BNCS User Guides

Atos IT Solutions and Services

Targeting V4.5 Panels

BNCS

Contents

BNCS

1 Introduction

This document describes how to "target" panels hosted within panel manager application panelman.exe. Targeting a view refers to the process of telling the controls what device to look at. You may have 10 identical ARCs in a system but you don't write a panel for each of them but rather one panel that you target to the ARC you want to adjust.

It is necessary to understand the way in which panel manager launches panels with the "Panel adjust" mechanism (e.g. PA ARC1). Please see the panel manager documentation for this.

1.1 Definition of Terms

A view	Traditionally an entire ApplCore application within "traditional" BNCS. A view describes a single UI "panel" (dialog), or perhaps several UI "panels" (dialogs) that go together to enable control of a particular piece of equipment or logical function.
An instance	An "adjustable" thing. This can refer to a single piece of equipment or several associated together that will be presented to the user in a single "view"
"Target" a view	tell a view which device(s) it should be looking at

1.2 Instances

It is necessary to understand the notion of an "instance". An instance is an "adjustable thing". This can mean either a single piece of equipment or a collection of associated equipment that makes up a view. An instance is the collection of data required to define that "adjustable thing".

Let's say you have a number of identical network chains with two ARCs one for "main" and one for "reserve". This is to be used for 2 network chains "Network A" and "Network B". "Network A" uses "ARC1" and "ARC2", and "Network B" uses "ARC3" and "ARC4".

Let's use this example and write down the information required to make this panel work.

Instance Name	**Data**
ARC1	device number=101
ARC2	device number=102
ARC3	device number=103
ARC4	device number=104
Network A	"main" Instance=ARC1 "reserve" Instance=ARC2
Network B	"main" Instance=ARC3 "reserve" Instance=ARC4

The table above describes 4 single instances (those that describe a single device) and 2 "composite" instances that describe 2 or more devices.

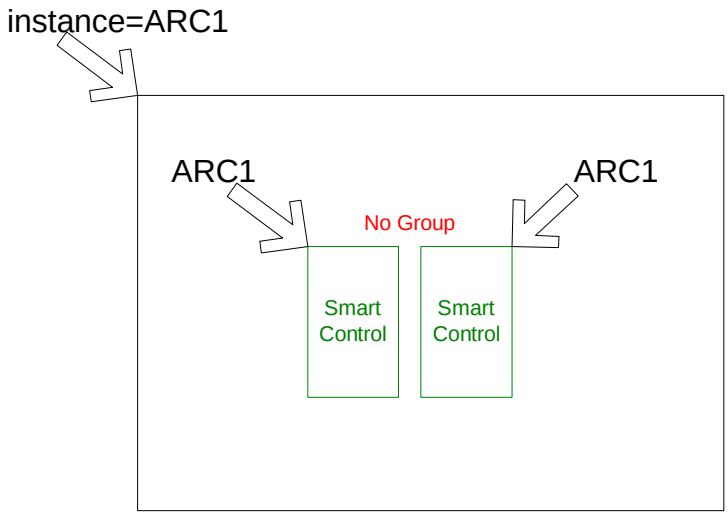
If we tell a view that it is to deal with ARC1 the controls on this view can determine for themselves that this is BNCS device number 101. It can then go and direct all the relevant controls on that view to look at ARC1.

Note: Quite often a device number must be qualified with an "offset" but this is left out of this example for clarity.

If we tell a view that it is to deal with "Network A" the view can determine that this is made up of two ARCs, one called "main" and one called "reserve" and that the ARC instances to use are "ARC1" and "ARC2" respectively. The view will then go off and find all the components in the view that are associated with "main" and target them to "ARC1", and similarly for "reserve"/"ARC2".

1.3 Example – simple instance

Let's take this simple view to which is determined as the view for instance "ARC1" (see the panelman documentation for this, but this is done in the type handlers documentation).



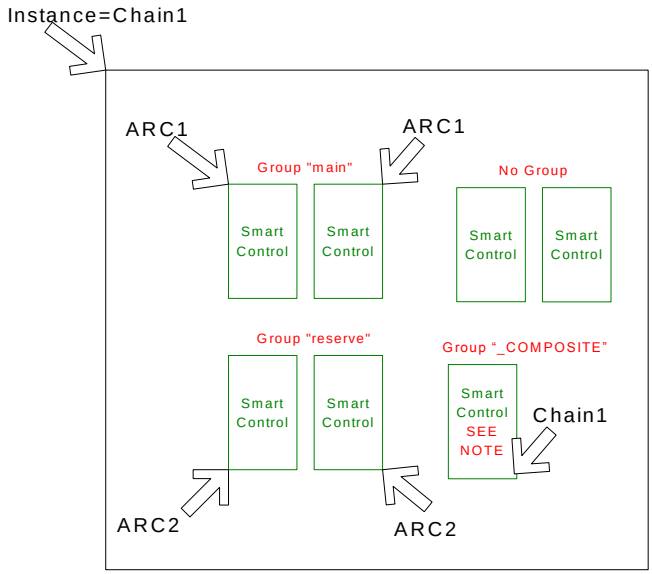
Controls must not be a member of a group. All controls are passed the instance of "ARC1" direct from the parent. It's then up to the controls to determine what that instance means to them.

Simple instances are normally a single UI loaded without a script from a bncs_ui description file.

1.4 Example – composite instance

Let's take this composite view to which is determined as the view for instance "Chain1".

The red "group" indications are actually configuration settings within each smart control rather than any external or graphical representation of group.



Controls must be a member of a group to be targeted. When the view is targeted, the group of each control is determined and this is tied up with the instance information and the relevant instance passed on to the controls. Note that no instance information is passed onto controls that aren't in a group.

Non-smart controls are not "targeted" to anything since they have no notion of a "target" or "group".

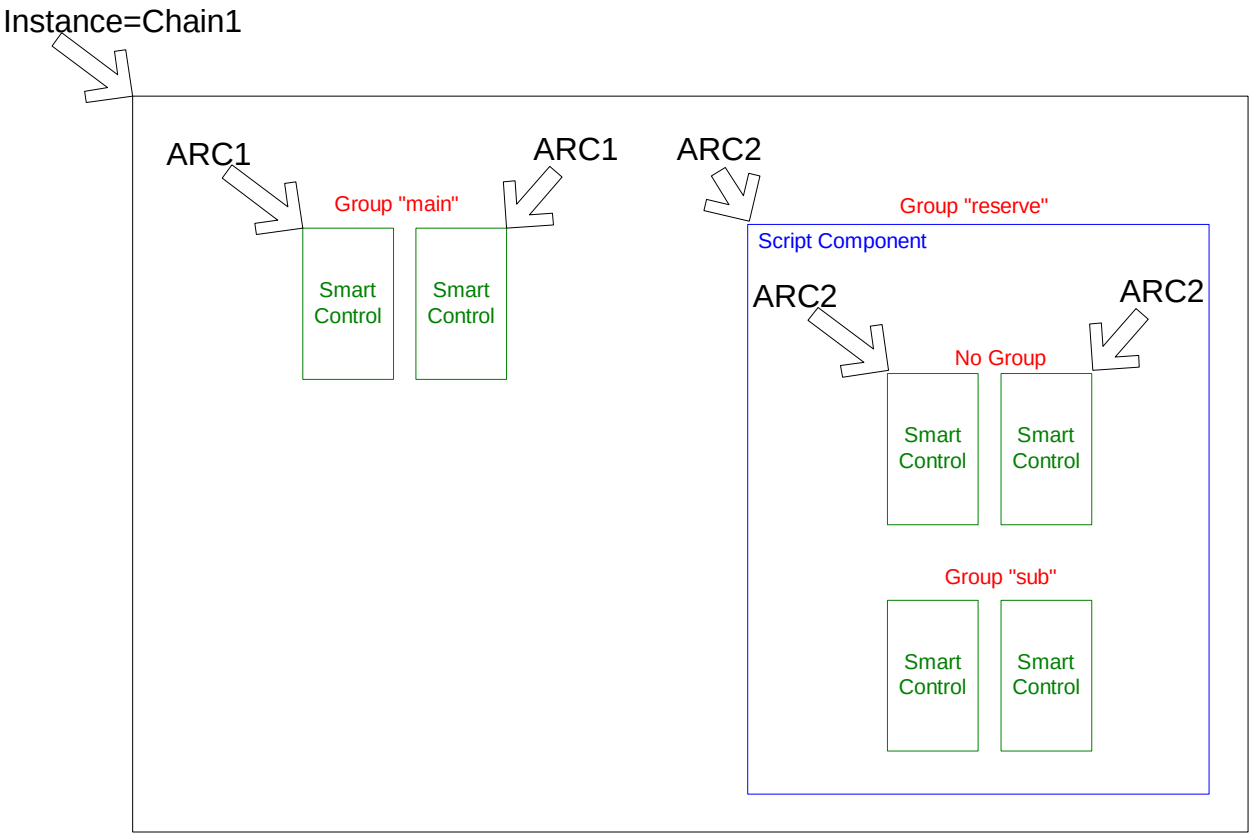
NOTE: Version 4.5.9 of dcw.dll allows for a "special group" on panels targeted to a composite instance. If the group is "_COMPOSITE" then the component is targeted at the top level instance not one of the named groups.

1.5 Example – composite instance with sub-script controls

This shows what happens with a simple script component placed on a normal dialog.

The script component acts just like another top level view. i.e. if you pass a simple instance to a script component it will pass it on to all its smart controls not in a group.

But what about script components that have multiple dialogues? Smart Controls are all treated equally as if all the controls existed on a single dialog.



This opens up another mind blowing topic. Considering the same example how do we target the "sub" group on the script control?

The gist is to remember that Script components are treated in exactly the same way as top level views. So if you pass it a composite instance it will target controls in groups. This means that you have to setup instances that point to instances that point to instances.

This would be an example that could be used for targeting the "sub" group:

Instance Name	Data
ARC1	device number=101
ARC2	device number=102
ARC3	device number=103
ARC4	device number=104
Chain 1	"main" Instance=ARC1 "reserve" Instance= Sub Chain 1
Sub Chain 1	"sub" Instance=ARC3

This means you can nest as many script controls as you like.

1.6 Complex instance

A complex instance is an instance that is presented via several other instances, for example a device may be presented via GPIs, several of these devices may be represented on a single GPI device, and a device may be spread across more than one GPI device. Provision is also given for devices that may have some parameters from GPIs, for instance a PSU may be via GPI but all other parameters via a normal control link.

To represent a complex instance the device type will the parameter slot set to zero, this indicated that the ref in instances needs to use the complex target to locate the real instance. The complex reference tells the client to look at a specific file for the instance information, this file contains details of each parameter for the selected instance, these details are the instance and parameter of the real presenting item.

1.7 Example - Complex instance

Device description

```
<param name="comms" slot="49" class="enum" style="button" access="readonly">
	<state value="0" caption="OK" style="" msg=""/>
	<state value="1" caption="Fail" style="" msg=""/>
</param>
<param name="PSU " slot="0" class="enum" style="label" access="readonly">
	<state value="0" caption="Fail" style="alarm_alarm" msg=""/>
	<state value="1" caption="OK" style="alarm_ok" msg=""/>
</param>
```

This device has 2 parameters, the comms state is from a standard driver, slot 49, it also has a PSU status from a separate device.

Instances

```
	<instance id="gpi_crate_1" type="CC_GPI96" ref="device=124,offset=0"
location="13B4.20.9" composite="no"/>

	<instance id=" arc1" type="complex_dev"
ref="device=123,offset=1000,complex=complex_dev_instances" location=""
composite="no"/>

	<instance id=" arc2" type="complex_dev"
ref="device=123,offset=1100,complex=complex_dev_instances" location=""
composite="no"/>
```

The instance file shown shows 3 instances, the gpi_crate_1 is a standard instance, while the arcs are complex instances, with the detail in "complex_instances\complex_dev_instances.xml"

Complex instance file

```
<instance id="arc1">
	<param id="PSU " device="gpi_crate_1" param="GPI Input 03"/>
</instance>
<instance id="arc2">
	<param id="PSU " device="gpi_crate_1" param="GPI Input 06"/>
</instance>
```

This file shows the details for the arc instances, in this case the only parameter that is via a separate device is the GPI, the real device and parameter for this parameter are detailed here, and require further lookups from instances and the device type to get the real device and slot numbers.