CA Spectrum®

监视用户指南

版本 9.3

CA Technologies 产品引用

本指南参考 CA Spectrum®。

联系技术支持

要获取在线技术帮助以及办公地址、主要服务时间和电话号码的完整列表，请联系技术支持：http://www.ca.com/worldwide。

目录

第 1 章： 使用监视

此部分包含以下主题：

关于监视

*监视*是一种为模型属性添加阈值的机制。您可以使用监视功能来详细监控路由器等网络元素。它们所提供的当前数据还可供网络分析中的其他 CA Spectrum 工具使用。

监视管理包括两个主要组件：在 SpectroSERVER 中执行监控功能的智能线路以及 OneClick 管理界面。您可以在模型“信息”选项卡上的“阈值和监视”视图下进行监视管理。

您可以对任何类型的属性动态应用监视，并可以针对阈值监控属性以生成事件和警报。您还可以将一个监视的属性复制到另一个监视中。这样，您可以在保持第一个监视的信息的同时，将新信息添加到第二个监视中。

您可以对任何模型类型的内部和外部属性设置监视。您还可以为一个属性设置多重监视。例如，您可以对一个设备设置两个数据包速率阈值监视。一个阈值在值超过 10,000 时生成黄色警报；另一个阈值在值超过 15,000 时生成红色警报。

注意：我们建议您在设置监视之前先熟悉 CA Spectrum 管理。有关详细信息，请参阅*《数据库管理指南》*、*《管理员指南》*和*《SpectroSERVER 性能管理指南》* 。熟悉 OneClick 界面也有帮助。

监视功能可让您执行以下任务：

- 对任何类型的属性动态应用监视。
- 针对阈值监控属性并生成事件和警报。
- 违反或重置监视时执行脚本。
- 生成监视报告。

注意：可以将您在一个版本的 CA Spectrum 中创建的监视迁移到更高版本中。

“阈值和监视”子视图

您可以在 OneClick 中创建、配置和管理监视。可以从“阈值和监视”子视图的表中查看和配置监视。

注意：您可以从模型的“信息”选项卡访问“阈值和监视”子视图。

“监视”表显示该模型上定义的每个监视的信息。“监视状态”列利用颜色代码显示监视状况，如下所示：

灰色

指示监视处于非活动状态。监视尚未激活，因此当前未运行。

蓝色

指示监视处于初始状态。监视已激活，但尚未首次运行。

绿色

指示监视处于活动状态，正在运行且未出现任何违反。

黄色

指示违反了监视阈值。

红色

指示监视未能评估。相关文本会解释原因。

工具栏按钮可让您执行以下操作：

- 激活
- 停用
- 创建
- 编辑
- 复制
- 删除
- 显示监视信息
- 打印监视信息
- 导出“监视”表

创建监视

您可以创建监视以监控网络模型的选定属性。

遵循这些步骤:

1. 单击模型的“信息”选项卡，并展开“阈值和监视”子视图。
2. 展开“监视”子视图。

   此时将显示“监视”表。
3. 在“监视”子视图中的工具栏上，单击  （创建新监视）。

   此时将打开“创建监视”对话框。

   注意：默认情况下选中“表达式”选项卡。
4. 为新监视输入名称，并从“数据类型”列表中选择数据类型。

   注意：此数据类型为监视目标属性。监视表达式的值必须为此数据类型。
5. 通过组合运算符、函数和模型类型属性，构建监视表达式：
   - 单击表达式区域左侧的按钮，以在监视表达式中的光标处插入运算符或函数。
   - 单击“属性”按钮并选择所需属性，以插入属性。

     注意：将 PollingStatus 属性设置为 True 可激活监视。
6. 单击“属性”选项卡并指定以下设置：

   默认情况下处于活动状态

   指定对于继承监视的所有模型，默认情况下监视是否处于活动状态。如果未设置此选项，需手动为所需模型激活监视。

   重要说明！对轮询的监视设置“默认情况下处于活动状态”，可能会对 SpectroSERVER 性能产生负面影响。

   根据要求进行评估

   仅当读取监视属性时，才计算监视表达式。

   更改时进行评估

   当监视表达式中的任何属性发生更改时，计算监视表达式。这些属性必须设置有“内存”或“数据库”标志。

   通过轮询评估

   在每个轮询时间间隔后评估监视。

轮询时间间隔

指定轮询频率（以秒为单位）。仅当您选择“通过轮询评估”时，才启用此字段。

注意：如果轮询时间间隔设置为 0，则监视不会变成活动状态。

在模型类型 <*模型类型*> 上创建监视

指定在其上创建监视的模型类型。单击“浏览”可选择不同的父模型类型。

注意：默认情况下在所选模型的模型类型上创建监视。不过，您可以为监视选择不同的父模型类型，以允许其他派生模型类型继承该监视。

使可继承

使监视可供从所选模型类型派生的所有模型类型的模型使用。

7. 如果您希望为监视附加一个阈值，请单击“阈值”选项卡，并选中“附加阈值”复选框。

   此时将启用“比较”、“通知”和“脚本”选项。

8. 指定下列“比较”选项。“比较”选项指定用于指定调整监视通知的边界或限制的阈值设置：

   违反阈值，如果值

   指定违反阈值的运算符。例如，选择“大于”。如果样本返回的值超过阈值属性中的值，则视为违反阈值。对于违反阈值的值，指定常量或者单击“属性”以选择属性。

   重置阈值，如果值

   确定何时将阈值状态“已违反”重置为“正常”。对于出现阈值违反后的每个后续样本，将使用反方向上的比较基元来比较该重置值与监视表达式。也就是说，如果比较基元为 >=（大于或等于），则当样本返回的值降到低于此重置值时，状态将重置为“正常”。将阈值比较设置为 == 或 != 时，重置值不可用，因为值大于或小于监视表达式时，都会视为违反监视（不管方向如何）。在这种情况下，会禁用此选项；否则，此项是强制性的。对于阈值重置值，请指定一个常量或者单击“属性”按钮以选择属性。

9. 指定下列“通知”选项。“通知”选项可让您指定在违反阈值时发送的通知类型。

无通知

违反阈值时，阻止生成通知。

生成事件

指定在违反或重置监视时 CA Spectrum 事件管理系统将生成的事件。选择此选项将启用事件代码的字段，如下所示：

- 当阈值被违反时 - 指定当违反阈值时生成的事件。
- 当阈值被重置时 - 指定当重置阈值时生成的事件。
- 当停用时/所有实例重置时 - 指定当重置此监视的所有实例或者停用违反的监视时生成的事件。

您可以进一步配置这些事件，以便生成或清除警报并参与事件规则和过程。并非所有字段都需要事件；只会针对那些包含有效事件代码的字段生成事件。

生成警报

指定在违反监视阈值时直接生成警报。还将生成一般事件 (0x480004) 并将其与该警报关联。如果重置监视且清除了警报，将生成一般重置事件 (0x480005)。选择此选项会启用以下字段：

- 重要级别 - 指定警报重要级别为次要、主要还是关键。
- 说明 - 指定警报原因说明。默认情况下提供一般性说明。“浏览”按钮可让您选择不同的原因说明，或者创建新的原因说明。新创建的说明存储在 OneClick 服务器上的目录 $INSTALL/custom/Events/CsPCause 中。如果您具有多个 OneClick 服务器，请手动复制这些文件。
- 用户可清除 - 设置此字段后，允许您手动清除监视警报。
- 用户清除警报时重置监视 - 手动清除警报时重置监视，即使监视尚未达到其重置值也是如此。可以对后续违反再次生成警报。仅当选择了“用户可清除”时，此选项才可用。

10. 指定下列“脚本”选项。“脚本”选项可让您指定在违反或清除监视阈值时将执行的脚本。

执行脚本，当阈值

启用脚本执行。选择触发脚本执行的阈值条件。

为每个实例执行

如果启用此选项，并且监视表达式包含列表属性，则为符合阈值条件的每个实例执行脚本。

脚本文件

指定 SpectroSERVER 上要执行的脚本文件。如果未指定目录路径，则使用默认目录 <*$SPECROOT*>/SS-Tools/SwScript。您可以通过在文件 <*$SPECROOT*>/SS/.watchrc 中设置 sw_script_path 来更改默认目录。

注意：脚本由启动 SpectroSERVER 的用户执行。因此，该用户必须具有访问和执行这些脚本的权限。如果在创建或修改监视期间检测到权限问题，则显示错误消息。如果后来更改了权限，并在试图执行脚本时检测到问题，则生成事件。

11. 单击“格局”按钮，指定要监视的目标格局。在 OneClick 环境中，您可以将监视分配给分布式服务器环境中的多个格局。默认情况下在所有格局中创建新监视。

    将打开一个对话框，其中显示有格局列表。

12. 将您不希望在其中创建监视的格局移至右侧。

    注意：如果在创建监视时某个格局不可用，则可以稍后向该格局中添加监视。使用“编辑监视”对话框 (p. 17)，然后单击“格局”。

13. 单击“确定”。

    监视即会创建。

详细信息：

属性 (p. 35)

创建和编辑警报原因说明

您可以创建和编辑为阈值违反生成的警报的原因说明。阈值违反的通知通过 CA Spectrum 事件和警报设施传送。这些功能支持使用外部脚本接口通过电话、寻呼机或电子邮件传送通知。

遵循这些步骤:

1. 选择监视，然后单击  （编辑）。

   将在“编辑监视”对话框中打开该监视。

2. 选择“阈值”选项卡，然后选择“生成警报”。

   此时将显示警报选项。

3. 单击“浏览”。

   此时将打开“选择警报原因代码”对话框。在底部显示警报详细信息。

4. 单击  （创建）以创建新警报原因，或者从列表中选择一个警报原因，然后单击  （编辑）以对其进行编辑。

   此时会打开一个对话框，允许您在其中创建或编辑警报原因。

   注意：“原因代码”字段是只读的。

5. 根据需要在字段中输入信息，然后单击“确定”。

   警报原因说明即会创建或修改。

复制警报

使用“复制”功能可创建彼此仅有很少区别的多个警报原因说明。

遵循这些步骤:

1. 在“监视”表中选择监视，然后单击  （编辑）。

   将在“编辑监视”对话框中打开该监视。

2. 在“阈值”选项卡上选择“生成警报”，然后单击“浏览”。

   此时将打开“选择警报原因代码”对话框。

3. 从列表中选择警报原因说明，然后单击  （复制）。

   此时将打开“复制警报原因”对话框。

4. 输入原因代码。
5. （可选）编辑其他警报属性。
6. 单击“确定”。

   新警报原因说明即会创建。

警报脚本示例

以下示例是一个您可以作为通知创建的 UNIX 脚本文件示例。您可以指定与阈值监视配合使用的脚本，以便在每当设置或清除警报时执行。此示例脚本名为 *sw_alrm_script*，安装 CA Spectrum 时，SS-Tools/SwScript 目录中包含它。

```
#!/bin/sh
############################################################
这是一个示例脚本，用户可使用它在监视存在阈值违反（监视状态为 #VIOLATED）或已违反的监视变为正常（监视状态为 #NORMAL）时执行重要任务。
#
# 在上述每种情况下，执行用户指定的脚本（创建监视时指定），同时提供二十个参数。这些参数是（按升序排序）：
#
# 1) DATE                       - 违反监视的日期
# 2) TIME                       - 违反监视的时间。
# 3) MTYPE_NAME                 - 监视的模型类型。
# 4) MODEL_HANDLE               - 模型的模型句柄。
# 5) MODEL_NAME                 - 监视的模型。
# 6) INSTANCE                   - 违反或重置的监视的实例（端口、主板等）。
# 7) ALARM_ID                   - 警报的 ID。
# 8) CONDITION                  - 警报的条件。
# 9) CAUSE_CODE                 - 警报的原因代码。
# 10) REPAIR_PERSON             - 分配来排除故障的修复人员。
# 11) ALARM_STATUS              - 警报的状态。
# 12) SCRIPT_TYPE               - 如果在违反时执行脚本，则将其设置为“VIOLATED”；如果在重置时执行脚本，则将其设置为“NORMAL”。
# 13) WATCH_NAME        -  为其执行脚本的监视的名称。
# 14) WATCH_CREATOR             - 监视的创建者的名称。
# 15) WATCH_SRC         - 监视表达式的公式。
# 16) WATCH_SRC_VAL             - 监视表达式公式的值。
# 17) WATCH_REF         - 阈值参考的公式。
# 18) WATCH_REF_VAL             - 阈值参考的值。
# 19) WATCH_RES         - 阈值重置的公式。
# 20) WATCH_RES_VAL             - 阈值重置的值。
############################################################
```

```
## 将参数值保存到变量中，方便以后使用。
## 记住，DATE 是第一个参数，WATCH_RES_VAL 是最后一个参数。
# 重要说明：
## 请特别注意变量“SCRIPT_TYPE”。如果监视具有违反状态，则 SCRIPT_TYPE 设置为“VIOLATED”。
如果监视为正常状态，则 SCRIPT_TYPE 设置为 “NORMAL” 。您可以使用这一变量确定在违反监视时
和监视正常时该执行什么操作。
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
## 例如，撰写一个可用于发送邮件的消息。
## 所撰写消息还可用于任何其他目的。
message()

{echo "监视状态通知"
        echo ""
        echo "监视状态为 $1"
        echo ""
echo "日期:                         $DATE"
echo "时间:                         $TIME"
echo "模型类型名称:                 $MTYPE_NAME"
echo "模型句柄:                     $MODEL_HANDLE"
echo "模型名称:           $MODEL_NAME"
echo "实例:                         $INSTANCE"
echo "警报 ID:            $ALARM_ID"
echo "条件:               $CONDITION"
echo "原因代码:           $CAUSE_CODE"
echo "修复人员:                     $REPAIR_PERSON"
```

```
echo "警报状态:                    $ALARM_STATUS"
echo "监视名称:            $WATCH_NAME"
echo "监视创建者:          $WATCH_CREATOR"
echo "监视表达式: $WATCH_SRC"
echo "监视表达式值:        $WATCH_SRC_VAL"
echo "监视参考:            $WATCH_REF"
echo "监视参考值: $WATCH_REF_VAL"
echo "监视重置:            $WATCH_RES"
echo "监视重置值: $WATCH_RES_VAL"}
## 通过邮件把所撰写消息发送给相关用户。
## <USER LIST> 是邮件收件人的列表。
message "$SCRIPT_TYPE" | mail <USER LIST>
```

激活或停用监视

监视在默认情况下处于活动还是非活动状态，取决于它的创建方式。

您可添加到系统的监视的数目或形式没有任何限制。这种灵活性也意味着，使用不当会耗尽系统资源，并导致性能降级。即使单个监视消耗最少的平台资源且生成较少的网络通信，我们还是建议停用或删除不必要的监视。此外，在设置引用大型设备属性列表的监视时请保持谨慎。

要为单个模型激活或停用监视，请从“监视”表中选择监视，然后单击  （激活监视）或  （停用监视）。您还可以选择要一次激活或停用的多个监视。

您可以使用属性编辑器，在多个模型上激活或停用多个监视。当您选择某个监视属性，并将它移动到属性编辑器的右侧面板时，将启用“激活监视”和“停用监视”选项。

注意：有关属性编辑器的详细信息，请参阅*《建模并管理您的 IT 基础架构管理员指南》*。

从命令行激活和停用监视

除 OneClick 控制台之外，您还可以从命令行激活或停用监视并查看数据配置文件。

以下命令列出模型类型的可用监视数据：

```
show watch [mh=模型句柄] [lh=格局句柄]
```

以下命令更新监视状态：

```
update action=操作代码 [watch=监视 ID] [mh=模型句柄]
```

注意：[watch=监视 ID] 参数仅适用于激活 (0x00480003) 和停用 (0x00480004) 操作。

编辑监视

您可以编辑监视，以更改其表达式、属性和阈值。

遵循这些步骤:

1. 单击模型的“信息”选项卡，并展开“阈值和监视”子视图。
2. 展开“监视”子视图。
3. 在“监视”表中选择监视，然后单击  （编辑）。

   将在“编辑监视”对话框中打开该监视。
4. 根据需要更改监视表达式、属性和阈值，然后单击“确定”。

   注意：有关监视表达式、属性和阈值的详细信息，请参阅创建监视 (p. 9)。

   监视即会编辑。

可继承监视编辑

当您编辑可继承监视时，默认情况下修改所选模型类型的监视定义。您可以为监视选择不同的父模型类型。使用“属性”选项卡的“模型类型”部分中的“浏览”按钮。

如果修改从监视的原始模型类型派生的模型类型的监视定义，则这些更改仅对该类型及其所派生类型的模型生效。发起监视的模型类型不会更改。

例如，最初在 Gen_IF_Port 上创建了某个监视并且该监视可继承。Serial_IF_Port 是从 Gen_IF_Port 派生的，因此 Serial_IF_Port 模型继承该监视。如果在 Gen_IF_Port 模型类型上修改了监视定义，则这些更改将传播到 Serial_IF_Port 模型。然而，如果在 Serial_IF_Port 上修改了监视定义，则新版本覆盖 Gen_IF_Port 上的定义。仅影响 Serial_IF_Port 模型，而 Gen_IF_Port 模型保持原始监视定义。由于存在可继承规则，因此您能够以不同方式为派生的模型类型重新定义或覆盖可继承监视。

复制监视

使用“复制”功能可创建彼此仅有很少区别的多个监视。

遵循这些步骤:

1. 在“监视”表中选择监视，然后单击  （复制监视）。

   此时将打开监视。

2. 为新监视输入名称，根据需要更改任何其他信息，然后单击“确定”。

   监视即会复制并创建一个新监视。

删除监视

如果您不再需要关联的阈值监控，则可以删除监视。

遵循这些步骤:

1. 在“监视”表中选择监视，然后单击  （删除）。

   如果该监视存在于多个格局中，则会打开一个对话框。左侧显示该监视存在于其中并将从中删除该监视的格局。

2. （可选）要在某个格局上保留该监视，请将该格局移动到右侧。
3. 单击“确定”。

   该监视即会从左侧列出的格局中删除。

显示监视信息

您可以显示监视信息，以查看关于监视的报告。您还可以打印该信息，以及将该信息导出到文本文件或 HTML 文件。

遵循这些步骤:

1. 在“监视”表中选择监视，然后单击  （信息）。

   此时“监视报告”对话框中将显示监视信息。

2. （可选）单击“打印”以打印监视信息。
3. （可选）单击“导出”以将信息保存到文本或 HTML 文件。

   该监视即会打印或导出。

生成关于多重监视的报告

您可以从 OneClick 管理页面生成关于多重监视的报告。

遵循这些步骤:

1. 在浏览器中打开 OneClick 主页，然后单击“管理”选项卡。

   此时将打开“管理页面”。

2. 单击“监视报告”链接。

   此时将打开“生成监视报告”对话框。其中将显示“选择格局”列表和“选择模型类型”列表。

3. 为您希望生成报告的监视选择格局和模型类型，然后单击“生成报告”。

   报告即会生成。

第 2 章： 事件管理

此部分包含以下主题：

事件

为了说明 CA Spectrum 高级事件处理，请考虑以下方案。出于测试目的，您需要暂时允许高子级计数。您希望仅当高子级计数条件持续存在超过一分钟时生成警报，并且在重置监视时清除所有此类警报。使用 EventPair 事件规则，CA Spectrum 可以执行以下任务：

- 违反 ChildLimit 监视时，生成初始事件。
- 阻止警报，除非在发生初始事件后 60 秒内*未*发生重置事件。
- 当重置或停用此监视的所有实例时，生成另一个事件。

注意：有关事件规则和创建事件的详细信息，请参阅*《事件配置用户指南》*。

配置事件

以下过程是一个事件配置示例。利用前一个主题中的示例构建，第一个事件是“初始子级计数太高”(0xffffffff)，第二个事件是“重置”(0xfffffffe)。如果违反子级计数阈值，且在 60 秒内未重置该阈值，则生成“子级计数太高”警报（事件/警报 0xfffffffd）。

遵循这些步骤:

1. 从“工具”菜单中依次选择“实用工具”、“事件配置”。

   此时将显示“事件配置”页面，其中包括“导航”、“内容”和“详细信息”面板。

2. 在“导航”面板中单击  （创建事件）。

   此时将显示“创建事件”对话框，“事件代码”字段中填充有默认值。

3. （可选）编辑事件代码并输入事件消息，然后单击“确定”。

   将会创建事件，其详细信息显示在右侧的“内容”和“详细信息”面板中。

4. 创建并保存初始子级计数太高事件 (0xffffffff)。

   此事件触发有条件生成警报和重置事件 (0xfffffffe) 的 EventPair 规则。监视使用重置事件评估子级计数条件的持久性，并在重置监视时清除 EventPair 规则生成的所有警报。

5. 如果子级计数居高不下，则创建并保存事件“0xfffffffd”和用于生成通知的相应警报。

6. 为初始“子级计数太高”事件 (0xffffffff)（在 60 秒内未接收到重置事件 (0xfffffffe) 时生成“0xfffffffd”事件和警报）创建并保存事件对规则。

   重要说明！如果在使用事件规则处理事件的同时更新 SpectroSERVER，则无法完成处理。调用此功能时正在处理的所有事件规则都将被刷新。

7. 依次单击“文件”、“全部保存”。

   将保存新事件。

8. 打开监视以进行编辑。

9. 在“阈值”选项卡上，在“通知”下选择“生成事件”。

   此时将显示事件代码字段。

10. 输入您之前在“当阈值被违反时”字段中创建的初始“子级计数太高”事件 (0xffffffff)。

11. 输入您之前在“当阈值被重置时”字段中创建的警报清除“重置”事件 (0xfffffffe)，然后单击“确定”。

    事件将与监视关联。

    如果 LAN_802_3 容器包含 5 个或更多个模型，则会生成初始“子级计数太高”事件，并评估事件规则。仅当在发生初始事件的 60 秒内未接收到重置事件时，才生成警报。如果重置监视，则会清除监视生成的任何“子级计数太高”警报。

此示例的 EventDisp 条目（如果需要）如下所示：

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

# generate an alarm on persistent child count too high0xfffffffd E 50 A
1,0xfffffffd

# if child count drops below threshold, clear any existing alarms0xfffffffe E 50
C 0xfffffffd
```

事件代码

对于每个事件，在 CA Spectrum 事件日志中都会生成事件代码。

注意：创建监视时不生成事件。

以下列表包含事件代码及其说明：

0x0048000a

无法为模型激活监视

0x00480000

监视修改

0x00480001

监视销毁

0x00480002

监视激活

0x00480003

监视停用

0x00480004

监视阈值存在违反

0x00480005

监视的阈值状态被重置（利用变量重置值）

0x00480006

监视实例存在违反

0x00480007

监视实例的阈值状态被重置（利用变量重置值）

0x00480010

可能的错误原因消息

0x00480012

记录间隔冲突

0x00480013

执行脚本时出错（仅限阈值监视）

0x00480014

重置模型的监视阈值，并说明原因

事件格式文件

您可以使用现有事件格式文件 Event00480004 和 Event00480005（位于 <*$SPECROOT*>/SG-Support/CsEvFormat 中）作为模板来创建自己的监视相关的事件格式文件。

注意：有关详细信息，请参阅 *《事件配置用户指南》*。

事件变量

与监视阈值违反 (0x00480004) 和重置 (0x00480005) 相关的事件都存在变量绑定。这对直接生成警报的监视和生成事件的监视都适用。清除事件只有一个变量绑定，即包含监视名称的 ID #2。

下表显示绑定变量及其内容：

变量绑定 ID	内容
2	监视名称
4	阈值参考值
5	阈值重置值（仅限重置事件）
6	比较字符串值
7	计算的监视值
8	监视具有实例时的实例对象标识符 (OID)，否则为空
9	内部；用于完成相应事件消息的字符串
10	内部；用于完成相应事件消息的字符串
11	内部；用于完成相应事件消息的字符串

第 3 章： 监视表达式公式规则

此部分包含以下主题：

监视表达式

监视表达式定义监视监控的内容。监视表达式可以像属性名称一样简单，也可以像公式一样复杂。监视表达式包括通过数学符号和布尔运算符（称为基元）彼此相关的属性和常量值。

表达式按照基元的优先级顺序从左到右进行计算：

优先级	基元
1	.# .
2	( )
3	DELTA ( ) COUNTER_DELTA ( )
4	!
5	| &
6	* /
7	+ -
8	= = != >= <= < >

详细信息：

属性 (p. 35)

基元

基元通常在表达式中使用，有左侧元素和右侧元素。例如，在 A+B 中，+ 就是定义左侧的 A 与右侧的 B 之间的关系的基元。括号用于指示具有多个组成部分的表达式的计算顺序。

可用基元如下所示：

- +
- -
- *
- /
- ==
- !=
- >
- <
- >=
- <=
- ！(not)
- & (and)
- ,
- |
- (
-)
- TRUE
- FALSE
- TIME
- DELTA
- MIN
- MAX
- INTEGER
- REAL
- UNSIGNED
- COUNTER_DELTA
- UNSIGNED64

您可以按照功能和适用性分组基元。只能在文本字符串之间或数值表达式中应用以下基元：

- = = 等于
- >= 大于或等于
- <= 小于或等于
- != 不等于
- > 大于
- < 小于

注意：这些基元不支持属性类型对象 ID、IP 地址和布尔值。

您只能为数值表达式应用以下基元：

- + 加
- - 减
- * 乘
- / 除

这些基元不支持以下属性类型：文本字符串、对象 ID、IP 地址和布尔值。支持的类型按以下顺序进行计算：

- 实数
- CPU 时钟周期、日期、标尺、计数器
- 枚举、整数

如果元素分别属于两个不同级别，则在计算时，将其视为属于较高级别。例如，将 5 + 5 作为整数计算，而将 5 + 5.1 作为实数计算。

注意：CA Spectrum 模型类型使用计数器这一属性类型存储无符号长整数。如果在表达式中非法使用无符号基元，例如，“文本字符串 + UNSIGNED(5)”，则生成计数器类型的错误消息。

对于布尔表达式，只能应用以下基元：

- ！逻辑否
- & 与
- | 或

下列基元由单词而非符号组成：

TIME

将当前时间表示为自格林尼治标准时间 1970 年 1 月 1 日 00:00 以来的秒数。

DELTA

计算一个属性在采样频率间隔的变化。例如，以 30 秒为间隔对名为 int1 的整数属性的监视可能生成的 DELTA 值如下所示：

间隔	值
0	100, 0
30	1000 900
60	1000 0
90	-1000 -2000
120	2000 3000

DELTA 基元支持下列属性类型：

- 整数
- 枚举
- 实数
- 短整型
- CPU 时钟周期
- 日期
- 标尺
- 计数器

在预期值永远不会变小（如使用 CPU 时钟周期和计数器）的情况下，使用 COUNTER_DELTA。

您可以按以下方式实例化 DELTA 表达式中的属性：

- DELTA (If_In_Octets.2)
- DELTA (If_Out_Octets.#)
- DELTA (TIME) 也受支持

注意：DELTA (TIME) 是监视的当前和上次评估之间的差值；它仅用于涉及内部属性的计算。要生成涉及外部属性的比率计算，请使用设备的时间，例如，DELTA (XflDELTA (Sys_Up_Time)，其中 Sys_Up_Time 是设备上的时间计数器。

COUNTER_DELTA

计算假定其值为递增无符号整数的属性的变化。该值可能重置为零，但是此基元的结果始终是无符号正整数。负属性值将被视为大型无符号值。

COUNTER_DELTA 基元与 DELTA 支持相同的属性类型，您还可以根据对 DELTA 的描述在 COUNTER_DELTA 表达式中实例化属性。
COUNTER_DELTA (TIME) 也受支持。

ATTR

指示基元后面括号中的十六进制数是属性标识符。此基元允许在存在重复名称时唯一标识属性，如以下示例中所示：

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

<*attr_id*> 值是“0x”或“0X”开头，后跟一到八之间的十六进制数。

TRUE

定义布尔常量 True。

FALSE

定义布尔常量 False。

MAX

标识用括号括住且由逗号分隔的两个表达式（例如，(attribute x+1, attribute y-5)）中的较大者。该表达式可能包括产生数字值（不是文本、字符串、八进制或布尔值）的操作数、属性或基元的任何组合。

MIN

标识括号中由逗号分隔的两个表达式（例如，(attribute x+1, attribute y-5)）中的较小者。该表达式可能包括产生数字值（不是文本、字符串、八进制或布尔值）的操作数、属性或基元的组合。

您可以使用以下基元（弹出选择器菜单上不显示）来指示由监视监控的列表属性的实例。

.

指示以下条目标识列表属性的特定实例。

例如，If_In_Octets.2 指定属性 If_In_Octets 的第二个实例。如果是 IP 地址表，则标识符可能是整个地址，而不是单个数字。

如果属性的指定实例不存在（即使对于激活监视的模型之一），则监视会失败。如果该实例对于某些模型存在，而对于其他模型不存在，则实例存在的那些模型的监视将成功，而其他模型的监视将失败。

.#

指示监视详细信息视图中显示的当前实例说明符值确定了监控哪些实例。

例如，如果实例说明符值为 ALL，则表达式 If_In_Octets.# 将监视应用于 If_In_Octets 属性的全部实例。然而，如果实例说明符值为 RANGE (1-3)，则仅监控实例 1、2 和 3。

如果指定了实例范围，则即使该范围内的实例之一对于其中一个模型不存在，监视也会失败。反之，对于该范围内的所有实例都存在的任何模型，监视将成功。

数据类型

下表显示您可以分配给各种属性类型的数据类型值。

注意：虽然 CA Spectrum 不支持监视数据类型为八位字节字符串的属性，但是您可以使用文本字符串数据类型监视这类属性。

表达式结果类型	可接受的目标属性类型
BOOLEAN	任意数字类型
TEXT STRING	文本字符串
INTEGER	整数、枚举、实数
ENUMERATION	整数、枚举、实数
REAL	整数、枚举、实数
DATE	实数、日期、CPU 时钟周期、计数器、标尺、计数器 64
TIME TICKS	实数、日期、CPU 时钟周期、计数器、标尺、计数器 64

表达式结果类型	可接受的目标属性类型
COUNTER	实数、日期、CPU 时钟周期、计数器、标尺、计数器 64
GAUGE	实数、日期、CPU 时钟周期、计数器、标尺、计数器 64
OBJECT ID	对象 ID
IP ADDRESS	IP 地址
COUNTER64	计数器 64、实数

例如，如果指定了一个名为 Int1 的整数属性，则允许表达式 Int1 = 50.5（需要转换），但不允许 Int1 = "一个字符串"。

这些规则在将值分配给布尔值和文本字符串属性方面最为灵活。可以将任何表达式计算为 0 和 1，并可以将其写入布尔值属性。例如，如果指定一个名为 Bool1 的布尔值属性，则表达式 Bool1 = 500 * 50 + 450 将 TRUE 赋予 Bool1。表达式 Bool1 = (500 * 50 + 450) * 0 将 FALSE 赋予 Bool1。同样，可以将任何表达式计算为文本字符串。如果指定一个名为 str1 的文本字符串属性，则表达式 str1 = 500 + 50 将字符串“550”赋予 str1。

注意：文本字符串属性类型接受引号中的文本字符串和未包含在引号中的算术表达式（如 500 + 50，产生字符串“550”）。

常量

与属性值和运算符一样，您可以直接将常量输入到表达式字段中。您可以在表达式公式中输入以下类型的常量：

- 无符号整数，由具有正值的一位或多位数字的序列表示。
- 实数，由零位或多位数字的序列后跟一个点，再后跟一位或多位数字的序列表示。例如，.7、1.7 或 23.24（但不能是 7）。
- 有符号整数，由值为正或负的一位或多位数字的序列表示。
- 文本字符串，由双引号括住的任何字符序列表示。例如，"一个字符串"、"5.25" 或 " "。

转换运算符

转换强制一种数据类型成为另一种数据类型。执行转换时，您面临着丢失被转换的值的某些部分的风险。当源数据类型的范围适合目标数据类型的范围时，没有必要进行转换。

例如，您可以在不转换的情况下将整数分配给实数，因为 -1.79769 e+308 <= -2,147,483,648，2,147,483,647 <= 1.79769 e+308。然而，当将整数赋值给计数器时，必须使用转换，因为范围不重叠。

对于相关数据类型的重叠范围，转换可正确工作。例如，将整数 5 转换为计数器与预期的行为相同。但是，负数并不在 Counter 的范围内。因此，将 -5 转换为 Counter 将产生无符号（正）结果 4,294,967,291，因为计算机以这种方式表示数字。反之，将大于 2,147,483,647 的计数器值转换为整数将产生负数。

您可以使用以下运算符将括号中表达式的结果转换为所选数据类型。

运算符	转换结果
UNSIGNED	通常，将 27 视为有符号整数，而将“UNSIGNED (27)”视为无符号整数 27。如果在表达式中有负数出现该运算符之后，则将其视为无符号。示例：将 UNSIGNED (-5) 视为 5。数字 -5 为 UNSIGNED，值为：4294967291。
INTEGER	在表达式中跟在此运算符后面的任何实数都将舍入为最接近的整数。示例：将 INTEGER (2.4) 视为 2，将 INTEGER (2.6) 视为 3。
REAL	示例：将 REAL (3) 视为 3.0。

文字数值的数据类型

监视表达式中的数值文字与一系列范围进行比较，以确定其数据类型。下表中的“类型”列是数值文字的实际数据类型，这些数据类型可直接与该列单元格中列出的其他类型相互交换。

下表显示确定文字数值的数据类型的方式：

最小值	最大值	类型
-2,147,483,648	2,147,483,647	整数、布尔值*、枚举
0	4,294,967,295	计数器、日期**、标尺、CPU 时钟周期

0	18,446,744,073,709,551,615	计数器 64
-1.79769 e+308	1.79769 e+308	实数

* 您还可以将常量 TRUE 和 FALSE 用于布尔值数据类型。任何非零值都相当于 TRUE。

**自 1970 年 1 月 1 日 00:00:00 UTC(0) 以来的秒数。

属性

您在监视表达式公式中输入的属性必须始终以字母开头，并包括字母、数字和下划线字符 (_) 的组合。用单引号括住名称与基元（TRUE、FALSE、TIME、DELTA、COUNTER_DELTA）相同的属性。您还可以使用 ATTR 基元，依据属性 ID 指定属性。

监视表达式 (p. 27)可以包括单个属性。要创建多属性表达式，请从按钮调色板中选择运算符，然后选择另一个属性，以此类推。

如果在表达式中指定列表属性，则必须包括实例信息。要指定实例，请附加“.”，后跟实例 ID 号（例如，**ifInOctets.2**）。或者，附加“.#”（例如，ifInOctets.#），并在“实例”字段中选择“全部”或“范围”。对于“范围”，请在“从”和“到”字段中指定低 ID 和高 ID。在“属性选择器”对话框的“类型”列中，用“[]”指示列表属性。

除了作为监视表达式的一部分输入的属性之外，您还可以动态地创建在存储监视信息和目标属性时需要的属性。

实例标识符

当您在监视表达式公式中使用列表属性时，可以使用实例标识符指定特定对象。

列表属性需要通过实例 ID 来充分标识一个对象。您可以在监视表达式中指定属性的一个特定实例、一系列实例或所有可能的实例。在构建 OID 时，它最多使用两个不同的子 OID：标准 SpectroSERVER OID 机制或监视实例说明符。SpectroSERVER 利用 OID 前缀和 OID 参考（可选）创建 OID。当您为“表达式”页上的监视表达式指定实例时，就已将其附加到 OID 中，如下所示：

```
OID = OID 前缀 + OID 参考 + 监视实例
```

如果您使用实例说明符指定范围，则计算该范围内每个实例的结果。如果所指定范围大于实际实例，则仅计算实际实例的结果。也就是说，如果您指定的范围是 1-10，且仅存在 3 个实例，则仅计算 3 个结果。

例如，您希望对路由器的 If_In_Octets 设置监视。您将范围 1-3 提供为实例说明符。因为 CA Spectrum 数据库用于 If_In_Octets 的 OID 前缀是 1.3.6.1.2.1.2.2.1.10，所以分配下列 OID：

- 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

如果您将 ALL 用作实例说明符，则动态确定 If_In_Octets 对象的所有实例。您还可以在监视表达式中指定实例，或者您可以将对象的特定实例与 RANGE 或 ALL 实例说明符混合使用。使用 Instance_of 基元（“.”）指定某个对象的实例将覆盖该对象的实例说明符。

例如，在以下公式中，使用 Instance_of 基元（后跟值）指定 If_In_Errors 对象的实例数 3，.# 基元表明应将当前实例说明符用于 If_In_Octets 对象。

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

假定当前的实例说明符为 ALL，而且 If_In_Errors 和 If_In_Octets 都是列表属性，其 OID 分别是 1.3.6.1.2.1.2.2.1.14 和 1.3.6.1.2.1.2.2.1.10。使用以下 OID，为每个实例执行除法运算：

```
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
```

请注意 Instance_of 基元如何覆盖 If_In_Errors 属性 OID 的实例说明符。Instance_of 基元允许指定特定实例，而不是每个属性使用每个可能的实例作为选择 ALL 的结果。但是，实例说明符仍然确定着特定监视的实例数。

模型的 INSTANCE_ID 和实例说明符应用于监视表达式中的每个列表属性。如果监视表达式包含使用不同实例的属性，则当您尝试添加监视时，将返回错误。

例如，如果您输入一个包含将主板编号用作实例的属性的公式，和另一个将主板端口用作实例的公式，则返回错误。

监视定义属性

当您为模型创建监视时，就创建了一个用于存储监视详细信息的属性。该属性具有计算机生成的名称。它是使用模型所属模型类型下的监视定义组中的活动数据库开发者 ID（已注册或默认值）创建的。

监视目标属性

当您为模型创建监视时，就同时创建了一个与该监视具有相同名称和数据类型的目标属性。该属性类似于任何其他模型类型属性。读取它时，会自动计算监视表达式。计算的结果就是读取操作的结果。

阈值参考和重置兼容性

对于阈值监视，阈值和重置表达式必须都属于与监视表达式兼容的类型。

下表显示结果类型和相应的可接受类型：

结果类型	可接受类型
BOOLEAN	布尔型
TEXT STRING	文本字符串
INTEGER	整数、枚举、实数
ENUMERATION	整数、枚举、实数
REAL	实数
DATE	实数、日期、CPU 时钟周期、计数器、标尺、计数器 64
TIME TICKS	实数、日期、CPU 时钟周期、计数器、标尺、计数器 64
COUNTER	实数、日期、CPU 时钟周期、计数器、标尺、计数器 64
GAUGE	实数、日期、CPU 时钟周期、计数器、标尺、计数器 64
OBJECT ID	对象 ID
IP ADDRESS	IP 地址
COUNTER64	计数器 64、实数

附录 A： 监视类型示例

此部分包含以下主题：

第一个监视方案

下列参数在 Hub_CSI_IRBM 上建立监视。此监视检查传输的总帧数和每 60 秒接收的冲突数。如果超过阈值 1000000，它会生成次要警报并显示消息“冲突太多”。阈值重置值指示，当总冲突数回落到 500,000 时，阈值状态将从“已违反”重置为“正常”（并清除该警报）。

您可以按以下方式为此方案创建监视：

1. 创建类型为“计数器”的非活动监视，并将其命名为 HubColls_v。在表达式中分配总冲突数 1000000。
2. 创建类型为“计数器”的非活动监视，并将其命名为 HubColls_r。在表达式中分配总冲突数 500000。
3. 创建类型为“计数器”的活动监视，并将其命名为 HubColls，然后按照下列监视示例中所示为其赋值。

示例：监视

第一个监视包括以下参数：

- 名称：HubColls_v
- 数据类型：计数器

- 表达式：
  - 表达式：1000000
- 实例：无
- 属性
  - 默认激活：非活动
- 阈值：无

第二个监视包括以下参数：

- 名称：HubColls_r
- 数据类型：计数器
- 表达式：
  - 表达式：500000
- 实例：无
- 属性
  - 默认激活：非活动
- 阈值：无

第三个监视包括以下参数：

- 名称：HubColls
- 数据类型：计数器
- 表达式
  - 表达式：Hub_Trans_Coll+Hub_Rec_Colls
- 实例：无
- 属性
  - 默认激活：活动
  - 评估：通过轮询
  - 轮询时间间隔：00:01:00
- 阈值
  - 附加阈值（已选中）- 违反阈值，如果值：> HubColls_v（创建名为 HubColls_v 的非活动计数器监视，表达式为 1000000）。
  - 重置阈值，如果值：HubColls_r（创建名为 HubColls_r 的非活动计数器监视，表达式为 500000）。

- 在创建 HubColls 监视之前，必须先创建属性 HubColls_v 和 HubColls_r。
- 生成警报（已选中）- 重要级别：次要警报，说明：CollsExceeded（创建名为 CollsExceeded 的新警报并显示消息“冲突太多”）。

第二个监视方案

网络管理员希望监控服务器上的磁盘利用率级别，但是希望从监控中排除 CD-ROM 驱动器。如果 CD 不存在，则 CD-ROM 驱动器通常显示 0% 的利用率；但是如果驱动器中存在充满内容的 CD，则其显示近 100% 的利用率。然而，这类信息对管理员无用。

为了检查磁盘利用率，管理员希望使用 RFC2790App，其返回设备类型作为 OID。每个 OID 映射到不同的设备类型。必须对照此 MIB 中的存储类型进行检查的任何表达式都必须使用其 OID 进行比较。（您可以比较字符串与 OID）。

下表显示的是 RFC2790App 设备类型与 OID 的映射。

设备类型	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

管理员必须在 RFC2790App 模型上创建两个监视。第一个监视设置一个在第二个监视中使用的布尔值，以便从监控中排除 CD-ROM 驱动器（OID 1.3.6.1.2.1.25.2.1.7）。第二个监视监控主机上其他类型存储设备的磁盘利用率（以百分比表示）。在此示例中，如果利用率超过 90%，则发送警报。

示例：监视

第一个监视包括以下参数：

- 名称：isNotCDROM
- 数据类型：布尔值
- 表达式：表达式：hrStorageType.# != 1.3.6.1.2.1.25.2.1.7
- 实例：全部
- 属性
  - 默认激活：活动
  - 评估：根据要求
- 阈值：无

第二个监视包括以下参数：

- 名称：PctDiskUsed
- 数据类型：实数
- 表达式
  - 表达式：REAL(hrStorageUsed.#)/REAL(hrStorageSize.#) * 100 * REAL(isNotCDROM.#)
  - 实例：全部
- 属性
  - 默认激活：活动
  - 评估：通过轮询
  - 轮询时间间隔：30 秒
- 阈值
  - 违反阈值，如果值：>= <*设置为可接受的磁盘利用率水平*>
- 警报
  - 警报重要级别：次要
  - 警报说明：DiskUtilAlarm
  - 用户可清除警报
  - 用户清除警报时不重置监视。
- 脚本：无

更改时进行评估监视

下列监视是更改时进行评估 (EoC) 监视的示例。这些监视确定视图是何时编辑的。因此，管理员通过运行一个搜索，即可准确显示 LAN 是何时编辑的，以及新模型是何时添加到这些 LAN 中的。发生更改时会评估两个属性，这两个属性不可能频繁地更改。

示例：更改时进行评估监视

第一个监视包括以下参数：

- 名称：Child_Count_Watch
- 数据类型：整数
- 表达式
  - 表达式：Child_Count
  - 实例：无
- 属性
  - 默认激活：活动
  - 评估：通过 EoC
  - 轮询时间间隔：无
- 阈值：无

第二个监视包括以下参数：

- 名称：Edit_Count
- 数据类型：整数
- 表达式
  - 表达式：EDIT_COUNT
  - 实例：无
- 属性
  - 默认激活：活动
  - 评估：通过 EoC
- 阈值：无

轮询阈值监视

在下列示例中，创建轮询阈值监视，以便每当 LAN_802_3 中的子级数超过 5 时生成次要警报。当子级计数降到低于 5 时，监视会清除警报。

首先，创建属性监视 ChildLimit_v（任意名称），并为其提供表达式“5”。创建监视详细描绘了此监视的创建过程。然后创建名为 Child_Limit 的轮询阈值监视。

示例：轮询阈值监视

第一个监视包括以下参数：

- Name: ChildLimit_v
- 数据类型：计数器
- 表达式：5
- 属性：无
- 阈值：无

第二个监视包括以下参数：

- 名称：ChildLimit
- 数据类型：整数
- 表达式
  - 表达式：Child_Count
- 实例：无
- 属性
  - 默认激活：活动
  - 评估：通过轮询
  - 轮询时间间隔：00:01:00
- 阈值
  - 附加阈值（已选中）
  - 违反阈值，如果值：> ChildLimit_v
  - 重置阈值，如果值：<= ChildLimit_v
  - 生成警报（已选中）
  - 重要级别：次要
  - 警报说明：ExceedChildLimit（创建名为 ExceedChildLimit 的新警报并显示消息“只允许 5 个子级”）。

第一个按需监视方案

只要从按需监视请求信息，就会评估按需监视。

以下简单按需监视的示例与轮询阈值监视（当 Cisco 路由器（模型类型：Rtr_CiscoIGS）即将耗尽内存时生成警报）一起使用。属性 freeMem 少于 1500，当其超过 2500 时进行清除。

示例：与轮询阈值监视一起使用的简单按需监视

第一个监视包括以下参数：

- 名称：MemLow
- 数据类型：整数
- 表达式
  - 表达式：1500
  - 实例：无
- 属性
  - 默认激活：非活动
  - 评估：根据要求
- 阈值：无

第二个监视包括以下参数：

- 名称：MemHigh
- 数据类型：整数
- 表达式
  - 表达式：2500
  - 实例：无
- 属性
  - 默认激活：非活动
  - 评估：根据要求
- 阈值：无

第三个监视包括以下参数：

- 名称：Mem_Good
- 数据类型：整数

- 表达式
  - 表达式：freeMem
  - 实例：无
- 属性
  - 默认激活：活动
  - 评估：通过轮询
  - 轮询时间间隔：1 分钟
- 阈值
  - 违反阈值，如果值：<= MemLow
  - 重置阈值，如果值：> MemHigh
- 警报
  - 警报重要级别：主要
  - 警报说明：Rtr+Memory
- 脚本：无

第二个按需监视方案

只要从按需监视请求信息，就会评估按需监视。

以下监视旨在向网络管理员显示随着 Cisco 路由器（模型类型：Rtr_ Cisco）性能的逐渐降级而变化的彩色警报条件。次要警报的值为 50 - 59；主要警报的值为 60 - 69，关键警报的值超过 70。

示例：按需监视

第一个监视包括以下参数：

- 名称：True_Ref
- 数据类型：布尔值
- 表达式
  - 表达式：1
  - 实例：无

- 属性
  - 默认激活：非活动
  - 评估：根据要求
- 阈值：无

第二个监视包括以下参数：

- 名称：Minor_Busy
- 数据类型：布尔值
- 表达式
  - 表达式：((busyPer >= 50) & (busyPer <60))
  - 实例：无
- 属性
  - 默认激活：活动
  - 评估：通过轮询
  - 轮询时间间隔：1 分钟
- 阈值
  - 违反阈值，如果值：> True_Ref
  - 重置阈值，如果值：<= True_Ref
- 警报：
  - 警报重要级别：次要
  - 警报说明：RtrBusy1
  - 脚本：无

第三个监视包括以下参数：

- 名称：Major_Busy
- 数据类型：布尔值
- 表达式
  - 表达式：((busyPer >= 60) & (busyPer < 70))
  - 实例：无
- 属性
  - 默认激活：活动
  - 评估：通过轮询
  - 轮询时间间隔：1 分钟

- 阈值
  - 阈值：违反阈值，如果值：== True_Ref
  - 重置阈值，如果值：!= True_Ref
- 警报
  - 警报重要级别：主要
  - 警报说明：RtrBusy 2
  - 脚本：无

第四个监视包括以下参数：

- 名称：Critical_Busy
- 数据类型：布尔值
- 表达式
  - 表达式：(busyPer >= 70)
  - 实例：无
- 属性
  - 默认激活：活动
  - 评估：通过轮询
  - 轮询时间间隔：60
- 阈值
  - 违反阈值，如果值：== True_Ref。
  - 重置阈值，如果值：！= True_Ref。
- 警报：
  - 警报重要级别：关键
  - 警报说明：RtrBusy3
  - 脚本：无

第三个按需监视方案

只要从按需监视请求信息，就会评估按需监视。

在此方案中，网络管理员希望在帧中继链接失败（导致激活拨号备份线路）时显示警报。拨号备份是 ISDN 接口类型 21（基本 ISDN 服务）。下列监视验证是否在任何 ISDN 端口上接收通信。如果通信正在流动，则将次要警报置于 Cisco 路由器（模型类型：Rtr_Cisco）上。

示例：按需监视

第一个监视包括以下参数：

- 名称：True_Ref
- 数据类型：布尔值
- 表达式
  - 表达式：1
  - 实例：无
- 属性
  - 默认激活：非活动
  - 评估：根据要求
- 阈值：无

第二个监视包括以下参数：

- 名称：ISDN_Backup
- 数据类型：布尔值
- 表达式
  - 表达式：((COUNTER_DELTA (ifInOctets.#) > 0) & (ifType.# == 21))
  - 实例：全部
- 属性
  - 默认激活：活动
  - 评估：通过轮询
  - 轮询时间间隔：1 分钟
- 阈值
  - 违反阈值，如果值：== True_Ref
  - 重置阈值，如果值：!= True_Ref
- 警报：
  - 警报重要级别：次要
  - 警报说明：Backup_Active
- 脚本：无

第四个按需监视方案

只要从按需监视请求信息，就会评估按需监视。

在此方案中，监视监控 ISDN 接口模型上的 ifOutOctets 值，以确定其是否增大或者是否为非零。此值指示接口是否处于活动状态。您可以修改轮询、日志记录、警报重要级别和警报文本以满足您的需求。第二个监视使用 true 或 false (1/0) 指示符：1 指示该接口正在发送 ifOutOctets，0 指示该接口未在发送。

示例：按需监视

第一个监视包括以下参数：

- 名称：isISDN
- 数据类型：布尔值
- 表达式
  - 表达式：(X_ifType == 21)
  - 实例：无
- 属性
  - 默认激活：活动
  - 评估：根据要求
  - 可继承：False
- 阈值
  - 阈值：无

第二个监视包括以下参数：

- 名称：ISDN_Up
- 数据类型：整数
- 表达式
  - 表达式：MAX(0, MIN(1, INTEGER(((COUNTER_DELTA (X_OutOctets) * INTEGER(isISDN)) > 0))))
  - 实例：无
- 属性
  - 默认激活：活动
  - 评估：通过轮询

  - 轮询时间间隔：0+00:05:00
  - 可继承：False
- 阈值：
  - 违反阈值，如果值：== 1
  - 重置阈值，如果值：!= 1
- 警报
  - 警报重要级别：次要
  - 警报说明：ErrorTholdAlarm
  - 用户可清除警报
  - 用户清除警报时不重置监视。
- 脚本：无

第一个可用性和测试监视方案

这里描述的示例监视用于可用性和测试。

在此方案中，创建了两个监视，一个监视创建名为 WatchLoad_v 的属性且表达式值为 .75，另一个监视使用 CPULoadRate 属性。在以下示例中，创建了一个名为 WatchLoad_Alarm 的警报说明，内容类似于“CPU 负载已超过 75%”。

示例：可用性和测试监视

第一个监视包括以下参数：

- Name: WatchLoad_v
- 数据类型：实数
- 表达式
  - 表达式：.75
  - 实例：无
- 属性
  - 默认激活：非活动
  - 评估：根据要求
- 阈值：无

第二个监视包括以下参数：

- 名称：CPU_Load
- 数据类型：实数
- 表达式
  - 表达式：CPULoadRate
  - 实例：无
- 属性
  - 默认激活：活动
  - 评估：通过轮询
  - 轮询时间间隔：1 分钟
- 阈值
  - 违反阈值，如果值：> WatchLoad_v
  - 重置阈值，如果值：< = WatchLoad_v
- 警报
  - 警报重要级别：次要
  - 警报说明：WatchLoad_Alarm
  - 脚本：无

第二个可用性和测试监视方案

这里描述的示例监视用于可用性和测试。

在此示例中，您创建一个 EoC 监视，以便当容器模型的组合条件属性超过 4 时生成次要警报。您首先创建一个名为 conditionCheck_ref 的参考属性，并在名为 ConditionCheck 的监视中使用该属性。然后使用“警报说明”对话框创建名为“ConditionCheckAlarm”（或者另一个适当的字符串）的警报说明。

参考属性具有以下参数：

- 名称：ConditionCheck_ref
- 数据类型：整数
- 表达式
  - 表达式：4
  - 实例：无

- 属性：无
- 阈值：无

示例：可用性和测试监视

该监视包括以下参数：

- 名称：ConditionCheck
- 数据类型：整数
- 表达式
  - 表达式：Composite_Condition
  - 实例：无
- 属性
  - 默认激活：活动
  - 评估：EoC
- 阈值
  - 违反阈值，如果值：> ConditionCheck_ref
  - 重置阈值，如果值：<= ConditionCheck_ref
- 警报：
  - 警报重要级别：次要
  - 警报说明：ConditionCheckAlarm
- 脚本：无

第三个可用性和测试监视方案

这里描述的示例监视用于可用性和测试。

在此示例中，如果负载值降到低于 10% 的时间超过 90 分钟，则创建警报。要配置此监视，请更改监视表达式以使用所需的外部属性，并将阈值设置为连续轮询数（等于所需持续时间除以轮询间隔）。

示例：可用性和测试监视

第一个监视包括以下参数：

- 名称：Watch_Load_Under_10_Pct
- 数据类型：整数

- 表达式
  - 表达式：MAX (0, MIN (1, (<此处放置负载低于 10% 时的测试表达式>)))
  - 实例：无
- 属性
  - 默认激活：非活动
  - 评估：根据要求
- 阈值：无

第二个监视包括以下参数：

- 名称：Watch_TimeTicker_LoadUnder10Pct
- 数据类型：整数
- 表达式
  - 表达式：(Watch_TimeTicker_LoadUnder10Pct + 1) * Watch_Load_Under_10_Pct
  - 实例：无
- 属性
  - 默认激活：非活动
  - 评估：根据要求
- 阈值：*所需的连续轮询数*

第四个可用性和测试监视方案

这里描述的示例监视用于可用性和测试。

第一个监视监控 CPU 的持续引用的使用率值（本示例中为 80%）。

如果 CPU 使用率维持在某个特定级别 (80%) 并持续一段时间，则第二个监视触发警报。此监视使用阈值 (3) 乘以轮询时间间隔（5 分钟）计算这个时间段。因此，如果 CPU 使用率超过 80% 达到 15 分钟，则此监视违反阈值，并触发警报。您可以调整这些值以满足您的需求。

示例：可用性和测试监视

第一个监视包括以下参数：

- 名称：CPU_Duration_Over_80
- 数据类型：整数

- 表达式
  - 表达式：MAX(0, MIN(1, INTEGER((INTEGER(cpqHoCpuUtilMin.#) >= 80))))
  - 实例：全部
- 属性
  - 默认激活：活动
  - 评估：根据要求
  - 可继承：False
- 阈值：无

第二个监视包括以下参数：

- 名称：CPU_Time_Duration
- 数据类型：整数
- 表达式
  - 表达式：((CPU_Time_Duration.# + 1) * CPU_Duration_Over_80.#)
  - 实例：全部
- 属性
  - 默认激活：活动
  - 评估：通过轮询
  - 轮询时间间隔：0 + 00:05:00
  - 可继承：False
- 阈值
  - 违反阈值，如果值：>= 3
  - 重置阈值，如果值：< 3
- 警报
  - 警报重要级别：次要
  - 警报说明：ErrorTholdAlarm
  - 用户可清除警报
  - 用户清除警报时不重置监视。
  - 脚本：无