluate: By Polling
 - Poll Interval: 1 minute
- Threshold
 - Threshold: Threshold violated if value: == True_Ref
 - Threshold reset if value: != True_Ref

- Alarm
 - Alarm Severity: Major
 - Alarm Description: RtrBusy 2
 - Script: None

The fourth watch consists of the following parameters:

- Name: Critical_Busy
- Data Type: Boolean
- Expression
 - Expression: (busyPer >= 70)
 - Instance: None
- Properties
 - Default Activation: Active
 - Evaluate: By Polling
 - Poll Interval: 60
- Threshold
 - Threshold violated if value: == True_Ref.
 - Threshold reset if value: != True_Ref.
- Alarm:
 - Alarm Severity: Critical
 - Alarm Description: RtrBusy3
 - Script: None

Third On-Demand Watch Scenario

On-Demand watches are evaluated whenever information is requested from them.

For this scenario, a network administrator wants to see an alarm anytime a Frame Relay link fails, causing the dial-backup lines to activate. The dial backup is ISDN Interface Type 21 (the basic ISDN service). The following watches verify whether traffic is received on any ISDN port. If traffic is flowing, a minor alarm is placed on the Cisco router (model type: Rtr_Cisco).

Examples: On-Demand Watches

The first watch consists of the following parameters:

- Name: True_Ref
- Data Type: Boolean
- Expression
 - Expression: 1
 - Instance: None
- Properties
 - Default Activation: Inactive
 - Evaluate: On Demand
- Threshold: None

The second watch consists of the following parameters:

- Name: ISDN_Backup
- Data Type: Boolean
- Expression
 - Expression: ((COUNTER_DELTA (ifInOctets.#) > 0) & (ifType.# == 21))
 - Instance: All
- Properties
 - Default Activation: Active
 - Evaluate: By Polling
 - Poll Interval: 1 minute
- Threshold
 - Threshold violated if value: == True_Ref
 - Threshold reset if value: != True_Ref
- Alarm:
 - Alarm Severity: Minor
 - Alarm Description: Backup_Active
- Script: None

Fourth On-Demand Watch Scenario

On-Demand watches are evaluated whenever information is requested from them.

For this scenario, the watch monitors the ifOutOctets value on ISDN interface models to see whether it increases or is non-zero. This value indicates whether the interface is active. You can modify the polling, logging, alarm severity, and alarm text to suit your requirements. The second watch uses a true or false (1/0) indicator: 1 indicates that the interface is sending ifOutOctets, and 0 indicates that the interface is not sending.

Examples: On-Demand Watches

The first watch consists of the following parameters:

- Name: isISDN
- Data Type: Boolean
- Expression
 - Expression: (X_ifType == 21)
 - Instance: None
- Properties
 - Default Activation: Active
 - Evaluate: On Demand
 - Inheritable: False
- Threshold
 - Threshold: None

The second watch consists of the following parameters:

- Name: ISDN_Up
- Data Type: Integer
- Expression
 - Expression: MAX(0, MIN(1, INTEGER((((COUNTER_DELTA (X_OutOctets) * INTEGER(isISDN)) > 0))))
 - Instance: None
- Properties
 - Default Activation: Active
 - Evaluate: By Polling
 - Poll Interval: 0+00:05:00
 - Inheritable: False

- Threshold:
 - Threshold violated if value == 1
 - Threshold reset if value != 1
- Alarm
 - Alarm Severity: Minor
 - Alarm Description: ErrorTholdAlarm
 - Alarm is user clearable
 - Watch is not reset upon user clearing of alarm.
- Script: None

First Usability and Testing Watch Scenario

The example watches that are described here are used for usability and testing.

For this scenario, you create two watches, one to create an attribute named WatchLoad_v with an expression value of .75, and a second watch to use the CPUloadRate attribute. In the following example, you create an Alarm Description named WatchLoad_Alarm that says, for example, “The CPU load has exceeded 75%.”

Examples: Usability and Testing Watches

The first watch consists of the following parameters:

- Name: WatchLoad_v
- Data Type: Real
- Expression
 - Expression: .75
 - Instance: None
- Properties
 - Default Activation: Inactive
 - Evaluate: On Demand
- Threshold: None

The second watch consists of the following parameters:

- Name: CPU_Load
- Data Type: Real

- Expression
 - Expression: CPULoadRate
 - Instance: None
- Properties
 - Default Activation: Active
 - Evaluate: By Polling
 - Poll Interval: 1 minute
- Threshold
 - Threshold violated if value: > WatchLoad_v
 - Threshold reset if value: < = WatchLoad_v
- Alarm
 - Alarm Severity: Minor
 - Alarm Description: WatchLoad_Alarm
 - Script: None

Second Usability and Testing Watch Scenario

The example watches that are described here are used for usability and testing.

In this example, you create an EoC watch that generates a minor alarm when Composite Condition attribute of a container model exceeds 4. You first create a reference attribute named `conditionCheck_ref` and use that attribute in a watch named `ConditionCheck`. Then use the Alarm Description dialog box to create an Alarm Description named “ConditionCheckAlarm” (or another appropriate string).

The reference attribute has the following parameters:

- Name: `ConditionCheck_ref`
- Data Type: Integer
- Expression
 - Expression: 4
 - Instance: None
- Properties: None
- Threshold: None

Example: Usability and Testing Watch

The watch consists of the following parameters:

- Name: ConditionCheck
- Data Type: Integer
- Expression
 - Expression: Composite_Condition
 - Instance: None
- Properties
 - Default Activation: Active
 - Evaluate: EoC
- Threshold
 - Threshold violated if value: > ConditionCheck_ref
 - Threshold reset if value: <= ConditionCheck_ref
- Alarm:
 - Alarm Severity: Minor
 - Alarm Description: ConditionCheckAlarm
- Script: None

Third Usability and Testing Watch Scenario

The example watches that are described here are used for usability and testing.

In this example, if the load value falls below 10% for longer than 90 minutes, you create an alarm. To configure this watch, change the watch expression to use your desired external attributes, and set the threshold to the number of consecutive polls equal to the desired time duration, divided by the polling interval.

Examples: Usability and Testing Watches

The first watch consists of the following parameters:

- Name: Watch_Load_Under_10_Pct
- Data Type: Integer
- Expression
 - Expression: MAX (0, MIN (1, (<Load under 10% test expression here>)))
 - Instance: None

- Properties
 - Default Activation: Inactive
 - Evaluation: On demand
- Threshold: None

The second watch consists of the following parameters:

- Name: Watch_TimeTicker_LoadUnder10Pct
- Data Type: Integer
- Expression
 - Expression: (Watch_TimeTicker_LoadUnder10Pct + 1) * Watch_Load_Under_10_Pct
 - Instance: None
- Properties
 - Default Activation: Inactive
 - Evaluation: On demand
- Threshold: *The desired number of consecutive polls*

Fourth Usability and Testing Watch Scenario

The example watches that are described here are used for usability and testing.

The first watch monitors the CPU for a sustained referenced usage value (80% in this example).

The second watch triggers an alarm if the CPU usage remains at a certain level (80%) for a sustained period of time. This watch calculates this time period using the threshold value (3), multiplied by the polling interval (5 minutes). Therefore, this watch violates the threshold and triggers an alarm if the CPU usage exceeds 80% for 15 minutes. You can adjust these values to suit your requirements.

Examples: Usability and Testing Watches

The first watch consists of the following parameters:

- Name: CPU_Duration_Over_80
- Data Type: Integer
- Expression
 - Expression: MAX(0, MIN(1, INTEGER((INTEGER(cpqHoCpuUtilMin.#) >= 80))))
 - Instance: All

- Properties
 - Default Activation: Active
 - Evaluate: On Demand
 - Inheritable: False
- Threshold: None

The second watch consists of the following parameters:

- Name: CPU_Time_Duration
- Data Type: Integer
- Expression
 - Expression: ((CPU_Time_Duration.# + 1) * CPU_Duration_Over_80.#)
 - Instance: All
- Properties
 - Default Activation: Active
 - Evaluate: By Polling
 - Poll Interval: 0 + 00:05:00
 - Inheritable: False
- Threshold
 - Threshold violated if value ≥ 3
 - Threshold reset if value < 3
- Alarm
 - Alarm Severity: Minor
 - Alarm Description: ErrorTholdAlarm
 - Alarm is user clearable
 - Watch is not reset upon user clearing of alarm.
 - Script: None

Index

A

- activating
 - multiple watches • 16
 - watches • 16
 - watches from the command line • 17
- alarm scripts • 14
- alarms
 - copying • 13
 - creating • 13
 - editing • 13
 - script example • 14
- attributes
 - about • 35
 - creating • 35
 - watch definition • 37
 - watch destination • 38

C

- casting operators • 34
- configuring
 - events • 21
- constants • 34
- copying
 - alarms • 13
 - watches • 18
- creating
 - alarm cause descriptions • 13
 - watches • 9

D

- data types
 - about • 33
 - of literal numbers • 35
- deactivating
 - multiple watches • 16
 - watches • 16
 - watches from the command line • 17
- deleting
 - watches • 18
- displaying
 - watch information • 19

E

- editing

- alarm cause descriptions • 13
- inheritable watches • 18
- watches • 17
- EoC watches • 43
- evaluate-on-change watches
 - about • 43
 - examples • 43
- events
 - about • 21
 - codes • 23
 - configuring • 21
 - format files • 24
 - variables • 24
- examples
 - alarm scripts • 14
- expression evaluation • 27

G

- generating
 - reports on multiple watches • 19

I

- instance identifiers • 36

O

- on-demand/pollled threshold watches
 - on-demand/pollled threshold scenarios • 45, 46, 48, 50

P

- polled threshold watches
 - examples • 44
- prerequisites • 7
- primitives • 28

R

- reset expressions • 38

S

- scenarios • 39, 41, 45, 46, 48, 50, 51, 52, 53, 54

T

- threshold reference • 38

threshold watches • 38

W

watch expressions • 27

watch type

examples • 39, 41

watches

about • 7, 43

activating • 16

activation from the command line • 17

copying • 18

creating • 9

deactivating • 16

deactivation from the command line • 17

defined • 7

definition attributes • 37

deleting • 18

destination attributes • 38

displaying • 19

editing • 17, 18

evaluate-on-change • 43

events • 21

examples • 39, 41, 45, 46, 48, 51, 52, 53, 54

expression formula rules • 27

functionality • 7

polled threshold • 44

prerequisites • 7

status • 8

table • 8

usability and testing scenarios • 51, 52, 53, 54