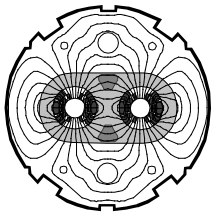
CERN
CH-1211 Geneva 23
Switzerland



the
Large
Hadron
Collider
project

LHC Project Document No.
UNICOS-PVSS

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

EDMS Document No.

Date: 2010-04-22

FUNCTIONAL SPECIFICATION

UNICOS HMI CONFIGURATION

Abstract

Describe the HMI configuration.

Prepared by:	***Checked by:***	***Approved by:***
AB/CO		

History of Changes

Rev. No.	*Date*	*Pages*	*Description of Changes*
1.1	28-10-2008		First version.
1.2	19-01-2009		2 new widgets: unDiagFEButton & unDeviceTreeOverview
1.3	22-04-2010		New HMI component widget: unCanvas.pnl

Table of Contents

1. INTRODUCTION

1.1 PURPOSE OF THIS DOCUMENT

- Describe how to configure the HMI
- Brief explanation of the HMI
- How to create new component and a new HMI

1.2 DEFINITIONS, ACRONYMS AND ABBREVIATIONS

-

1.3 PRE-REQUISITE

PVSS 3.6 + Latest patch + unicos-pvss-4.2 must be installed.

2. UNICOS BASED HMI

The unicos HMI from version 4.2 onwards is entirely customizable by mean of an xml file describing the content of the HMI. A set of graphical components can be used to build the HMI look and feel. New components can also be created.

The xml configuration file must be in the 'data' folder of any project path of the project.

2.1 XML CONFIGURATION FILE

The format of the xml HMI configuration is:

```
<HMI>
        <properties>
        ...
        </properties>
        <component>
        ...
        </component>
        ...
</HMI>
...
<HMI>
        <properties>
        ...
        </properties>
        <component>
        ...
        </component>
        ...
</HMI>
```

- One <HMI> section per screen number (PVSS module)
- As many <HMI> as needed, for example 3 screens contains 3 <HMI> sections
- In each <HMI>
  - <properties> of the HMI: filename, size x and size y, screen number, x and y position, libs, HMI functions and name. The HMI functions are not mandatory, if there are present, they must be for the screen number 1 to be executed.
  - List of components <component> section: filename and <properties> section composed of name, x and y position, size x and size y, rotation angle and dollar parameters.

2.2 HMI EXECUTION FLOW

The HMI panel file name must be a copy of 'vision/graphicalFrame/unicosHMI_1280x996.pnl'. To start a HMI, one must create a panel that set the global variable 'g_s_unicosHMI_previewConfig' to the xml config file name relative to one of the project path (e.g.: 'data/unicosHMI_2screen.xml') and then call the unGraphicalFrame_loadHMI function. The panel 'vision/graphicalFrame/unicosHMI.pnl' is one example. The startup script is:

```
main()
{
  string sFile;
  dyn_string exInfo, ds;
  dyn_float df;

  sFile = UNICOSHMI_1SCREEN_CONFIGURATION_FILE;
  g_s_unicosHMI_previewConfig = sFile;
  unGraphicalFrame_loadHMI(exInfo);
  if(dynlen(exInfo) > 0) {
    fwExceptionHandling_display(exInfo);
    unGraphicalFrame_ChildPanelOnCentralReturn("vision/MessageWarning", "ERROR",
                          makeDynString("$1:Create "+g_s_unicosHMI_previewConfig+" and restart!"),
                          df, ds);
  }
  g_modules_opened = true;
  ModuleOff(myModuleName());
}
```

The function unGraphicalFrame_loadHMI will open the HMI filename in a new Module named name_screen (where name is the entry value of <name> and screen is the entry value of <screen>).

The data flow of HMI panel is:

- Load the xml file
- Check if there is any bad initialization
- Init the HMI

The HMI panel will execute the HMI functions (if there are any) only and only if screen is equal to '1'. Three global bool variables can be used to get the current state of the init phase:

- g_b_unicosHMI_init1 set to true after the HMI init function and before the HMI tree register function

- g_b_unicosHMI_initEnd set to true at the end of the HMI tree complete initialization. From that point it is considered that the list of devices is completely initialized.
- g_b_unicosHMI_menu set to true at the end of the menu configuration initialization

Usually all the graphical element of HMI component are disabled (button, list, etc.) and enabled when g_b_unicosHMI_initEnd is set to true. A script like the one that follows can be used to wait the completion of the init phase:

```
while(!g_b_unicosHMI_initEnd)
  delay(1);
```

2.3 XML CONFIGRATION FILE: DETAILED DESCRIPTION

2.3.1 <PROPERTIES> OF THE <HMI>

The mandatory fields are:

- <filename>: the panel file name that will be opened
- <sizeX>: the size x of the panel file name, the value must be the same as the one in the panel file name
- <sizeY>: the size y of the panel file name, the value must be the same as the one in the panel file name
- <screen>: screen number between 0 to 99
- <name>: the name of the panel opened, name of the panel in the module

The optional fields are:

- <x>: the x postion on which the panel will be opened, 0 is the default if the the entry is not present
- <y>: the y postion on which the panel will be opened, 0 is the default if the the entry is not present
- <libs>: libs to load, there is no limitation in the number of libs, any errors in the libs will be thrown on the log, there is no checking
- <initFunction>: init function, initialization of variable, etc. this function must return true in case of success and false in case of failure, this function is executed by an evalScript therefore any dpConnect type call and its callback will end. If the result is false the HMI will stop its execution, raise an exception and exit.
- <applicationUpdateFunction>: callback function triggered whenever the device list is updated, callback of a dpConnect call.
- <registerFunction>: register function, this function is used to register to the different system, it is usually to initialize the device list global variables. This function is started by a startThread call.
- <waitRegisterFunction>: end initialization function, it is usually used to initialize the global variables after the initialization. This function is triggered when all the register functions have been initialized and executed. This function is executed by an evalScript therefore any dpConnect type call and its callback will end.

Example of HMI properties with HMI functions:

```
<properties>
        <filename>vision/graphicalFrame/unicosHMI_1274x996.pnl</filename>
        <sizeX>1274</sizeX>
        <sizeY>996</sizeY>
        <screen>1</screen>
        <x>0</x>
```

```
        <y>0</y>
        <libs>unicosHMI.ctl</libs>
        <initFunction>unicosHMI_init</initFunction>
        <applicationUpdateFunction>unicosHMI_applicationUpdateDeviceList</applicationUpdateFunction>
        <registerFunction>unicosHMI_registerTreeDeviceOverview</registerFunction>
        <waitRegisterFunction>unicosHMI_waitTreeDeviceOverviewRegistration</waitRegisterFunction>
        <name>unicosHMI</name>
</properties>
```

Example of HMI properties without HMI functions:

```
<properties>
        <filename>vision/graphicalFrame/unicosHMI_1274x996.pnl</filename>
        <sizeX>1274</sizeX>
        <sizeY>996</sizeY>
        <screen>2</screen>
        <x>1280</x>
        <y>0</y>
        <name>unicosHMI</name>
</properties>
```

2.3.2 <COMPONENT>

The mandatory fields are:

- <filename>: the panel file name that will be added with addSymbol
- <properties> section:
  - the mandatory fields are:
    - <name>: the name of the symbol to be added
  - The optional fields are:
    - <x>: the x position in the HMI panel, 0 is the default if the the entry is not present
    - <y>: the y position in the HMI panel, 0 is the default if the the entry is not present
    - <sizeX>: the desired x size, the component will be scaled based on the ratio sizeX / size x of the panel, 0 is the default if the the entry is not present and this means no scaling. The scaling is based on a calculation therefore the size may be greater than the HMI panel size if one wants to fit perfectly the component into the HMI. For instance a component panel of size 120 containting a button of size 27, one has to put 240 to get the button twice bigger in x and not 52.
    - <sizeY>: the desired y size, the component will be scaled based on the ratio sizeY / size y of the panel, 0 is the default if the the entry is not present and this means no scaling. The scaling is based on a calculation therefore the size may be greater than the HMI panel size if one wants to fit perfectly the component into the HMI. For instance a component panel of size 80 containting a button of size 27, one has to put 160 to get the button twice bigger in y and not 52.
    - <angle>: the rotation angle, the component will be added and rotated
    - <dollar>: dollar parameter and dollar parameter value as it is in a PVSS panel, $dollarParam:dollarValue. One <dollar> entry per dollar parameter of the component panel.

Example of a component with size:

```
<component>
        <filename>objects/UN_GRAPHICALFRAME/unArea.pnl</filename>
        <properties>
                <name>header</name>
                <x>1</x>
                <y>1</y>
                <sizeX>1276</sizeX>
                <sizeY>91</sizeY>
        </properties>
</component>
```

Example of a component without size:

```
<component>
        <filename>objects/UN_GRAPHICALFRAME/unUnicosIconExtended.pnl</filename>
        <properties>
                <name>Icon</name>
                <x>3</x>
                <y>24</y>
        </properties>
</component>
```

Example of a component with angle:

```
<component>
        <filename>objects/UN_GRAPHICALFRAME/unNavigationPopup.pnl</filename>
        <properties>
                <name>NavigationPopup</name>
                <x>20</x>
                <y>20</y>
                <angle>45</angle>
        </properties>
</component>
```

Example of a component with dollar parameter:

```
<component>
        <filename>objects/UN_GRAPHICALFRAME/unCascadeMenu.pnl</filename>
        <properties>
                <name>CascadeManagement</name>
                <x>920</x>
                <y>30</y>
                <dollar>$MENU_ITEM:Management</dollar>
        </properties>
</component>
```

2.4 HMI CONFIGURATION UTILITY

One does need to start the HMI interface to test if the properties of the components are correctly positioned. One can use from the gedi module or from a UI module the panel vision/graphicalFrame/unicosHMI_configuration.pnl (Figure 1).

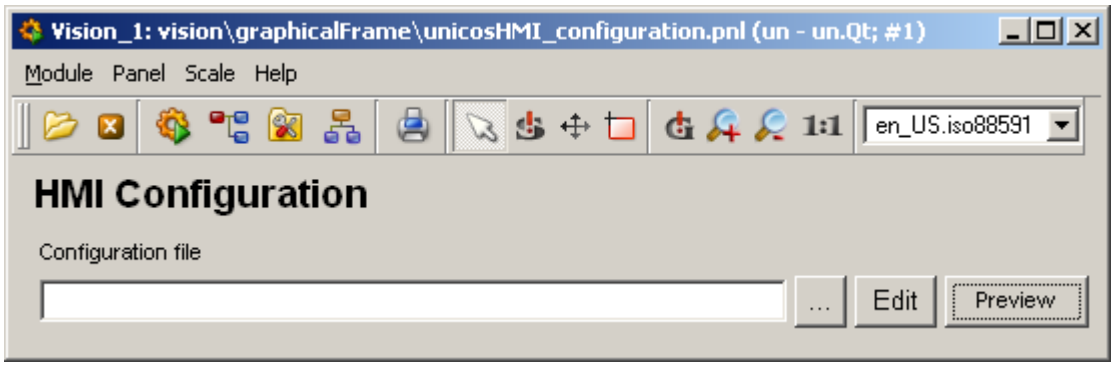


Figure 1: unicosHMI_configuration

2.5 UNICOS HMI COMPONENTS

The lists of components provided within the unicos-pvss package are the following:

- Filename (Figure 2): objects/UN_GRAPHICALFRAME/unAlarmListButton.pnl
  - Panel size x: 120
  - Panel size y: 28
  - Graphical element: button
    - Size x: 27
    - Size y: 27
  - Purpose: open the alarm list, the previous used configuration (filter, etc.) is kept in the button variables and given to the alarm list panel.

Figure 2: unAlarmListButton.pnl

- Filename (Figure 3): objects/UN_GRAPHICALFRAME/unAlarmRow.pnl
  - Panel size x: 1199
  - Panel size y: 35
  - Purpose: alarm row. The alarm row is hidden or shown based on the 'show alert row' configuration of the unicos-pvss based application. A click on the left button hides or shows the alarm row. The 'domain' and 'nature' fields are not shown; they can be activated by a right click.

Figure 3: unAlarmRow.pnl

- Filename (Figure 4): objects/UN_GRAPHICALFRAME/unAlarmRowExtended.pnl
  - Panel size x: 1199
  - Panel size y: 35
  - Purpose: extended alarm row, all the fields are shown.

Figure 4: unAlarmRowExtended.pnl

- Filename (Figure 5): objects/UN_GRAPHICALFRAME/unAlarmRowReduced.pnl
  - Panel size x: 1199
  - Panel size y: 35
  - Purpose: reduced alarm row, same as above but without the 'domain' and 'nature' fields, they can be activated by a right click

Figure 5: unAlarmRowReduced.pnl

- Filename (Figure 6): objects/UN_GRAPHICALFRAME/unApplicationName.pnl
  - Panel size x: 300
  - Panel size y: 30
  - Graphical element: text field
    - Size x: 175
    - Size y: 30
  - Purpose: show the application name

Figure 6: unApplicationName.pnl

- Filename (Figure 7): objects/UN_GRAPHICALFRAME/unArea.pnl
  - Panel size x: 200
  - Panel size y: 100
  - Graphical element: rectangle
    - Size x: 200
    - Size y: 100
  - Purpose: this component can be used to draw a header and footer area with a separator.

Figure 7: unArea.pnl

- Filename (Figure 8): objects/UN_GRAPHICALFRAME/unBeep.pnl

- Panel size x: 120
- Panel size y: 45
- Graphical element: rectangle
  - Size x: 44
  - Size y: 44
- Graphical element: text field1
  - Size x: 29
  - Size y: 20
- Graphical element: text field2
  - Size x: 29
  - Size y: 20
- Purpose: to display the beep state configured for the current application. The run-time result can be like in Figure 9.

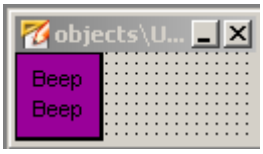


Figure 8: unBeep.pnl

Figure 9: unBeep.pnl at run-time

- Filename (Figure 10): objects/UN_GRAPHICALFRAME/unCanvas.pnl
  - Panel size x: 120
  - Panel size y: 120
  - Graphical element: Canvas
    - Size x: 120
    - Size y: 120
  - dollar:
    - $sName: the canvas module name
    - $sPanelFileName: the panel file name to show
    - $sDollarParameter: the $-param of the panel $sPanelFileName. The $-param are encapsulated by “”, each $-param is separated by ‘,’, there is no limitation in the type of characters, ‘ ’ are allowed. The $-param will be given as it is to the panel. E.g. of $-param:

      $sDollarParameter:"$d1:ho $la","$d2:d2 , hola","$d3:d3hola"
  - Purpose: show a panel in the Canvas module with or without dollar parameters.

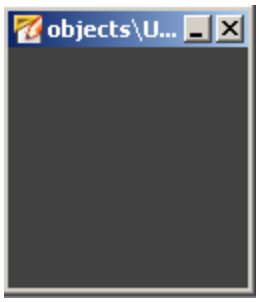


Figure 10: unCanvas.pnl

- Filename (Figure 11): objects/UN_GRAPHICALFRAME/unCascadeMenu.pnl
  - Panel size x: 120
  - Panel size y: 17
  - Graphical element: cascade button
    - Size x: 100
    - Size y: 17
  - dollar:
    - $MENU_ITEM: shape name as the one configured in the menu configuration utility, "" means all the shape names
  - Purpose: Access the configured menu item, the run-time result can be like in Figure 12.

Figure 11: unCascadeMenu.pnl

Figure 12: unCascadeMenu.pnl at run-time

- Filename (Figure 13): objects/UN_GRAPHICALFRAME/unContextualButton.pnl
  - Panel size x: 459
  - Panel size y: 53
  - Graphical element: canvas
    - Size x: 459
    - Size y: 53
  - Purpose: show the contextual button.
  - Constraint: only one component per module.

Figure 13: unContextualButton.pnl

- Filename (Figure 14): objects/UN_GRAPHICALFRAME/unCurrentUser_text_time.pnl
  - Panel size x: 144
  - Panel size y: 53
  - Purpose: show the current user log in and current time, login window, access control utility

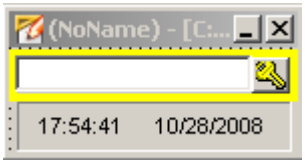


Figure 14: unCurrentUser_text_time.pnl

- Filename (Figure 15): objects/UN_GRAPHICALFRAME/unDeviceOverview.pnl
  - Panel size x: 120
  - Panel size y: 28
  - Graphical element: button
    - Size x: 27
    - Size y: 27
  - Purpose: open the device overview or device tree overview, the unUserPanel.pnl or unUserPanelWithPanel.pnl must be in HMI to be able to show the device overview or device tree overview

Figure 15: unDeviceOverview.pnl

- Filename (Figure 16): objects/UN_GRAPHICALFRAME/ unDeviceTreeOverview.pnl
  - Panel size x: 120
  - Panel size y: 28
  - Graphical element: button
    - Size x: 27
    - Size y: 27
  - Purpose: open the device tree overview, the unUserPanel.pnl or unUserPanelWithPanel.pnl must be in HMI to be able to show the device tree overview

Figure 16: unDeviceTreeOverview.pnl

- Filename (Figure 17): objects/UN_GRAPHICALFRAME/unDiagFEButton.pnl

- Panel size x: 120
- Panel size y: 28
- Graphical element: button
  - Size x: 27
  - Size y: 27
- Purpose: open the Front-End diagnostic panel

Figure 17: unDiagFEButton.pnl

- Filename (Figure 18): objects/UN_GRAPHICALFRAME/unEventList.pnl
  - Panel size x: 120
  - Panel size y: 28
  - Graphical element: button
    - Size x: 27
    - Size y: 27
  - Purpose: open the event list, the previous used configuration (filter, etc.) is kept in the button variables and given to the event list panel

Figure 18: unEventList.pnl

- Filename (Figure 19): objects/UN_GRAPHICALFRAME/unInfoText.pnl
  - Panel size x: 363
  - Panel size y: 28
  - Graphical element: text field
    - Size x: 363
    - Size y: 28
  - Purpose: display information text with the functions unGraphicalFrame_setLoadingText, unGraphicalFrame_setVisibleLoadingText. The run-time result can be like in Figure 20.
  - Constraint: only one component per module.

Figure 19: unInfoText.pnl

Figure 20: unInfoText.pnl at run-time

- Filename (Figure 21): objects/UN_GRAPHICALFRAME/unMenu.pnl
  - Panel size x: 120
  - Panel size y: 28
  - Graphical element: text field
    - Size x: 31
    - Size y: 20
  - Purpose: show all the configured menu of the menu configuration utility like any WindowsXP menu. The run time result can be like in Figure 22. The menu text is automatically resized to be adjusted to the shape name size.

Figure 21: unMenu.pnl

Figure 22: unMenu.pnl at run-time.

- Filename (Figure 23): objects/UN_GRAPHICALFRAME/unMenuButton.pnl
  - Panel size x: 120
  - Panel size y: 50
  - Graphical element: button
    - Size x: 120
    - Size y: 50
  - dollar:
    - $MENU_ITEM: shape name as the one configured in the menu configuration utility, "" means all the shape names
  - Purpose: Access the configured menu item, the run-time result can be like in Figure 24.

Figure 23: unMenuButton.pnl

Figure 24: unMenuButton.pnl at run-time

- Filename (Figure 25): objects/UN_GRAPHICALFRAME/unMessageText.pnl
  - Panel size x: 390
  - Panel size y: 35

- Graphical element: text field
  - Size x: 11
  - Size y: 41
- Graphical element: table
  - Size x: 381
  - Size y: 36
- Purpose: display the last messages, open the message text history log, the previous used configuration (filter, etc.) is kept in the button variables and given to the message text history log

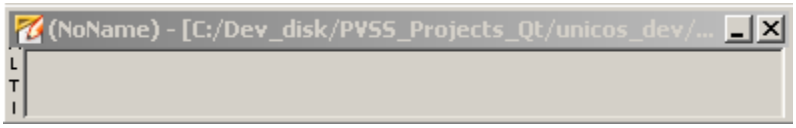


Figure 25: unMessageText.pnl

- Filename (Figure 26): objects/UN_GRAPHICALFRAME/ unMessageTextButton.pnl
  - Panel size x: 120
  - Panel size y: 50
  - Graphical element: button
    - Size x: 120
    - Size y: 50
  - Purpose: open the message text history log, the previous used configuration (filter, etc.) is kept in the button variables and given to the message text history log

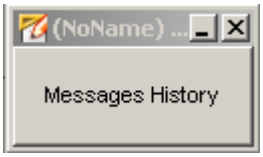


Figure 26: unMessageTextButton.pnl

- Filename (Figure 27): objects/UN_GRAPHICALFRAME/unNavigationPopup.pnl
  - Panel size x: 120
  - Panel size y: 28
  - Graphical element: button
    - Size x: 27
    - Size y: 27
  - Purpose: show the navigaton panel configured in the application

Figure 27: unNavigationPopup.pnl

- Filename (Figure 28): objects/UN_GRAPHICALFRAME/unNavigatorHistory.pnl

- Panel size x: 640
- Panel size y: 36
- Purpose: internet explorer like navigation, previous, next, current, list of accessed view, favorites, home, refresh

Figure 28: unNavigatorHistory.pnl

- Filename (Figure 29): objects/UN_GRAPHICALFRAME/unObjectListButton.pnl
  - Panel size x: 120
  - Panel size y: 28
  - Graphical element: button
    - Size x: 27
    - Size y: 27
  - Purpose: open the object list, the previous used configuration (filter, etc.) is kept in the button variables and given to the object list panel

Figure 29: unObjectListButton.pnl

- Filename (Figure 30): objects/UN_GRAPHICALFRAME/unPrintButton.pnl
  - Panel size x: 120
  - Panel size y: 28
  - Graphical element: button
    - Size x: 27
    - Size y: 27
  - Purpose: print

Figure 30: unPrintButton.pnl

- Filename (Figure 31): objects/UN_GRAPHICALFRAME/unSelectDevice.pnl
  - Panel size x: 490
  - Panel size y: 20
  - Purpose: general purpose select utility. The run time result can be like in Figure 32.
  - Constraint: only one component per module.

Figure 31: unSelectDevice.pnl

Figure 32: unSelectDevice.pnl at run-time

- Filename (Figure 33): objects/UN_GRAPHICALFRAME/unSelectRemaining.pnl
  - Panel size x: 250
  - Panel size y: 10
  - Purpose: show the remaining time before the automatic deselect of a device

Figure 33: unSelectRemaining.pnl

- Filename (Figure 34): objects/UN_GRAPHICALFRAME/unSeparatorDoubleVertical.pnl
  - Panel size x: 120
  - Panel size y: 28
  - Graphical element: 2 ThumbWheel
    - Size x: 2
    - Size y: 27
  - Purpose: separator vertical. The run-time result can be like in Figure 35.

Figure 34: unSeparatorDoubleVertical.pnl

Figure 35: unSeparatorDoubleVertical.pnl at run-time

- Filename (Figure 36): objects/UN_GRAPHICALFRAME/unSeparatorHorizontal.pnl
  - Panel size x: 200
  - Panel size y: 28
  - Graphical element: ThumbWheel
    - Size x: 200
    - Size y: 1
  - Purpose: separator horizontal. The run-time result can be like in Figure 37.

Figure 36: unSeparatorHorizontal.pnl

Figure 37: unSeparatorHorizontal.pnl at run-time

- Filename (Figure 38): objects/UN_GRAPHICALFRAME/unSeparatorVertical.pnl
  - Panel size x: 120
  - Panel size y: 28
  - Graphical element: ThumbWheel
    - Size x: 2
    - Size y: 27
  - Purpose: separator vertical. The run-time result can be like in Figure 39.

Figure 38: unSeparatorVertical.pnl

Figure 39: unSeparatorVertical.pnl at run-time

- Filename (Figure 40): objects/UN_GRAPHICALFRAME/unSystemStatus.pnl
  - Panel size x: 120
  - Panel size y: 45
  - Graphical element: rectangle
    - Size x: 44
    - Size y: 44
  - Graphical element: text field1
    - Size x: 29
    - Size y: 20
  - Graphical element: text field2
    - Size x: 29
    - Size y: 20
  - Purpose: show the system status of the application. The run-time result can be like in Figure 41.

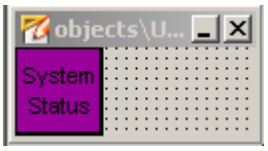


Figure 40: unSystemStatus.pnl

Figure 41: unSystemStatus.pnl at run-time

- Filename (Figure 42): objects/UN_GRAPHICALFRAME/unTrendTree.pnl
  - Panel size x: 120
  - Panel size y: 28
  - Graphical element: button
    - Size x: 27
    - Size y: 27
  - Purpose: open the trend tree, the unUserPanel.pnl or unUserPanelWithPanel.pnl must be in HMI to be able to show a trend

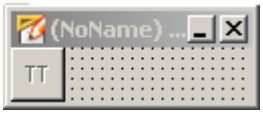


Figure 42: unTrendTree.pnl

- Filename (Figure 43): objects/UN_GRAPHICALFRAME/unUnicosIcon.pnl
  - Panel size x: 120
  - Panel size y: 65
  - Graphical element: rectangle
    - Size x: 65
    - Size y: 65
  - Purpose: show the icon of the application. The run-time result can be like in Figure 44.

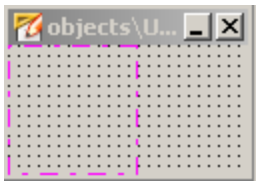


Figure 43: unUnicosIcon.pnl

Figure 44: unUnicosIcon.pnl at run-time

- Filename (Figure 45): objects/UN_GRAPHICALFRAME/unUnicosIconExtended.pnl
  - Panel size x: 120
  - Panel size y: 65
  - Graphical element: rectangle
    - Size x: 65
    - Size y: 65
  - Purpose: show the icon of the application, access to all the configured menu of the application by clicking on the icon. The run-time result can be like in Figure 46.

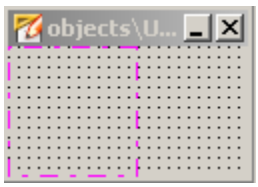


Figure 45: unUnicosIconExtended.pnl

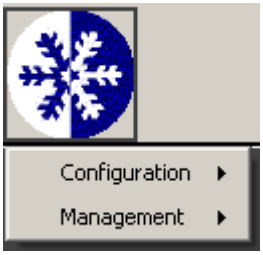


Figure 46: unUnicosIconExtended.pnl at run-time

- Filename (Figure 47): objects/UN_GRAPHICALFRAME/unUserPanel.pnl
  - Panel size x: 200
  - Panel size y: 200
  - Graphical element: canvas
    - Size x: 200
    - Size y: 200
  - Purpose: user panel area to be used with the navigation history, trend tree, widnow tree, contextual button
  - Constraint: only one component per module and no unUserPanelWithPanel.pnl in the same module

Figure 47: unUserPanel.pnl

- Filename (Figure 48): objects/UN_GRAPHICALFRAME/unUserPanelWithPanel.pnl
  - Panel size x: 200
  - Panel size y: 200
  - Graphical element: canvas
    - Size x: 200
    - Size y: 200
  - dollar:
    - $sPanel: panel name, plot, page, reference to the window tree to be opened, same format as the one for the default panel set in the application configuration
  - Purpose: user panel area to be used with the navigation history, trend tree, widnow tree, contextual button
  - Constraint: only one component per module and no unUserPanel.pnl in the same module

Figure 48: unUserPanelWithPanel.pnl

- Filename (Figure 49): objects/UN_GRAPHICALFRAME/unWindowTree.pnl
  - Panel size x: 120
  - Panel size y: 28

- Graphical element: button
  - Size x: 27
  - Size y: 27
- Purpose: open the window tree, the unUserPanel.pnl or unUserPanelWithPanel.pnl must be in HMI to be able to show a trend

Figure 49: unWindowTree.pnl

3. HOW TO CREATE A NEW HMI

To create a new HMI proceed as follow:

1. open the vision/graphicalFrame/unicosHMI_1274x996.pnl

2. change the size (Figure 50)

3. save the panel and use the panel file name in the <name> entry of the <properties> section of the HMI (section 2.3.1)

4. create the xml configuration file that will be used for the global var g_s_unicosHMI_previewConfig, the xml files data/unicos_1screen.xml, data/unicos_2screen.xml or data/unicos_3screen.xml can be taken as example

5. create a panel to start the new HMI, vision/graphicalFrame/unicosHMI.pnl can be taken as an example. The following initialize script must be used:

```
main()
{
  string sFile;
  dyn_string exInfo, ds;
  dyn_float df;

  sFile = “data/myHMIConfigurationFile.xml”;
  g_s_unicosHMI_previewConfig = sFile;
  unGraphicalFrame_loadHMI(exInfo);
  if(dynlen(exInfo) > 0) {
    fwExceptionHandling_display(exInfo);
    unGraphicalFrame_ChildPanelOnCentralReturn("vision/MessageWarning", "ERROR",
                          makeDynString("$1:Create "+g_s_unicosHMI_previewConfig+" and restart!"),
                          df, ds);
  }
  g_modules_opened = true;
  ModuleOff(myModuleName());
}
```

Figure 50: HMI size

4. HOW TO CREATE A NEW HMI COMPONENT

To create a new HMI proceed as follow:

1. create a panel

2. add any graphical element needed: button, text field, etc.

3. set the size x and y (Figure 51). The minimum value for size x is 120

4. set the Ref. point to 0, 0 (Figure 51). The x and y value in the <x> and <y> section of the component (section 2.3.2) will be the one corresponding to the real position in the panel. Ref.point value different to 0 lead to an offset position in x and y

5. if the graphical element must be disabled until the end of the initialization use a script similar to the following one

```
while(!g_b_unicosHMI_initEnd)
  delay(1);
component.enabled(true);
```

6. if the component is of type menu, menu button, the following initialize script must be used because the button is updated by callback function of the graphicalFrame lib.

```
main()
{
  dyn_string dsMenu, dsItem;
  string sKey = myModuleName()+"."+myPanelName();

  synchronized(g_m_dsPopupMenu) {
    if(mappingHasKey(g_m_dsPopupMenu, sKey)) {
      dsMenu = g_m_dsPopupMenu[sKey];
      dsItem = g_m_dsPopupMenuKey[sKey];
    }
    if(dynContains(dsMenu, menuButton.refName()) <= 0) {
      dynAppend(dsMenu, menuButton.refName());
      dynAppend(dsItem, $MENU_ITEM);
      g_m_dsPopupMenu[sKey] = dsMenu;
      g_m_dsPopupMenuKey[sKey] = dsItem;
    }
  }

  while(!g_b_unicosHMI_menu)
    delay(1);
  menuButton.enabled(true);
}
```

7. if the component is of type cascade menu, the following initialize script must be used because the cascade button is updated by callback function of the graphicalFrame lib.

```
main()
{
  dyn_string dsMenu, dsItem;
  string sKey = myModuleName()+"."+myPanelName();

  synchronized(g_m_dsCascadeMenu) {
```

```
    if(mappingHasKey(g_m_dsCascadeMenu, sKey)) {
      dsMenu = g_m_dsCascadeMenu[sKey];
      dsItem = g_m_dsCascadeMenuKey[sKey];
    }
    if(dynContains(dsMenu, menuButton.refName()) <= 0) {
      dynAppend(dsMenu, menuButton.refName());
      dynAppend(dsItem, $MENU_ITEM);
      g_m_dsCascadeMenu[sKey] = dsMenu;
      g_m_dsCascadeMenuKey[sKey] = dsItem;
    }
  }
  while(!g_b_unicosHMI_menu)
    delay(1);
  menuButton.enabled(true);
}
```

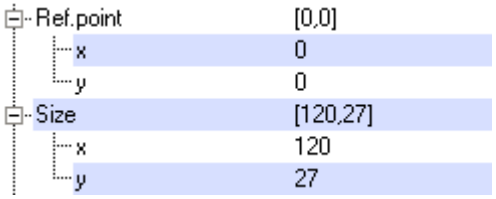


Figure 51: HMI component size and ref point

LIST OF FIGURES