

Customizing ProCurve Manager Plus



How to use the Configurable Integration
Platform to customize and extend
ProCurve Manager Plus

Objectives	3
Requirements	3
Introduction to the Configurable Integration Platform	3
Enhancing the Management of Non-ProCurve Devices	4
Instructions for adding support for non-ProCurve devices	5
Plugging in Other Web-based Applications into PCM+	7
Step 1: Creating the property file	8
Step 2: Copying the file to the right place	9
Step 3: Restart the PCM+ client	9
Customizing the PCM+ User Interface	10
Adding simple actions to the global toolbar or menu	10
Adding "context sensitive" toolbar buttons	12
Making the button appear in more than one group	14
Adding "right-click" actions	14
Passing parameters to action files	15
Controlling which users can use the new triggers	15
Extending PCM+'s Capability to Receive and Decode SNMP Traps	17
Creating a drop-in .trp definition file	17
Location of your .trp file on the management server	17
Naming the root node of your .trp file	17
"Well-known" variables	18
Example .trp file with no variables	18
Example .trp file with variables	18
Resources	19
Troubleshooting CIP	19
TrapPrinter	20
Installation	20
Command-line Usage	20
Sample output	20

Objectives

This paper provides simple step-by-step instructions on how to customize and extend the capabilities of ProCurve Manager Plus using the Configurable Integration Platform (CIP) interfaces. The purpose of these interfaces is to allow non-programmers to integrate other applications and extend the management capabilities of PCM+ with respect to non-ProCurve devices and systems. The specific techniques that will be described in this document include:

- How to better manage non-ProCurve devices using PCM+
- How to integrate other web-based applications into PCM+ to create a “single pane of glass” to manage your network
- How to create custom buttons and menu items to launch your own scripts and tools from within PCM+
- How to enable PCM+ to receive and decode SNMP events (traps) from other applications or unmanaged devices

Requirements

The descriptions in this document assume you have ProCurve Manager Plus installed and licensed. Most of these features are only functional with the “Plus” version.

There is an accompanying zip file available at procurve.com/resources/cpm. This zip file contains samples of all the property files described in this document. They will be valuable as a starting point for making your own customizations.

Introduction to the Configurable Integration Platform

ProCurve Manager is based on a very modular and extensible framework. Thus, there are many ways to extend and/or enhance the capabilities of PCM. However, this paper focuses on mechanisms that can be utilized by nonprogrammers. Some of the mechanisms described in this paper are the same as those used internally by PCM, with a small layer of code provided to enable their use without programming. Others were developed as part of PCM, with the express purpose of enabling nonprogrammers to extend the product. Together, they provide a wide array of ways to tailor PCM to your particular needs. These various mechanisms for extending and customizing PCM are referred to collectively as the “Configurable Integration Platform” or CIP.

The basic mechanism used by most of these extensions relies on the user supplying a “property file.” All of these property files will have the same type of structure or format. The values in the property file will determine the purpose and function. Here are some general principles to keep in mind when creating and/or modifying these property files:

- Always create or edit these files with a simple text editor, such as WordPad or Notepad. Do not edit these files with MS Word or another high-end word processing program since the file format created by such applications will not be usable by PCM.
- Pay close attention to the structure of the file. Many will have “sections” delimited by opening and closing braces (“{” and “}”). If you modify such a section, ensure that you have not deleted one of these delimiters accidentally, or the file will not be processed correctly.
- Pay close attention to the destination directory designated for the file. When you are through creating or modifying the file be sure to save it in the correct location.

At various times in the instructions below, you will be required to “Restart the PCM services.” Rather than repeat the instructions on how to do that each time, please refer to the following steps:

1. Close all open PCM clients.
2. Open the NT Services window on the system where you installed the ProCurve Manager server. To do this, go to the control panel and click on “Administrative Tools.” Then, click on “Services.”
3. Right-click on the “HP ProCurve Datastore” service and select “Restart” from the pop-up menu.

4. It will prompt you if the "HP ProCurve Traffic Launch Service" and the "HP ProCurve Network Manager Server" should be restarted. Select "Yes."

Enhancing the Management of Non-ProCurve Devices

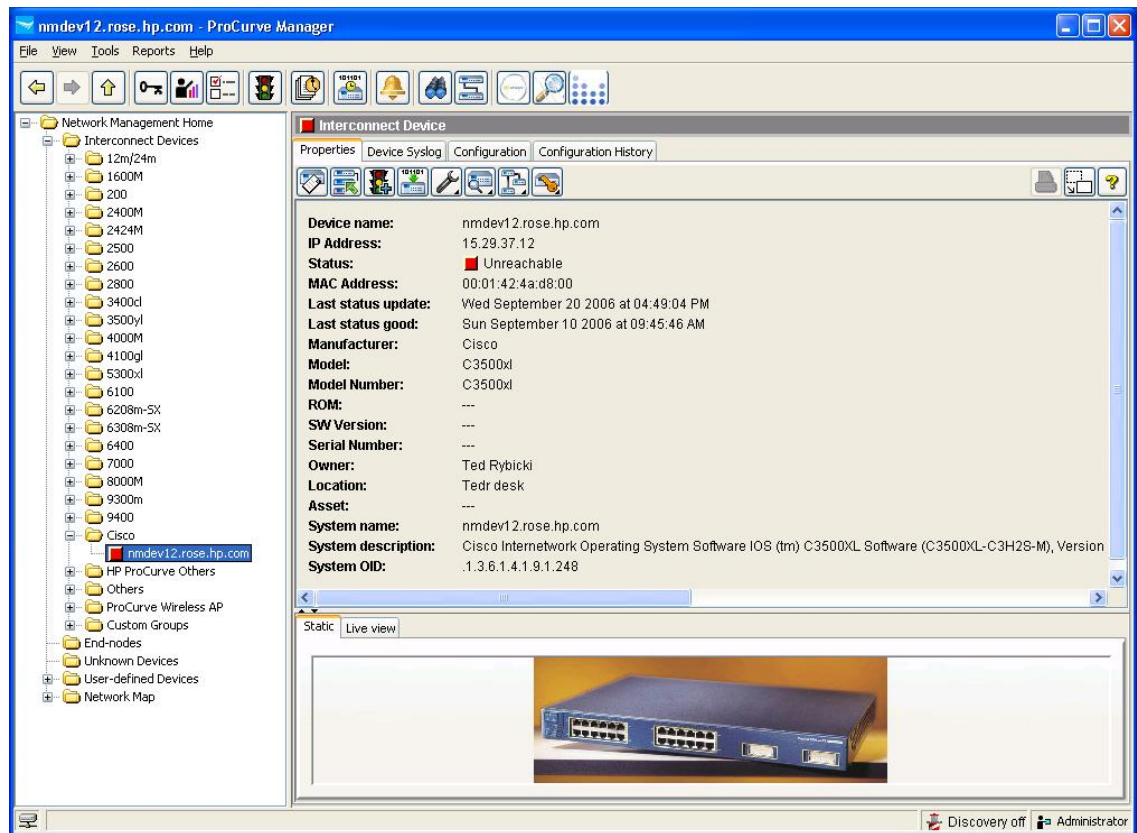
Although you invariably will experience the best and most powerful features of ProCurve Manager Plus when managing ProCurve-branded devices, there are times when a network administrator finds non-ProCurve devices in the mix of their network. These devices are not ignored by PCM. Even without any customizations, PCM will provide some basic management facilities for non-ProCurve devices. These capabilities include:

- **Discovery:** PCM generally will discover non-ProCurve switches. Of course, the device must be SNMP-accessible to PCM using the default SNMP "read" and "write" community names, which you can specify in the PCM preferences setup. Additionally, if you wish PCM to place the device in the correct location in the "map," then the device should support the appropriate discovery protocol. (PCM supports devices which implement both the LLDP and CDP protocols). LLDP (Link Layer Discovery Protocol) is the industry standard Layer 2 discovery protocol, and CDP is the Cisco Discovery Protocol still supported by many networking devices.
- **Monitoring:** If PCM has been able to discover a switch, it will monitor it for connectivity on a regular basis (on a schedule configurable by the PCM administrator).

However, there are more capabilities in PCM that you can enable for such third-party devices with a little work. By creating a custom property file to describe certain attributes of a new device type, you can enhance the basic management of these devices in PCM. Here is a short list of such capabilities:

- Customize the default group to which such devices are added. By default, non-ProCurve devices are all placed into a folder called "Other." If instead you would like to automatically classify different devices into their own custom folder, you can do so. For example, you could create a folder for Cisco devices and a folder for Enterasys devices. As these devices are discovered, they can be classified automatically into their correct folder.
- You can enable the "Live View" of the device in PCM for non-ProCurve devices, if the devices support such a view. If the device provides a web-based UI that contains a convenient view of the status of the device, that view can be shown in the "Live View" tab associated with the device.
- Enable the launching of the full web-based UI to the device.
- Show a picture of the device, which can help your technicians be sure they have selected the correct device.

Below is a screenshot of PCM with a Cisco device selected (using the .oid file shown below the screenshot):



```

Cisco3500xl {
    WebViewEnabled=true
    model=C3500xl
    class=Cisco
    product=C3500xl
    SYSOID=.1.3.6.1.4.1.9.1.248
    vendor=Cisco
    Capabilities {
        isCLI=true
        isSwitch=true
        isCDP=true
        isSFLOW=false
    }
    ImageInfo {
        jarname=ciscoimages.zip
        mapIcon=ciscoicon.jpg
        image=cisco3500.jpg
    }
}

```

This file will be described in more detail in the next section.

Instructions for adding support for non-ProCurve devices

To enable these features, you must create a property file for each type of device you need to support. Each property file will describe to PCM the capabilities and properties of that type of device. The device type is identified by the SNMP system object ID (or sysoid). For example, the sysoid for a Cisco C3500xl is ".1.3.6.1.4.1.9.1.248".

Here are the steps you should follow to add support for a non-ProCurve device:

1. Create the property file describing the device. A sample file is shown and described below these instructions.
2. Save this property file (with a unique name ending in .oid) in the <installdir>\PNM\server\config\devConfig directory.
3. Create a .zip file containing the images (described below) for the device. You also must copy this .zip file into the <installdir>\PNM\server\config\devConfig directory.
4. Restart the PCM services.

Shown below are the contents of one such property file. This file would be used to add support for Cisco C3500xl devices:

```
Cisco3500xl {
  WebViewEnabled=true
  model=C3500xl
  class=Cisco
  product=C3500xl
  SYSOID=.1.3.6.1.4.1.9.1.248
  vendor=Cisco
  Capabilities {
    isCLI=true
    isSwitch=true
    isCDP=true
    isSFLOW=false
  }
  ImageInfo {
    jarname=ciscoimages.zip
    mapIcon=ciscoicon.jpg
    image=cisco3500.jpg
  }
}
```

Notes about this file:

- This file is included in the sample files associated with this paper. The name of the file is Cisco3500.oid.
- The indentation (white space) used in this file is only to make the file more readable but not necessary. The opening and closing braces are necessary, however.
- Description of properties:
 - WebViewEnabled: Specifies whether the device supports a web-based view that can be presented in PCM's "Live View" tab.
 - The default URL that PCM will use to get the "Live View" is: <http://<device IP address>>. Some devices have a view more appropriate to the "Live View" buried deeper in the device's UI hierarchy. For example the "Live View" for ProCurve devices can be found at: http://<device IP address>/configuration/device_viewf.html. If your device also requires some special path, you can specify that path as the value of another property called "WebViewPath" (not shown in the file above). The WebViewPath should be set to the part of the URL following the IP address. For example, for ProCurve devices, the .oid file includes the following property:
WebViewPath=/configuration/device_viewf.html
 - If a different protocol (other than http) should be used to get the live web view, then that also can be specified with a property called "WebProtocol." For example, if the device in question only supports https, you can specify that with the following additional property:
WebProtocol=https
 - Model, vendor and product: These properties will display in the "Device Properties" tab in PCM.

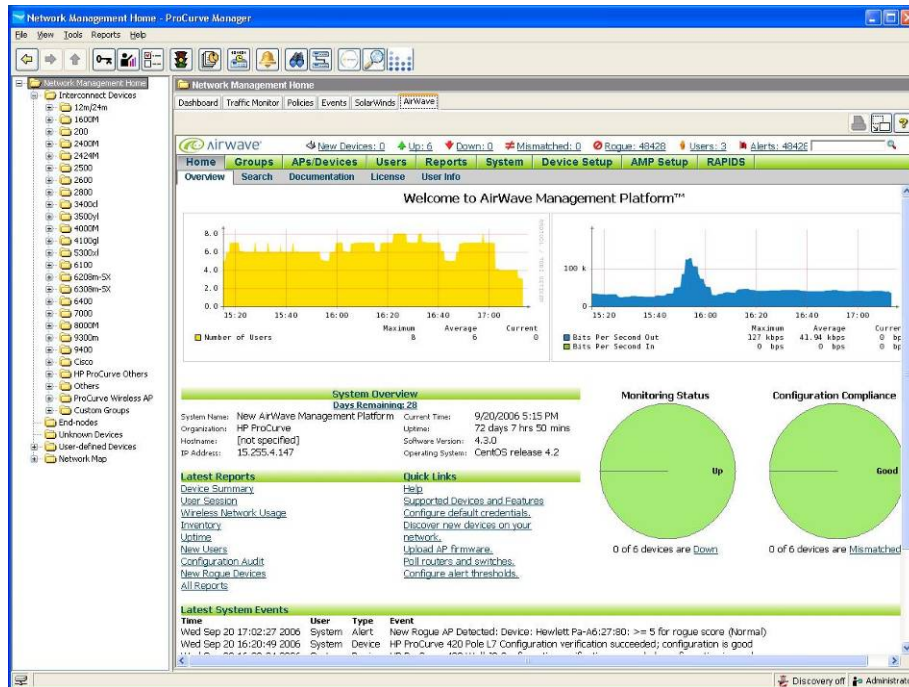
- Class: Whatever value you specify here will be used to create a folder within the PCM tree by that name. All devices with the specified class will be grouped in that folder. In this example, all the devices will appear in a folder named "Cisco."
 - SYSOID: You must specify the SNMP system object ID here. Be sure to include the leading "."
- Capabilities: The capabilities section of the file describes the PCM properties necessary to enable some functionality. These properties are described below:
 - isCLI: Indicates if the device allows Telnet access. If set to "true," PCM will enable a right-click action to launch a telnet session to the device.
 - isSwitch: Generally, you will set this to true. Any device that routes, forwards or bridges network traffic should have it set to true.
 - isCDP: Indicates whether the device supports the Cisco Discovery Protocol. If set to true, PCM will attempt to read CDP information from the device, which will enable PCM to construct more accurate network maps, and will discover the network topology much more quickly.
 - isSFLOW: Indicates if the device supports sFlow, which is used by Traffic Monitor.
- The final section of the file is the "ImageInfo" section. This section specifies where PCM can find images it should display when the device is selected. The images for the device should be in a .zip file, and the "jarname" property must be set to the name of the .zip file containing the images.
 - There are two images which you can specify for each device, the "mapIcon" and the "image."
 - The "mapIcon" specifies the name of the small image used to represent the device on the network map (it should be a small image, no larger than 64x64 pixels).
 - The "image" property must specify the name of the large image, which is displayed on the device properties tab when the device is selected in the PCM tree.
 - The .zip file should be copied into the same directory as the .oid file, that is: `<installdir>\PNM\server\config\devConfig`
 - The files associated with this paper contain a zip file with the images for the Cisco device described in the sample .oid file above. See the ciscoimages.zip file. You should copy this file to the `<installdir>\PNM\server\config\devConfig` directory.

Tip: If you are creating several .oid files in order to support different types of devices, you may put all the images in the same .zip file and reference the same .zip file in each .oid file.

Plugging in Other Web-based Applications into PCM+

You can plug in the user interface for other web-based applications into the PCM+ user interface to give you a single integrated pane of management. PCM+ will allow you to create a custom "tab" with contents that will be the application of your choice (as long as that user interface is a web-based user interface supported by Internet Explorer).

For example, below is a view of PCM+ with the AirWave Management Platform application displayed in a tab named "AirWave."



This can be accomplished in three easy steps:

1. Create a simple text property file describing the application you wish to plug in.
2. Copy this file into a particular location on the PCM server.
3. Restart your PCM+ client (no need to restart the server).

Below are the details and examples needed for completing these steps.

Step 1: Creating the property file

You need to create a property file that specifies the attributes of the application. The format of the file is simple. Here is a sample file:

```
AirWaveTab {
    Scope=WebTab
    TabName=AirWave
    NodeName=Network Management Home
    URL=https://10.3.4.147
}
```

Notes on this property file:

- The file can be named anything you want, but it must have the ".trg" extension.
- Be sure to create and save the file with a simple text editor such as WordPad or Notepad. Do not create the file with MS Word or another high-end word processing program.
- The Scope property must be set to "WebTab." That specifies that a custom tab should be created for the new application.
- The TabName property can be set to any value you like. Whatever you put there will appear as the name of the tab in PCM. In this case, we chose to call it "AirWave."
- The NodeName property specifies the name of the node in the PCM+ navigation tree that will be associated with the tab. If you look at the screenshot above, you will see that the "Network Management Home" node in the tree is selected. The tab we created for the AirWave application will only appear when that node is selected. You may specify the name of any node in the PCM+ tree, including the names of Custom Groups which can be quite useful for plugging in applications for specific groups of devices.
- Finally, the URL property must specify a web address/path to the server of the application. In this case, the URL needed to launch the AirWave Management Platform is <https://10.3.4.147>.

Step 2: Copying the file to the right place

Once you have created the property file, you must save it in the *installdirectory>\PNM\server\config\devconfig\extern* directory.

Step 3: Restart the PCM+ client

No need to restart the PCM+ server. You should now see your tab.

If it doesn't appear, check the syntax of the file carefully to ensure it matches the format shown above. Also, make sure it was copied into the correct location on the PCM+ server. See the section on Troubleshooting for more suggestions.

A sample file is included in the [files associated](#) with this paper. See the file called "GoogleTab.trg."

Customizing the PCM+ User Interface

PCM provides a mechanism whereby you can plug in external tools and have them launched from within the PCM user interface. There are two ways that external applications can be launched from within PCM:

1. Global tools. These are applications that are launched from the “global” toolbar or PCM application menus. These applications can be launched at any time regardless of what you have selected within PCM. They are always enabled and available.
2. Contextual tools. These are applications that are associated with a particular context within PCM. In other words, contextual tools can be executed only when a particular device or type of device is selected in the PCM navigation tree on the left. Contextual tools can be broken down further into two types: toolbar items and right-click menu items. The instructions below will show you how to create both types.

In either case, the way you plug in applications into PCM is by creating simple text property files and copying them into the correct location. PCM will then read these files and create the appropriate buttons and/or menu items.

The instructions below will give you detailed step-by-step instructions on how to add your own buttons and menu items, but first, an explanation of the two basic concepts that pertain to these tasks is required.

Triggers

Triggers are visible items in the User Interface that you will use to launch or execute some action. Examples of triggers are toolbar buttons, menu items, and right-click menu items. For each trigger, you will create a small, simple text property file that describes the properties of the trigger, such as the icon to use, the tool tip to show if the user hovers over the button, or the name of the menu item.

Actions

Actions are what you actually want to do when a given trigger is fired. Generally, actions consist of the command line used to launch an external application, or the URL of a website to open up in the browser. For each action, you will create a simple text file that describes properties of the action, such as the command line of the external application, or the URL of a website.

Note that many triggers can refer to a single action, so that you can create multiple ways to launch an action, such as from a toolbar button, a global menu item, or a right-click menu item. All could end up launching the same action, if you desire.

Adding simple actions to the global toolbar or menu

This section of the paper will show you how to add a “global” toolbar button that, whenever activated, will launch the browser to a URL of your choice.

1. The first step is to create an “action” property file. Action files must have an extension of “.uda” and you must copy them to the <installdirectory>\PNM\server\config\devconfig\extern directory. This file will define exactly what command you want executed when the item is activated. Below is a sample action file to launch a browser to Google. You can find a copy of this file in the files associated with this paper. Look for a file called “LaunchGoogle.uda.”

```
Google {
    Name=Launch Google
    Type=WEB
    Command=www.google.com
    Target=Client
}
```

Notes about this file:

The name before the curly braces is the “**Action ID.**” This name will be used later to refer to this particular action. In this case, we have used the identifier “Google.”

The “Type” specifies the type of action you want to execute. There are three types:

- WEB - Will launch a browser to a specified URL
- CLI - Will execute a command line to launch an external application
- POLICY - will execute a previously created policy

The “Command” provides the necessary information to execute the action. In the case of a WEB type, it will specify the URL. If the type is CLI, it will specify the command-line to launch. If the type is POLICY, it will specify the name of the policy to execute.

2. Once you have examined the file, you need to copy it to the correct location.

Copy **LaunchGoogle.uda** to the

<installdir>\PNM\server\config\devconfig\extern directory.

3. Now you must create the “**trigger**” file for the action. The trigger file specifies how the action is to be initiated. Trigger files always must have the “.trg” extension. As discussed above, it can be a global trigger or a contextual trigger. For this example, we will create a global trigger. Below is a sample of the trigger file for a global toolbar button. (Look for the “LaunchGoogle.trg” file in the files attached to this paper.)

```
LaunchGoogle {
    Scope=Global
    Type=TOOLBAR
    Name=Launch Google
    Global {
        ToolGroup=CustomTools
    }
    ActionID=Google
    Permission=PER_OPERATOR_1
    Tooltip=Launch Google
    Icon=Google.jpg
    Jarname=icons.jar
}
```

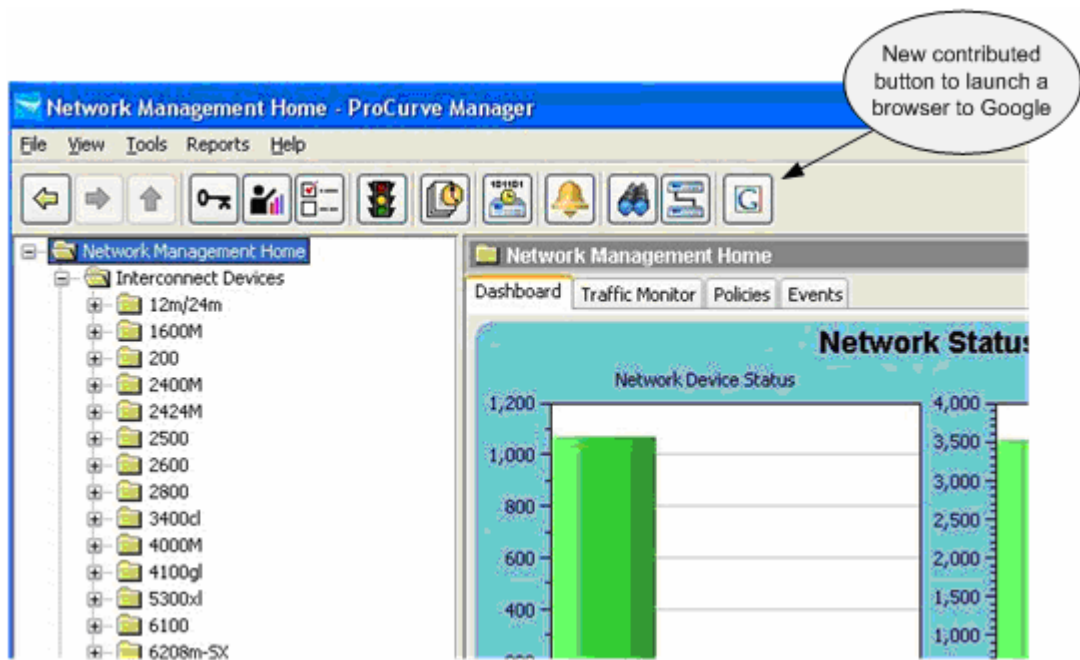
The important entries in this file are:

- Scope - This entry must be “Global” for this type of trigger.
- Type - This entry specifies whether you want a toolbar button or a menu item.
- ActionID - This entry specifies the action you want to execute when the button is pressed.
- Icon - Specifies the name of the icon to use as a graphic for the button.
- Jarname - Specifies the archive containing the icon. (This way, you can have a single archive with all the images and icons used by all your triggers). This file can be either a standard .zip file or a Java .jar file (which is just a .zip file with the extension changed to .jar).

1. As before, you need to copy the LaunchGoogle.trg trigger file to the <installdir>\PNM\server\config\devconfig\extern directory.

You also need to copy the **icons.zip** file to the same directory. You also will find the **icons.zip** file in the [files associated](#) with this paper.

2. You are now ready to shut down the PCM client and restart the client. You do not need to restart the PCM services or reboot the PC. Once you have reopened the PCM client, you should see a new button on the global toolbar. It should look like the screenshot shown below, and pressing on the new button should launch the browser to the Google website.



Adding “context sensitive” toolbar buttons

This section will describe how to add a button on one of the device group tab views that will be enabled only when a device of a certain type is selected. This is called a “contextual” trigger because you must be in a particular context in order for the button to be enabled.

Suppose you had a tool such as a MIB browser that is useful for performing certain specific management or monitoring tasks on network devices. You might want to be able to select a particular device somewhere in PCM and then, launch the tool in such a way that the IP address of the selected device is passed to the command line of the tool. This way it will come up ready to interact with the specified device. This section will describe how to do exactly that.

These instructions presuppose that you have previously installed the tool you wish to use, such as the ProCurve MIB browser application. To demonstrate this, we will configure PCM to create a button in the toolbar for the 4100 device group. When activated, this button will launch the MIB browser for the currently selected device. The button will be enabled when one, and only one, device is selected in the table.

1. As before, the first step is to create the action file. Here is the action file you will use. (You can find this file in the zip [file associated](#) with this paper under the name MibBrowser.uda.)

```
MibBrowser {
    Name=MIB Browser
    Type=CLI
    Command=C:\Program Files\Hewlett-Packard\ProCurve MIB
    Browser\ bin\ProCurve MIB Browser.exe %ip
    Target=Client
}
```

Notice the **%ip** at the end of the command line. This indicates that when the command is activated, an IP address for the currently selected device will be substituted here. See Section 6.2.3 called “Passing Parameters to Action files” below for more details.

If you are using this file to launch a different application of your own, you need to edit the Command to specify the correct command line. You also will want to change the Action ID and probably the Name, as well.

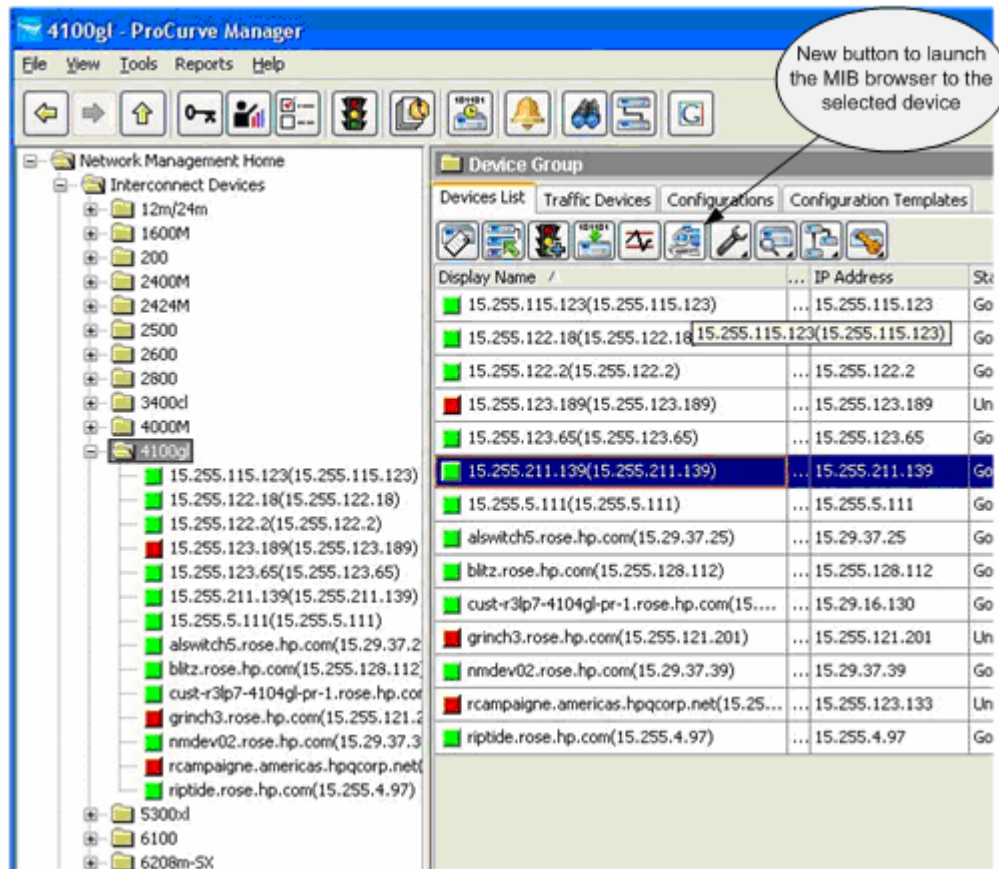
2. As before, copy this file to the
<installdir>\PNM\server\config\devconfig\extern directory.
3. Now we must supply the "trigger" file. You also will find this example trigger file in the [zip file associated](#) with this paper under the name LaunchMIB4100.trg. The file is shown below:

```
LaunchMIB4100 {
  Scope=Context
  Type=TOOLBAR
  Name=ProCurve MIB Browser
  Context {
    GroupTab {
      Selection=1
      GroupName=4100
    }
  }
  ActionID=MibBrowser
  Permission=PER_OPERATOR_1
  Tooltip=Launch MIB Browser
  Icon=mibbrowser.gif
  Jarname=icons.zip
}
```

Important differences to note in this file from the trigger file we used in the previous example:

- a) The Scope has changed from "Global" to "Context" to indicate that this trigger only applies within a particular context of PCM.
- b) Note the "Context" entry, which specifies the name of the group that this trigger will apply to, as well as the "Selection=1" property, which specifies that the trigger should be enabled only when one and only one device is selected.

1. As before, you now need to copy this file to the
<installdir>\PNM\server\config\devconfig\extern directory.
The **icons.jar** file (which you already copied to the extern directory in the previous example) also contains the image for the new button, so you do not need to copy this file again.
2. You are now ready to shut down the PCM client and restart the client. You do not need to restart the PCM services or reboot the PC. Once you have reopened the PCM client, you can do the following to test the new trigger:
 1. Select the 4100 device group in the left-hand navigation tree.
 2. Select any device in the table view on the right.
 3. The new button (shown below) should be enabled. Clicking on it should launch the MIB browser. (See image below.)



Making the button appear in more than one group

It would be inconvenient if you had to create a separate trigger file for every group within PCM where you wanted the button to appear. Fortunately, you don't have to do that. You can modify the "GroupName" property in the trigger file to specify a comma-separated list of group names. The new button will appear in every group named in the list. For example, below is what the GroupName property should be set to, in order to make the button appear in several groups. (Note that this feature is supported only if you have PCM 2.1 with update #7 applied. Earlier versions of PCM do not support this.)

GroupName=4100, 2500, 2600, 3400c1, 5300x1, Interconnect Devices

Adding "right-click" actions

There may be times when it is more convenient to add a "right-click" menu item to a device rather than to add a toolbar button to a group. This way, you can right-click on the device, even in the PCM navigation tree or from the network map, and still be able to launch your custom actions.

Creating a "right-click" menu item is very similar to the example above, with just a few changes to the trigger file:

- The Type property changes from "TOOLBAR" to "RIGHTCLICK."
- Instead of the GroupTab entry, you must specify a Device entry that specifies the type of devices to which you wish to add the right-click trigger. This can be specified in two ways:
 - You can specify an IP address (in which case the right-click menu item will appear on just that one device). An example of the entry would look like this:


```
Device {
```

```

        DevType=IP
        Value=10.1.2.25
    }
    o You can specify a sysoid value, in which case the right-click menu item will
      appear on all devices of that specified type. An example of the entry would look
      like this:
      Device {
          DevType=OID
          Value=.1.3.6.1.4.1.11.2.3.7.11.32
      }

```

A complete right-click trigger file is shown below:

```

LaunchMIBb {
    Scope=Context
    Type=RIGHTCLICK
    Name=MIB Browser
    Context {
        Device {
            DevType=OID
            Value=.1.3.6.1.4.1.11.2.3.7.11.32
        }
    }
    ActionID=MibBrowser
    Permission=PER_OPERATOR_1
    Tooltip=Launch MIB Browser
    Icon=mibbrowser.gif
    Jarname=icons.jar
}

```

Passing parameters to action files

As was demonstrated in Section 6.2, it is possible to define the Command in the action file so that it can be passed information about the selected device when the action is triggered. The example we used above was to include the "%ip" parameter in the command line. However, there are several other predefined parameters which you also can include as needed in your command lines. Here is a complete list:

- %ip - Is replaced with the IP address of the selected device
- %read - Is replaced with the SNMP "read" community name for the selected device
- %write - Is replaced with the SNMP "write" community name for the selected device
- %manuser - Is replaced with the telnet "manager" username for the selected device
- %manpass - Is replaced with the telnet "manager" password for the selected device
- %opuser - Is replaced with the telnet "operator" username for the selected device
- %oppass - Is replaced with the telnet "operator" password for the selected device
- %oid - Is replaced with the sys object ID of the selected device
- %gid - Is replaced with the group name of the selected group (if one is selected)

Controlling which users can use the new triggers

When you create a trigger, you can specify what type of user is allowed to exercise the action. There are three choices: Administrator, Operator and Viewer. If you noticed in the trigger files above, each file contained a line that specified the permission like this:

```
Permission=PER_OPERATOR_1
```

This entry determines what kind of user is allowed to activate the item. The allowed values for the "Permission" property are:

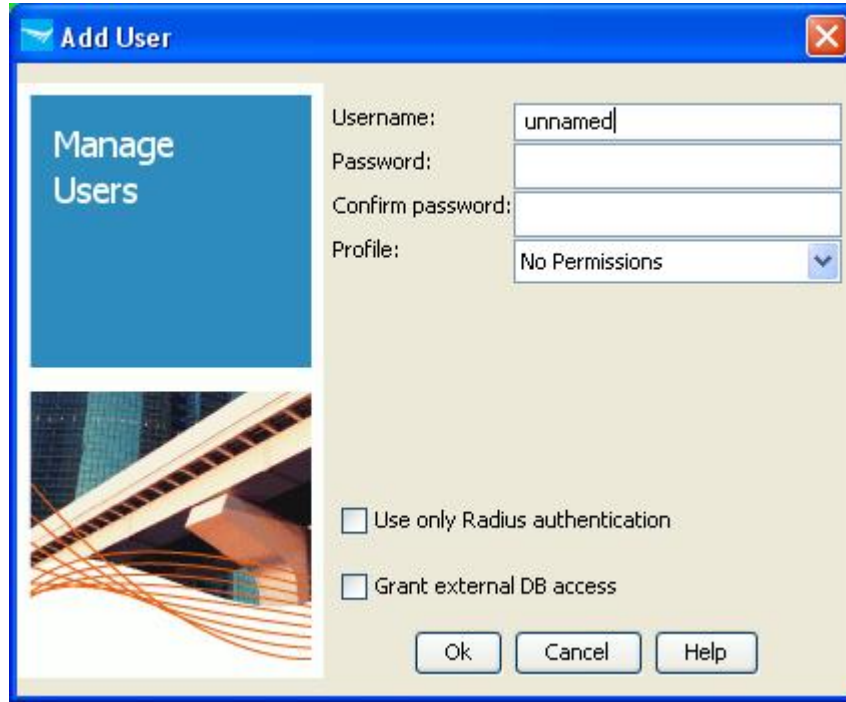
- a) PER_ADMIN_1 The trigger will only be enabled for Administrators. This is the most restrictive setting.

- b) PER_OPERATOR_1 The trigger will only be enabled for Administrators and Operators.
- c) PER_VIEWER_1 The trigger will be enabled for all users. This is the least restrictive setting.

To demonstrate this concept, we will create a user with viewer privileges and verify that the new buttons are disabled for that user.



1. To create a new user, click on the “Manage Users” button
2. Click on Add... (the dialog shown below will appear)



- a) Type in the name of a new user
- b) Enter the user’s password (twice)
- c) Select “Viewer” in the “Profile” drop-down box.
- d) Click OK to save the user.

1. Close the Manage Users dialog.



2. Now log in using the new user you created. To do this, click the “Login” button
3. Once successfully logged in as the new user, you should be able to verify that any buttons created with the triggers from the examples above remain disabled (because those triggers specified that the user must be an “Operator” in order to be able to exercise the feature).

Extending PCM+'s Capability to Receive and Decode SNMP Traps

PCM also will allow you to specify information on how to decode and display SNMP traps from third-party applications or services not ordinarily supported by PCM.

The mechanism of processing third-party traps uses the existing trap processing system. To process a specific third-party trap, a configuration file must be created and "dropped in" into PCM. The Event Manager Server Component is responsible for handling the drop-in files and processing the traps.

Creating a drop-in .trp definition file

A .trp is a property file which defines the attributes necessary for PCM to decode the trap. Each file should contain the following attributes:

- **SEVERITY** - The severity of the event, it is up to the developer to determine this. Possible values are:
 - Informational
 - Warning
 - Minor
 - Major
 - Critical
- **FRIENDLY_NAME** - This is a descriptive name used by the customer to identify the event
- **BASE_TEXT** – This is the base text for the third-party trap. It can have placeholders in it, such as %VARIABLE_NAME_1, %VARIABLE_NAME_2. If the BASE_TEXT key entry is not in the definition file, PCM will make a "best effort" attempt to convert the event to a string, but results may be unpredictable. There are "well-known" variable names that PCM uses for things like Policies and Actions on events; if you wish to utilize this feature of PCM, the third-party trap must define these variables. See Section 7.4. This is the text that will be visible to the customer from the Event Browser.
- **VARIABLE_NAME_X** – "X" is the variable number. So, for example, if you have three variables, they would be named VARIABLE_NAME_1, VARIABLE_NAME_2, and VARIABLE_NAME_3. The VARIABLE_NAME key can specify where to find the value in two ways:
 - The first is by simply defining the **INDEX** tag. The INDEX tag defines the index into the array of values encoded in the SNMP trap.
 - The second is by defining the INDEX tag as well as the **TABLE_NAME** tag. The TABLE_NAME tag should be used when the value at the specified index needs to be translated to another value. PCM will retrieve the value at the specified index of the SNMP trap, and use it to find a matching property in the specified table. If such a matching property is found, then the value associated with that property is returned and substituted in the proper place in the BASE_TEXT string.
- **XXX_TABLE** – This is a list of name/value pairs used to translate values located at an index of the SNMP trap to another value.

Location of your .trp file on the management server

After creating your .trp file, place it in the .../server/config/devConfig/extern directory on the management server.

Naming the root node of your .trp file

It is critical that the root node of the .trp adheres to the following naming convention:

- The name of the root node of your .trp files must be the OID of the trap with "." delimiter replaced with a "_" delimiter. For example, if the OID of the trap is 1.3.4.1.6.1.11 the root node name will be 1_3_4_1_6_1_11.

“Well-known” variables

As mentioned above, there are some well-known variable names that PCM looks for when processing traps. It is not mandatory to define these names, but if these variable names pertain to the values in your trap, it is strongly recommended to define these variables. The “well-known” names are as follows:

- END_NODE_IP_LIST – A list of one or more IP addresses that belong to one or more end-nodes. End-nodes are defined as a Server, client machine, printer, etc.
- END_NODE_MAC_LIST – A list of one or more MAC addresses that belong to one or more end-nodes. End-nodes are defined as a Server, client machine, printer, etc.
- PORT_LIST – A list of one or more ports
- DEVICE_IP_LIST
- DEVICE_MAC_LIST
- RISING_THRESHOLD – The rising threshold that was exceeded
- FALLING_THRESHOLD – The falling threshold that was violated
- THRESHOLD_DELTA – The delta between the threshold and the value that was violated

Example .trp file with no variables

```
1_3_1_4_6_1_11{
    SEVERITY=Informational
    FRIENDLY_NAME=IDS initialization trap
    BASE_TEXT=IDS started and running
}
```

Example .trp file with variables

Below is an example .trp file that can be used to decode an Airwave Management Platform event indicating that an AP has gone down. This file can be found as an example in the [associated files](#) under the name **AmpAPdown.trp**.

```
1_3_6_1_4_1_12028_4_15_13 {
    SEVERITY=Major
    FRIENDLY_NAME=AP Down
    BASE_TEXT=AP Down: IP=%DEVICE_IP_LIST :
    Description=%DESC
    VARIABLES {
        DEVICE_IP_LIST {
            INDEX=3
        }
        DESC {
            INDEX=2
        }
    }
}
```

Example .trp file with variables and tables

```
1_3_1_4_6_1_13{
    SEVERITY=Critical
    FRIENDLY_NAME=Rogue AP detected
    BASE_TEXT= Rogue AP %IP_ADDRESS detected on radio
    %RADIO_NUM. Detected by %DETECTION_METHOD
    VARIABLES{
        IP_ADDRESS {
            INDEX=0
        }
    }
```

```

        RADIO_NUM {
            INDEX=1
        }
        DETECTION_METHOD {
INDEX=2
TABLE_NAME=DETECTION_TABLE
        }
    }
    TABLES {
        DETECTION_TABLE {
1=Scanning
2=Association
3=Attempted Authentication
            DEFAULT=unknown
        }
    }
}

```

*NOTE: If names in the XXX_TABKE keys contain a "." they will substituted with a "_". So if the value in a PDU is an OID, all the "." delimiters will be replaced with a "_".

Values must be contained on a single line. So for example, the BASE_TEXT value (above) must not span more than one line (it is wrapped over two lines here just so it can fit on the printed page).

Resources

Troubleshooting CIP

If you are not getting the expected results, here are some things to check.

- Are you running the latest version of PCM? Some of the CIP features described here are not enabled unless you have the latest release of PCM with all the auto-update patches applied. At a minimum, you should have PCM 2.1 with auto-update #7 applied.
- Did you save the property file with a simple text editor rather than a word processing program? Try opening the property files you created with Notepad to verify that the file is readable.
- Double-check the syntax of the property files. Are all opening braces ("{") matched by a closing brace?
 - Check the Events tab in PCM. If PCM encounters a CIP property file with bad syntax, it will create an event indicating the file that caused the problem. The source of the event will be "CoreServer (Config.Integration)," and the detailed message will read, "Syntax error parsing user-defined Trigger file (<filename>)." The severity level of the error will be "Warning."
- Is the file stored in the correct directory?
 - Most CIP files should be copied to "<installdir>\PNM\server\config\devConfig\extern" (for most default installations, <installdir> will be C:\Program Files\Hewlett-Packard).
 - The .oid files needed to add support for non-ProCurve devices are the exception to the above rule. These files should be copied to "<installdir>\PNM\server\config\devConfig". Note that the image zip files containing the images and icons for the non-ProCurve devices also should be in the same directory as the .oid files.
- Did you restart the PCM client? Note that for adding support to decode new SNMP traps (events), the PCM server must be restarted, as well.
- Is the name of the main property unique? In the property files, note that they all start with a name followed by a curly brace, for example:

```

MibLaunchTrigger {
    ...
}

```

In this case, the "MibLaunchTrigger" must be a unique name. If some other property file also uses the name "MibLaunchTrigger" as the main property, then only one of them will be acknowledged and used.

TrapPrinter

TrapPrinter is a simple utility that can be used to scan MIB definition files for NOTIFICATION-TYPE definitions. This simplifies the tedious task of reading through a MIB file looking for the exact OIDs and description of the traps.

TrapPrinter is a Java application, so you must have the Java 1.5 runtime environment (or JRE) on your PATH. The TrapPrinter tool is included as a separate zip file inside the file associated with this paper. Following are some basic instructions for installing and using this tool.

Installation

- The TrapPrinter is packaged into a zip file called "TrapPrinter.zip"
- Create a directory for TrapPrinter and copy the TrapPrinter.zip file to this directory. For example: C:\TrapPrinter
- Extract the files in TrapPrinter.zip to the directory you just created
- Please insure that you have the Java runtime environment on your PATH. It should be a version of Java greater than or equal to version 1.5

Command-line Usage

Once the TrapPrinter tool is extracted to a directory of your choosing, you are ready to use the tool.

1. Open a DOS command shell window
2. Go to the directory where you extracted TrapPrinter
3. Usage: TrapPrinter [mibfile] > [outputfile]
For example, included with the TrapPrinter is a MIB definition file for the AirWave Management Platform (AWAMP-MIB.my). So, if you issue the following command:

```
TrapPrinter AWAMP-MIB.my > amptraps.txt
```

You will see after running that command that a new file called "amptraps.txt" has been created. It will contain the listing of all the documented traps in the file.

Sample output

The output of TrapPrinter gives a listing where each line of output represents one unique trap definition. Each line will contain multiple fields where:

- Field 1 is OID of the trap
- Field 2 is Name of the trap
- Fields 3 to n Names of the fields in the trap in the order contained in the trap

For example, below is an excerpt from the output resulting from TrapPrinter running against the AirWave MIB file supplied as a sample:

```
1.3.6.1.4.1.12028.4.15.0.9      poorSignalAP awampEventID awampEventSeverityCode  
                                  awampEventDescription awampAPIP  
  
1.3.6.1.4.1.12028.4.15.0.11    unauthenticatedClient awampEventID awampEventSeverityCode  
                                  awampEventDescription  
  
1.3.6.1.4.1.12028.4.15.0.12    rogueAPDetected awampEventID awampEventSeverityCode  
                                  awampEventDescription  
  
1.3.6.1.4.1.12028.4.15.0.13    downAP awampEventID awampEventSeverityCode  
                                  awampEventDescription awampAPIP  
  
1.3.6.1.4.1.12028.4.15.0.14    discoveredAP awampEventID awampEventSeverityCode  
                                  awampEventDescription awampAPIP  
  
1.3.6.1.4.1.12028.4.15.0.15    upAP awampEventID awampEventSeverityCode awampEventDescription  
                                  awampAPIP
```

This output is useful for creating your own trap definition files, as described in Section 7.

To find out more about
ProCurve Networking
products and solutions,
visit our web site at

www.procurve.com



© 2004 Hewlett-Packard Development Company, L.P. The information contained herein is subject to change without notice. The only warranties for HP products and services are set forth in the express warranty statements accompanying such products and services. Nothing herein should be construed as constituting an additional warranty. HP shall not be liable for technical or editorial errors or omissions contained herein.

4AA1-0929ENW, 04/007