CERN
CH-1211 Geneva 23
Switzerland

the
Large
Hadron
Collider
project

LHC Project Document No.
EN-UNICOS

CERN Div./Group or Supplier/Contractor Document No.
EN / ICE

EDMS Document No.

Date: 2010-03-23

FUNCTIONAL SPECIFICATION

UNICOS PVSS ADDING FRONT-END ADDING DEVICE PROCEDURE

Abstract

This document describes the procedure to a front-end or a device...

Prepared by:	***Checked by:***	***Approved by:***
UNICOS core team EN/ICE		

History of Changes

Rev. No.	*Date*	*Pages*	*Description of Changes*
1.1 Draft	23-March-2010		First version

Table of Contents

1. INTRODUCTION

1.1 PURPOSE OF THIS DOCUMENT

UNICOS stands for **UN**ified **I**ndustrial **CO**ntrol **S**ystem. The supervision device hierarchy is based on a front-end device containing process devices (time stamping in the front-end is allowed). A front-end device is an entity representing a piece of hardware or software entity holding devices. A front-end is also the entity through which the devices are accessible. Process device or device represents a piece of hardware sensor or software entity. A device is attached to one and only one front-end. Typical examples of front-end devices are PLCs, Front-End Computers (FEC), OPC servers, ELMB; while process devices are Analog Input, Digital Input, ELMB channel, etc. Front-end device and process device can be with or without hardware link.

This document describes the internal behavior and data flow of the device and front-end device.

1.2 DEFINITIONS, ACRONYMS AND ABBREVIATIONS

From this point onwards, the following acronyms will be used:

- DPT: PVSS data point type.
- DP: PVSS data point, instance of the DPT.
- DPE: PVSS data point element.
- Device: instance of the device type.
- Front-end: instance of the front-end device type
- Alarm: PVSS alert config on a DPE.

1.3 CONTACT AT CERN

Any problem during installation or development can be reported to the following email address: UNICOS.Support@cern.ch.

2. NAMING CONVENTION

In the following sections the examples are based on the unPackageTemplate component. The following keywords will be used:

- TAG_prefix: prefix of the package
- TAG_PREFIX: TAG_prefix in upper case letter
- TAG_FrontEndType: the front-end device type
- TAG_FRONTENDTYPE: TAG_FrontEndType in upper case letter
- TAG_package: the package name
- TAG_PACKAGE: TAG_package in upper case letter
- TAG_DeviceType: the device type
- TAG_DEVICETYPE: TAG_DeviceType in upper case letter
- TAG_WidgetType: the widget name type

3. UNICOS DEFINITION

- Component: software entity, e.g.: systemIntegrity, unCore,
- Package (Figure 1): a Package is a set of component combined and configured together, it extends UNCORE to a specific domain, for intance: Device component, utilities, etc. E.g.: unCPC package, QPS package.
- Application (Figure 2): set of package combined and configured together to produce a control and/or monitoring application. E.g.: LHC Cryogenics and LHC GCS with the unCPC package.

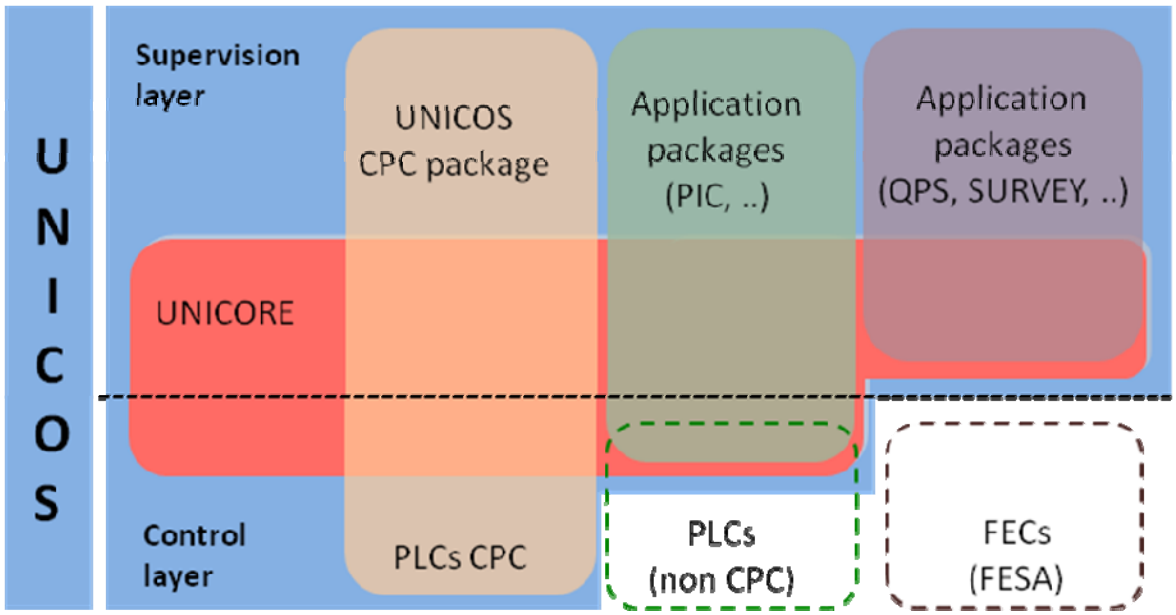

Figure 1: UNICOS Framework and package.

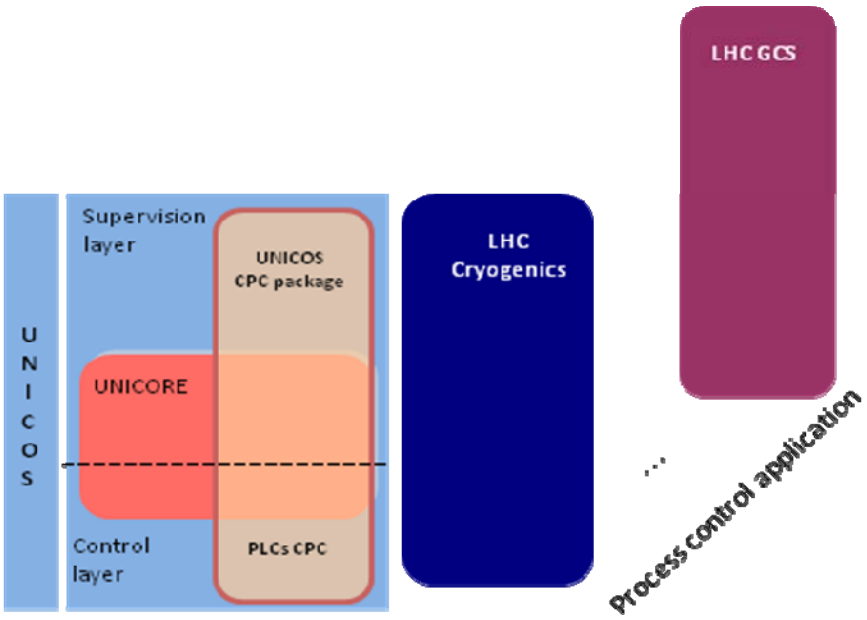

Figure 2: UNICOS Framework application.

4. UNICOS DEVELOPMENT

There are different types of development:

- On a device of a package:
  - Add new widget to the list of widget of a device: section 5.1.
  - Import/export the device with a front-end type of another package: section 5.2.
  - Any other modifications of device behavior must be discussed and agreed with the responsible of the package, section 5.3.
- On a front-end of a package:
  - Any modifications of the front-end behavior must be discussed and agreed with the responsible of the package, section 6.
- Add a new front-end: section 7.1.
- Add a new device: section 7.2.

The following documents are available:

- unicos-pvss-adding-front-end_and_device.pdf: front-end and device concept, detailed description of the data flow.
- unicos-pvss-framework.pdf: description of the UNICOS framework and concept.
- device-with-UNICOS.pdf: proxy definition and usage

An xml file for the package must be created: section 9.

The examples below are based on the unPacakgeTemplate component.

5. DEVICE DEVELOPMENT

5.1 ADD NEW WIDGET TO THE LIST OF WIDGET OF A DEVICE

For a given package, new device widget can be added:

1. Check with the package developer if the device widget function accepts new widget and what are the widget extentions that can be done.
2. Create the widget and all the associated files. Add the widget in the device package catalog.
3. Add this widget into the list of possible widget for this device
4. If in the new widget, the list of shapes to animate is different from the list of shapes of the existing widget, add the multi-widget function animation and disconnection in the device library.
5. Test the widget animation and disconnection.
6. Package this new development with the existing device package.

Example:

1. Copy panels/objects/TAG_PACKAGE/TAG_prefixWidget_TAG_DeviceTypeTAG_WidgetType.pnl and images/objects/TAG_PACKAGE/TAG_prefixWidget_TAG_DeviceTypeTAG_WidgetType.pnl/ TAG_prefixWidget_TAG_DeviceTypeTAG_WidgetType.png
2. If in the new widget, the list of shapes to animate is different from the list of shapes of the existing widget, replace TAG_WidgetType by your new widget name and copy the functions:

TAG_prefixTAG_DeviceType_TAG_WidgetTypeAnimation and TAG_prefixTAG_DeviceType_TAG_WidgetTypeDisconnection

3. If the new widget has the same shapes, then in the widget just use TAG_WidgetType of another widget.

5.2 IMPORT/EXPORT THE DEVICE WITH A FRONT-END TYPE OF ANOTHER PACKAGE

Existing device can be used with different front-end:

1. Check with the package developer if the device type can be with different front-end
2. Define the type of configuration for the device; it can be different than existing ones. Take care to the device DPE: they must be set or configured correctly in order to have the correct functioning of the widget, faceplate, etc. animation.
3. Add the import/export functions in the device library
4. Test the import/export in all the possible cases
5. Package this new development with the existing device package.

Example:

1. Copy the functions: TAG_FrontEndType_TAG_DeviceType_checkConfig and TAG_FrontEndType_TAG_DeviceType_setConfig
2. Replace TAG_FrontEndType by the front-end type name
3. Copy the function: TAG_FrontEndType_TAG_DeviceType_ExportConfig and replace TAG_FrontEndType by the front-end type name

5.3 OTHER DEVELOPMENT

The device behavior can be entirely modified. These modifications must be done with the agreement of the package responsible. Here is the list of non exhaustive modifications:

- Existing widget: modify the look and feel, add new device DPE, etc. this must be done in all the widgets and in the widget functions.
- Faceplate: modify the look and feel, add new device DPE, etc. this must be done in all the faceplate and in the faceplate functions.
- Device action: add a new action, modify the default action access rights, etc. this must be done in the device action interface and device action libs.
- Right click: add a new menu, modify the existing one, etc. this must be done in device libs
- Snaphot: modify the returned data. This must be done in the device action interface and device action libs.

6. FRONT-END DEVELOPMENT

The front-end behavior can be entirely modified. These modifications must be done with the agreement of the package responsible. Here is the list of non exhaustive modifications:

- Widget: modify the look and feel, add new front-end DPE, etc. this must be done in all the widgets and in the widget functions.
- Faceplate: modify the look and feel, add new front-end DPE, etc. this must be done in all the faceplate and in the faceplate functions.

- Front-end diagnostic: modify the look and feel, add new front-end DPE, add new front-end actions, etc.
- Front-end systemIntegrity: add new systemAlarm, add new option, new configuration data, modify the counter checking, etc.
- Right click: add a new menu, modify the existing one, etc. this must be done in front-end libs

7. PACKAGE DEVELOPMENT

7.1 ADD A NEW FRONT-END

To create a new front-end type:

1. Create the front-end PVSS DPType, with the necessary DPE: unicos-pvss-unCore-data-flow.pdf, section 4.1.1.
2. Set the front-end device definition: unicos-pvss-unCore-data-flow.pdf, section 4.2.
3. Create the front-end check/delete/import functions: unicos-pvss-unCore-data-flow.pdf, section 5.
4. Create the front-end export functions: unicos-pvss-unCore-data-flow.pdf, section 6.
5. Create the front-end diagnostic panel: unicos-pvss-unCore-data-flow.pdf, section 7.2 and 7.3.
6. Create the front-end device functions: unicos-pvss-unCore-data-flow.pdf, section 8.
7. Create the front-end icons for the tree device overview if needed: unicos-pvss-unCore-data-flow.pdf, section 8.5.
8. Create the widget, catalog of widget, widget param: unicos-pvss-unCore-data-flow.pdf, section 8.2.
9. Create the faceplate: unicos-pvss-unCore-data-flow.pdf, section 8.3.
10. Configure the trending: unicos-pvss-unCore-data-flow.pdf, section 4.3.2 and 8.3.5
11. Create the front-end systemIntegrity: unicos-pvss-unCore-data-flow.pdf, section 4.4 and 7.1.
12. Test import, export, widget, faceplate, right-click, snapshot, etc. in local and distributed environment.

7.2 ADD A NEW DEVICE

To create a new device type:

1. Create the device PVSS DPType, with the necessary DPE: unicos-pvss-unCore-data-flow.pdf, section 4.1.2.
2. Set the device definition: unicos-pvss-unCore-data-flow.pdf, section 4.2.
3. Set the JCOP device definition: unicos-pvss-unCore-data-flow.pdf, section 5.1.
4. Create the device check/import functions: unicos-pvss-unCore-data-flow.pdf, section 5.
5. Create the device export functions: unicos-pvss-unCore-data-flow.pdf, section 6.
6. Create the device functions: unicos-pvss-unCore-data-flow.pdf, section 8.

7. Create the device icons for the tree device overview if needed: unicos-pvss-unCore-data-flow.pdf, section 8.5.
8. Create the widget, catalog of widget, widget param: unicos-pvss-unCore-data-flow.pdf, section 8.2.
9. Create the faceplate: unicos-pvss-unCore-data-flow.pdf, section 8.3.
10. Configure the trending: unicos-pvss-unCore-data-flow.pdf, section 4.3.2 and 8.3.5
11. Create the device action interface, select, acknowledge: unicos-pvss-unCore-data-flow.pdf, section 9.
12. Test import, export, widget, faceplate, right-click, snapshot, etc. in local and distributed environment.

8. UTILITY DEVELOPMENT

New utilities can be created for the UNICOS. Contact the UNICOS support for more detail.

9. PACKAGING

The development must be packed into an xml file. This package will be installed via the JCOP installation tool. The files to add are:

- .xml: the xml file containing all the files
- .dpl: the dpl file containing the DPEtype, DP and DPE definition; e.g.: front-end and device configuration, settings, trending, etc. Care must be taken to the settings of other packages that have to be overwritten (this setting will be overwritten during the installation of the original package)
- .config: list of libs to add in the config file
- Any dependencies to other package or component
- The date and version
- .init: panel or script launched during the installation. For instance if a confirmation has to be requested from the user for setting a configuration
- .postInstall: post install script to set the configuration, care must be taken to the setting that can be overwritten by other package. The windows and linux OWS config file are usually created during this step.
- Widget: all the widget file, .png file, param panels
- Device and front-end Faceplate, faceplate status, device action button, device trend panel
- Front-end diagnostic panel
- Libs: device library, front-end library
- colorDB file, .cat (catalog file)
- test panels
- front-end and device type bmp file for the tree device overview
- any utilities: scripts, panels, libs, etc.

This package xml file must be tested on a new project and on an existing one.

LIST OF FIGURES

REFERENCE

1. unicos-pvss-framework.pdf
2. device-with-UNICOS.pdf
3. unicos-systemIntegrity-device-type.pdf
4. unicos-pvss-unCore-data-flow.pdf