



Extending the Workspace

Practical Examples

Citect SCADA

Bradley Cecil
1.0 / 2019

AVEVA Group plc
High Cross, Madingley Road
Cambridge CB3 0HB, UK
Tel +44 (0)1223 556655
Fax +44 (0)1223 556666

aveva.com

Table of contents

1. INTRODUCTION	4
2. SETUP.....	4
3. PAGE NAVIGATION	4
3.1. ADDING MULTIPLE TAB LEVELS TO PAGE NAVIGATION ZONE.....	4
3.1.1. Create a new 2 row tab Navigation Zone Page	5
3.1.2. Modify your Workspace Master Page to use the new 2 Row Navigation Page	6
3.2. USING TREE NAVIGATION	7
3.2.1. Configuring the Navigation Menu	7
3.2.2. Create a Tree Navigation Page	8
3.2.3. Create a new Master Page	9
3.2.4. Add a Handler for Interacting with the Tree	10
3.2.5. Adding a Watcher to Synchronize the Tree.....	10
3.2.6. Trying it out	12
3.2.7. Frequently Asked Questions	12
3.3. USING MENU NAVIGATION.....	13
3.3.1. Configuring the Navigation Menu	13
3.3.2. Create a Menubar Control	14
3.3.3. Create a Replacement Header bar Content Page	14
3.3.4. Create a New Master Page.....	14
3.3.5. Trying it Out!	14
3.4. CREATE A POPUPMENU TIED TO WORKSPACE NAVIGATION	14
4. POPUP FACEPLATES	15
4.1. DISPLAYING MULTIPLE POPUP FACEPLATE WINDOWS AT THE SAME TIME	15
4.1.1. Create a Page to Host your Faceplates in a Popup	15
4.1.2. Add a Workspace Handler for right clicking on Equipment	18
4.1.3. Modify your Master Page to not include a Faceplate Zone	18
4.2. DISPLAYING A SINGLE POPUP FACEPLATE WINDOW	18
5. USING AND CUSTOMISING CONTROLS	18
5.1. USING AND CUSTOMISING THE TREE CONTROL.....	19
5.1.1. Create a genie to represent your data	19
5.1.2. Create a data source which maintains a Citect Menu.....	20
5.1.3. Configure Treeview Properties.....	21

5.1.4. Implement the View Model Init Function	22
5.1.5. Implement the Datasource Item Retrieve Function.....	23
5.1.6. Implement the Datasource Watcher Function.....	24
5.1.7. Implement Click Handlers	25
5.1.8. Trying it Out.....	25
5.2. USING AND CUSTOMISING THE LIST CONTROL	25
5.2.1. Create display formats	26
5.2.2. Implement the Initialisation function.....	27
5.2.3. Implement an Exit function	28
5.2.4. Create a genie to represent your data	29
5.2.5. Implement Fill function	30
5.2.6. Implement Click Handlers	32
5.2.7. Trying it out!	32
6. GENERAL WORKSPACE CUSTOMISATIONS	32
6.1. ADDING AN ALARM BANNER TO A MASTER PAGE	32
6.1.1. Create an Alarm Banner Page	33
6.1.2. Insert an Alarm List on the Page	34
6.1.3. Configure the Alarm List.....	34
6.1.4. Set the Alarm Banner Page as the Default Page	34
6.2. ADDING ALARM LIMIT CONFIGURATION TO THE INFORMATION ZONE	34
6.3. ADDING A TAB TO THE INFORMATION ZONE.....	34
6.4. ADDING AN EQUIPMENT DEBUG INFO ZONE TAB	35
6.5. COLLAPSIBLE PANES	35
6.5.1. Design your Master Page.....	36
6.5.2. Write Code to Toggle Your Panes and Adjust the Panes.....	36
6.5.3. Bind your Toggle Function to a Command	37
6.6. AUTO EXPANDING AND COLLAPSIBLE FACE PLATE PANE.....	37
6.6.1. Design your Master Page.....	38
6.6.2. Monitor for Context Change	39
6.6.3. Bind your Toggle Function to a Command	40
6.7. THE AUTO CLOSE POPUP PATTERN.....	41
7. CREATING YOUR OWN THEME COMBINATION	41
8. PRINTING THE CONTENT PAGES IN YOUR SYSTEM	42

1. Introduction

Welcome to the Citect SCADA Workspace Extension guide. This guide shows you different ways to extend the workspace which help you understand how to do simple to more complex customisations.

2. Setup

All the example in this guide consist of a single project that all use a common include project called: “Starter”. The starter project is simply “Create new start project, select “SA_Style_1_MultiRes” with two small modifications. This technique allows you to use the examples with the latest “SA_Style_1_MultiRes” starter project at any-time.

The modifications are:

- 1. Delete the label “_CALLBACK_EquipmentRightClick(sEquip, sEquipSecondary)”
- 2. Delete the Level 1 menu item “MyPlant” from menu “Navigation”. Each example project provides their own home page.

Visualization Menu Configuration Pages					
Save Discard Copy Paste Delete Row(s)					
Row	Page	Level 1	Level 2	Level 3	
1	Navigation	MyPlant			

Once you have restored the Starter project, restore each workspace example project that you want to explore.

3. Page Navigation

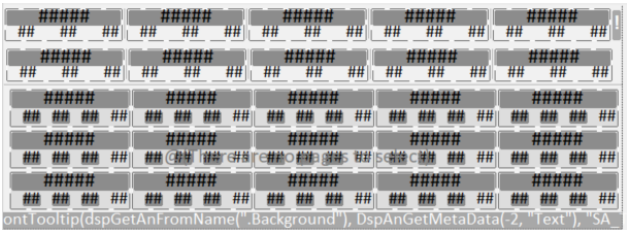
3.1. Adding Multiple Tab Levels to Page Navigation Zone

<TODO> Show finished screen

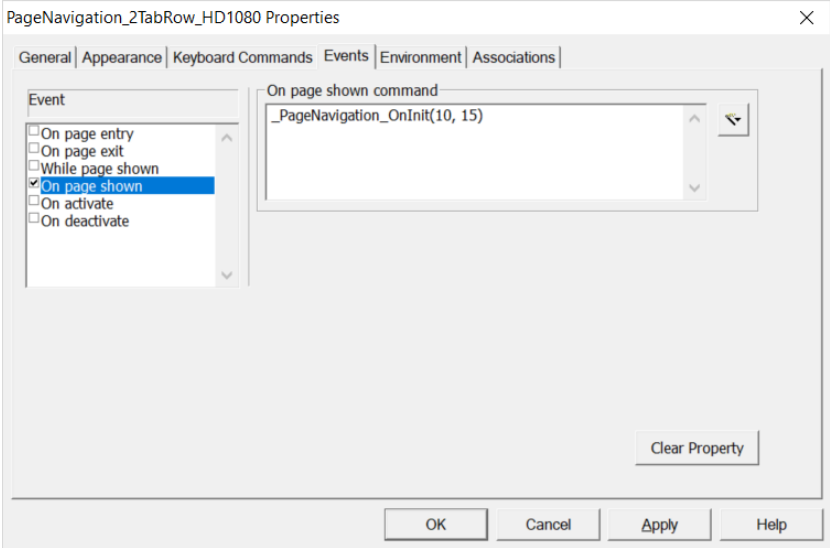
[WARNING] Adding more Tabs, means more buttons. Every button has a minimum of 12 alarm counts (top 3 priority based system). The default page navigation zone has approximately 5 x 20 x 12 (1200) alarm counts when all buttons and tabs are configured. To get the summary counts on the tabs, the page navigate zone needs to count the alarms for *every* page button. Alarm Counts can have a large impact on your alarm servers. Please ensure your Alarm Servers have the capacity to service the increase load of counting; taking into account the number of simultaneously connected clients.

3.1.1. Create a new 2 row tab Navigation Zone Page

- 1. Open the page "PageNavigation_1TabRow_HD1080"
 - a. If you are using UHD4K, open that page instead / or as well
- 2. Delete the bottom row of buttons
- 3. Duplicate the top row of tabs and reposition them as shown below:



- 4. Open the "Page Properties from the File menu
- 5. Go to the Events tab and select "On page shown". Configure the "On page shown command" as follows.

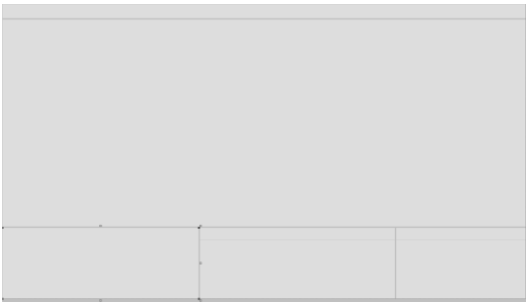


The number "10" means there are now "10" tabs. The "15" means there are now "15" buttons per tab.

- 6. Save the page as “PageNavigation_2TabRow_HD1080”
 - a. If you are using UHD4K save with _UHD4K

3.1.2. Modify your Workspace Master Page to use the new 2 Row Navigation Page

- 1. Open your Master Page (e.g. Master_PageMenu1_HD1080
- 2. Display the Genie Parameters form for the Page Navigation Zone pane genie. (bottom left of your master page)



- 3. Change the default page to “PageNavigation_2TabRow”

Workspace - Pane Properties

Name: Navigation

Basic Pane Properties:

Default Page: PageNavigation_2TabRow_HD1080

Display Mode: Stretch

Is Default Pane?: FALSE Display Scrollbars: FALSE

Autofill Behavior:

Fill Mode: Static

Content Types:

Excluded Autofill Panes:

Tabbed Pane Properties:

Tab Header Pane Name:

Tab Control Name:

Equipment Reference Associations:

Categories:

OK Cancel Help

- 4. Save the page
- 5. Compile and Run up Citect SCADA

3.2. Using Tree Navigation

Example Project: Example_TreeNavitationSimple

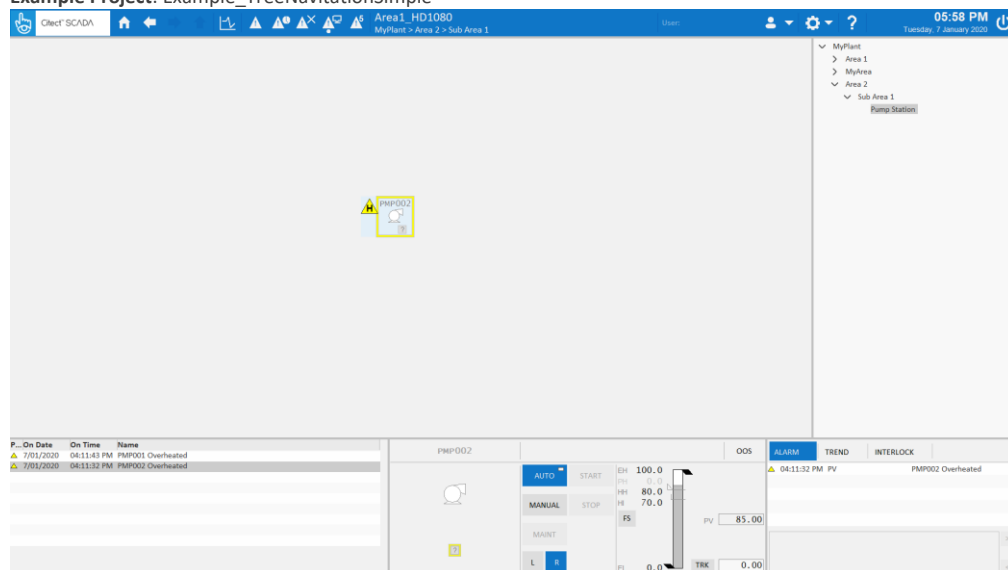


Figure 1 - Tree based Navigation

To replace Page based Navigation with Tree based navigation can be done using the following design steps:

- Configuring the Navigation Menu
- Create a Tree Navigation Page
- Create a new Master Page
- Add a Handler for interacting with the tree
- Adding a Watcher to Synchronize the Tree
- Trying it out!

3.2.1. Configuring the Navigation Menu

The workspace uses the “Navigation” menu as its source of menu items for navigation. We are going to continue to use this method as built-in design for this menu. As we are using a tree, we are now free from the restriction of only using the first 3 levels. From 2018 onwards you can configure up to 6 levels from within Citect Studio.

1. In Citect Studio, open the Visualisation > Menu Configuration activity
2. Configure the menu as follows:

Row	Page	Level 1	Level 2	Level 3	Level 4	Menu Command	Target Page	Comment	Order
1	Navigation	MyPlant				Navigation_ShowTarget	Instructions_HD108		
2	Navigation	MyPlant	Area 1	Page A1 A		Navigation_ShowTarget	Area1_A_HD1080		1
3	Navigation	MyPlant	Area 1			Navigation_ShowTarget	Area1		
4	Navigation	MyPlant	Area 2			Navigation_ShowTarget	Area2		2
5	Navigation	MyPlant	Area 2	Sub Area 1		Navigation_ShowTarget	Area2_SubArea1		
6	Navigation	MyPlant	Area 2	Sub Area 1	Pump Station	Navigation_ShowTarget	PumpStation		

3. [Save] your changes.

3.2.2. Create a Tree Navigation Page

A page is required to host the tree control

- 1. Create a new page based on the Blank template
- 2. Change the size of the page to:
 - a. Width: 388
 - b. Height: 760
- 3. Paste the treeview genie from the "sa_contorls" library onto your page at position 4, 4.
- 4. Select and right click on the treeview gene, and display the Genie Properties for the Treeview genie
- 5. Give the genie the name of "Treeview". Our cicode is going to reference this genie.
- 6. Display the Genie Parameters for the Treeview genie
- 7. Modify the properties as follows:

TreeView

Basic Setup:

Datasource

Navigation

Use Display Names

FALSE

Search:

Search Box Name

Search

Display:

TreeView Height

976

TreeView Width

386

Row Height

24

Display Checkboxes

FALSE

Auto-check Children

FALSE

Display Alarms

FALSE

Display Shelved

TRUE

Events:

On Init Complete Function

On Check Function

On Left Click Function

MenuTree_OnClick

On Right Click Function

Customization:

View Genie

sa_controls.treeviewitem_hd1080

View Init Function

_TreeView_InitViewGenie_HD1080

Fill Function

_TreeView_Update

View Model Data Size

0

View Model Init Function

Datasource Item Retrieve Function

Is Source Ready Function

Datasource Watcher Function

Can Check Items Function

AN

5

OK

Cancel

Help

8. Press [OK] and [SAVE] the page as “MenuTree_HD1080”

3.2.3. Create a new Master Page

1. Open an existing Master page such as “MasterPage_PageMenu1_HD1080”

2. Change the dimensions of the Content pane to to:

a. **Width:** 1528

b. **Height:** 760

3. Move the existing “Navigation” pane to the right of the Content Page

4. Change the Navigation pane dimensions to to:

a. **Width:** 388

b. **Height:** 760

5. Open the Genie Parameters of the “Navigation” pane genie

6. Change the default page to: "MenuTree_HD1080"
7. [OPTIONAL] In the gap where the Navigation Pane use to be, create a new pane. We will host an "Active Alarms" display here
 - a. Paste the pane genie from the "sa_workspace" library onto your page and position where the Navigation Zone pane was originally (bottom left of your Master page)
 - b. Resize the pane to:
 - i. Width: 720
 - ii. Height: 260
 - c. See 6.1 Adding an Alarm Banner to a to create your Alarm Banner
8. [Save As] your master page with a new name (e.g. MasterPage_TreeMenu1_HD1080)
9. Go to Citect Studio, and then navigate to the Visualisation > Pages Activity. **Ensure the Content Type field for your master page is set to "Master". This ensures it displays in the Computer Setup Wizard.**

3.2.4. Add a Handler for Interacting with the Tree

When the operator clicks on items in the navigation tree, we want to trigger the navigation to the page the item represents.

1. Create a new code file in your project. E.g. "Code.ci"
2. Add a new function as follows:

```
FUNCTION MenuTree_OnClick(INT hMenuItem)
    _Navigation_RunCommandFromMenuHandle (hMenuItem,
    TRUE);
END
```

The RunCommandFromMenuHandle function is part of the workspace. The second parameter, set to TRUE, tells the workspace to save this action to the navigation stack. This allows the back/forward buttons to work.

3. [Save] the file.

3.2.5. Adding a Watcher to Synchronize the Tree

When Navigation actions occur in the Workspace, for example by selecting "Navigate" from an alarm, we want to update the selected item in the tree based upon the currently displayed page. This can be achieved by adding a cicode object that calls a function to keep track of the active page, and when it changes locate and highlight it in the tree.

To do this:

1. Open your "Menutree_HD1080" page
2. Add a Cicode Object to your MenuTree_HD1080 page
3. Display the properties for the Cicode Object and enter the following in the Command Field:


```
MenuTree_SyncSelection()
```

4. [Save] the page
5. Switch to the Cicode Editor
6. Open your code file that you created earlier
7. Now create the “MenuTree_SyncSelection” function. This function follows the same pattern as the Navigation Zone page. It keeps a cache of the current page and when it differs it will store the new page, look up the new page in the Navigation Menu, turn it into a path and then Locate it in the tree.

```

FUNCTION MenuTree_SyncSelection()

    STRING sContext, sStartupContext;
    STRING sCachedPrimaryPage;
    STRING sCurrentPrimaryPage;
    STRING sNavigationPath;
    INT hMenuItem;

    INT nWinCurr = WinNumber();
    INT nTreeviewAN = DspGetAnFromName("Treeview");

    // Check if the current page has changed from the cached page
    sCachedPrimaryPage = PageGetStr(g_sCachedPrimaryDisplayPage, nWinCurr);
    sCurrentPrimaryPage = PageGetStr(g_sPrimaryDisplayPage, Workspace_GetWindow());

    IF (sCachedPrimaryPage = sCurrentPrimaryPage) THEN
        // It hasn't changed, so leave selection as is
        RETURN;
    END

    // Update the cache with the new current page
    PageSetStr(g_sCachedPrimaryDisplayPage, sCurrentPrimaryPage, nWinCurr);

    IF (sCurrentPrimaryPage <> "") THEN
        // Find the menu item corresponding to the current page
        hMenuItem = _Navigation_FindMenuItem(sCurrentPrimaryPage);
        IF (hMenuItem <> BAD_HANDLE) THEN
            sNavigationPath = _Menutree_GetNavigationPath(hMenuItem);
            Treeview_Locate(nTreeviewAN, sNavigationPath);
        END
    END
END

```

8. Now add a function to return the menu path (i.e. “MyPlant.Area 2.Sub Area 1”). The tree control’s “Locate” method looks items up by path.

```

PRIVATE
STRING
FUNCTION _Menutree_GetNavigationPath (INT hMenuHandle)

    STRING    sMenuName;
    STRING    sMenuPath;
    INT       hMenuItem = hMenuHandle;

    WHILE (hMenuItem <> BAD_HANDLE) DO
        sMenuName = MenuNodeGetProperty (hMenuItem, 0);

        IF (sMenuName = g_sNavigation_CoreNavigationMenu OR
            sMenuName = g_sNavigation_HeaderBarMenu OR
            sMenuName = g_sNavigation_EquipmentModel) THEN
            // Don't add the root items "Navigation", "HeaderBar" or "__EquipmentModel" to
the path
            hMenuItem = BAD_HANDLE;
        ELSE
            // Add the item to the path
            IF (sMenuPath <> "") THEN
                sMenuPath = sMenuName + "." + sMenuPath;
            ELSE
                sMenuPath = sMenuName;
            END

            // Move to parent item
            hMenuItem = MenuGetParent (hMenuItem);
        END
    END

    RETURN sMenuPath;
END

```

3.2.6. Trying it out

The “Example_TreeNavigationSimple” demonstrates the steps in this tutorial. It also comes with some default equipment so you can try out the tree synchronisation.

1. Run the computer Setup Wizard and select the “DEFAULT” profile. This profile has the Startup Context set to “MyPlant”, and your Startup page to “Master_TreeMenu_HD1080”.
2. Run Citect SCADA up
3. By default you will see the “Instructions” page.
4. You can use the tree to navigate around the project.
5. When on the Instruction page, click the “ON” buttons to trigger alarms on Pump 1 and Pump 2.
6. Right click on the alarms and select “Navigate”. Notice how the tree automatically locates the page in the tree?
7. Also try clicking on the navigation genie on the “Area 1” page. Notice how the tree updates when this link is clicked.

3.2.7. Frequently Asked Questions

Q. Can I get the Alarm Counts on the nodes like the Page Navigation Zone?

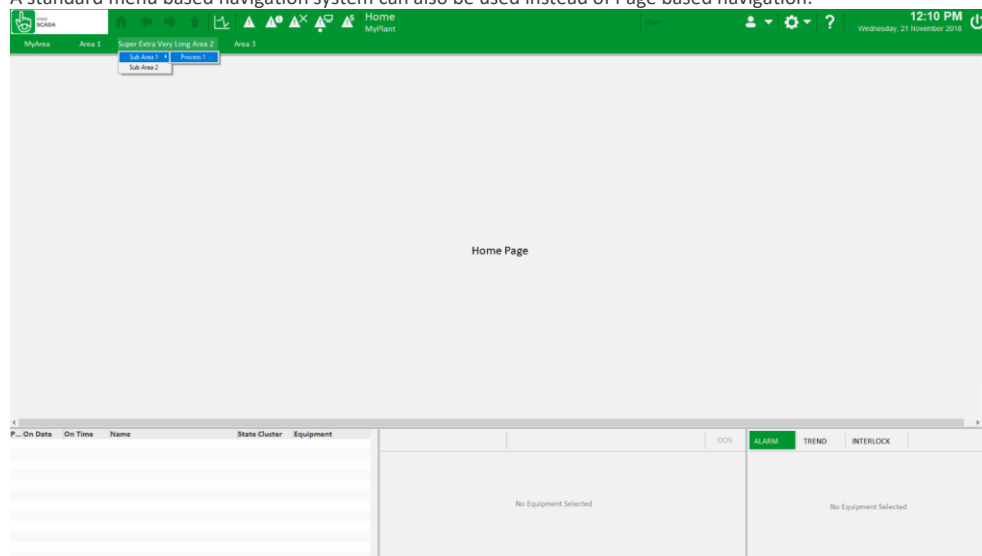
A. Not out of the box. You will need to write this code yourself. You will need to understand and study the PageNavigation.ci code and have a look at customising the Treeview to do this. Note: you should also consider the complexity of allowing more tree levels than we do as you would need to handle rolling up counts.

Q. Can I put a search box on this, as I have a large menu hierarchy?

A. Yes, Open the DefaultTrend_HD1080 page, and copy the searchbox, and any other control such as + / - and Clear. Paste and position them onto your page. Ensure the pasted Text Box ActiveX Object Name matches the name you specify in your Treeview Genie Parameters – Search Box Name field. That's it!

3.3. Using Menu Navigation

A standard menu based navigation system can also be used instead of Page based navigation.



To replace Page based Navigation with Menu based navigation can be done using the following design steps:

- Configuring the Navigation Menu
- Create a Menubar Control
- Create a Replacement Header bar Content Page
- Create a New Master Page
- Create a Popumenu tied to Workspace Navigation
- Trying it out!

3.3.1. Configuring the Navigation Menu

The workspace uses the “Navigation” menu as its source of menu items for navigation. We are going to continue to use this method as built-in design for this menu. As we are using a tree, we are now free from the restriction of only using the first 3 levels. From 2018 onwards you can configure up to 6 levels from within Citect Studio.

1. In Citect Studio, open the Visualisation > Menu Configuration activity

2. Configure the menu as follows:

Row	Page	Level 1	Level 2	Level 3	Level 4	Menu Command	Target Page	Comment	Order
1	Navigation	MyPlant				Navigation_ShowTarget!	Instructions_HD108		
2	Navigation	MyPlant	Area 1	Page A1 A		Navigation_ShowTarget!	Area1_A_HD1080		1
3	Navigation	MyPlant	Area 1			Navigation_ShowTarget!	Area1		
4	Navigation	MyPlant	Area 2			Navigation_ShowTarget!	Area2		2
5	Navigation	MyPlant	Area 2	Sub Area 1		Navigation_ShowTarget!	Area2_SubArea1		
6	Navigation	MyPlant	Area 2	Sub Area 1	Pump Station	Navigation_ShowTarget!	PumpStation		

3. [Save] your changes.

3.3.2. Create a Menubar Control

3.3.3. Create a Replacement Header bar Content Page

3.3.4. Create a New Master Page

1. Open an existing Master page such as "MasterPage_PageMenu1_HD1080"
2. Change the dimensions of the Content pane to: (adjust the top down)
 - a. **Width:** 1920
 - b. **Height:** 726
3. Change the dimensions of the Header pane to: (adjust the bottom down)
 - a. **Width:** 1920
 - b. **Height:** 88
4. Open the Genie Parameters of the "HeaderBar" pane genie
5. Change the default page to: "MenuHeaderbar_HD1080"
6. [OPTIONAL] In the gap where the Navigation Pane use to be, create a new pane. We will host an "Active Alarms" display here
 - a. Paste the pane genie from the "sa_workspace" library onto your page and position where the Navigation Zone pane was originally (bottom left of your Master page)
 - b. Resize the pane to:
 - i. **Width:** 720
 - ii. **Height:** 260
 - c. See 6.1 Adding an Alarm Banner to a to create your Alarm Banner
7. [Save] your master page with a new name (e.g. MasterPage_MenuBar_HD1080)
8. Go to Citect Studio, and then navigate to the Visualisation > Pages Activity. Ensure the Content Type field for your master page is set to "Master". This ensures it displays in the Computer Setup Wizard.

3.3.5. Trying it Out!

3.4. Create a Popupmenu tied to Workspace Navigation

4. Popup Faceplates

Faceplates by default are displayed in the Faceplate Zone as part of the standard Workspace starter pages:

- Master_PageMenu1_HD1080
- Master_PageMenu1_UHD4K

<INSERT IMAGE OF FACEPLATE ZONE>

For systems which would like to maximize screen space for mimics, or would prefer faceplates in popup windows the workspace has native support for these including support for the new auto-filling capabilities.

To change/create a workspace to have a popup faceplate first requires you to decide how your operators want to interact with the equipment during operation.

So first answer this question: “Do you need to see the faceplates for multiple equipment at the same time?”

4.1. Displaying Multiple Popup Faceplate Windows at the same time

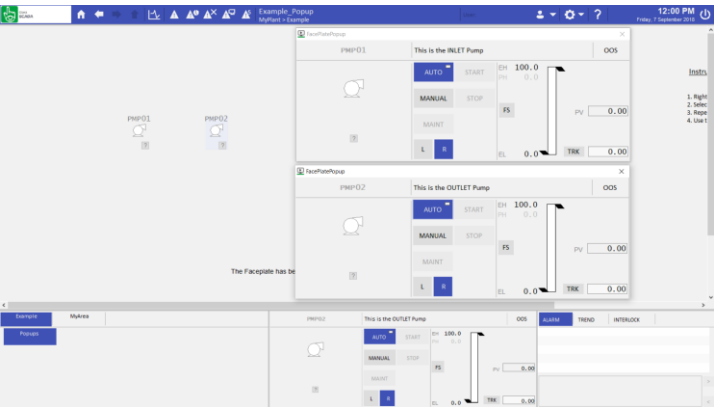


Figure 2 - Multiple Face plates displayed

This design requires the following steps:

- Modify your Master Page to not include a Faceplate Zone
- Add a Workspace Handler for right clicking on Equipment
- Create a Page to Host your Faceplates in a Popup

4.1.1. Create a Page to Host your Faceplates in a Popup

To ensure only one popup is ever displayed, modify your right click handler cicode:

Commented [B1]: This needs to be revised. Not sure it is bullet proof.

Commented [BC2]: Need to remove that faceplate at the bottom

The key change is the addition of “128” to the WinNewAt call. This will ensure the faceplate popup page is only ever displayed once.



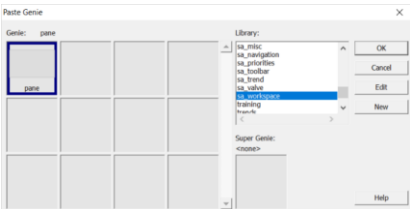
Figure 3 - Faceplate Page

- 1. Create a new Page based on the Situational Awareness HD0189 Blank Page Template
- 2. Change the size to be:

Width: 718px
Height: 260px

The size should match the size of your intended faceplates + the size of your faceplate header (if you have one). The size we have chosen here is designed to host the workspace’s usual embedded Faceplate Zone as a popup.

- 3. Add a Pane Genie (sa_workspace library) for the Faceplate Header



Width: 716 px
Height: 44 px

- 4. Add a Pane genie (sa_workspace library) for the Faceplate Body
- Width:** 716 px
Height: 216 px
- 5. Display the “Genie Parameters” for the Faceplate Header Pane and configure it as follows:

Workspace - Pane Properties

Name

FacePlateHeader

Basic Pane Properties:

Default Page

FacePlateHeader_HD1080

Display Mode

Stretch

Is Default Pane?

FALSE

Display Scrollbars

FALSE

Autofill Behavior:

Fill Mode

Static

Content Types

Excluded Autofill Panes

Tabbed Pane Properties:

Tab Header Pane Name

Tab Control Name

Equipment Reference Associations:

Categories

OK

Cancel

Help

6. Display the “Genie Parameters” for the faceplate body pane and configure it as follows:

Workspace - Pane Properties

Name

FacePlate

Basic Pane Properties:

Default Page

FP_DCOL_HD1080

Display Mode

Stretch

Is Default Pane?

TRUE

Display Scrollbars

FALSE

Autofill Behavior:

Fill Mode

Static

Content Types

Excluded Autofill Panes

Content

Tabbed Pane Properties:

Tab Header Pane Name

Tab Control Name

Equipment Reference Associations:

Categories

MEO

OK

Cancel

Help

7. Save the Page (e.g. “FaceplatePopup_HD1080”)

4.1.2. Add a Workspace Handler for right clicking on Equipment

Create a new cicode file in your project. E.g. “MyWorkspace.ci”

- 1. Create a new function with the following code:

```
FUNCTION MyEquipmentRightClickCallback(STRING sEquip, STRING sEquipSecondary)

    STRING sPage;
    INT nX, nY;
    INT iSelection;

    DspPopupMenu(0, "Configure");

    iSelection = DspPopupMenu();

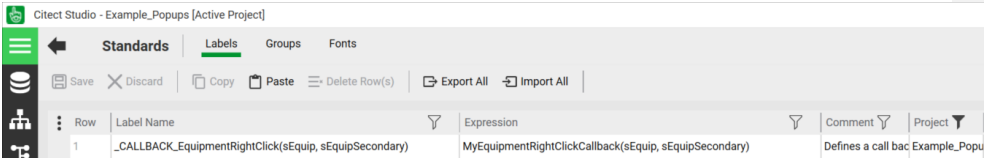
    SELECT CASE iSelection
    CASE 1
        sPage = "FaceplatePopup"
    END SELECT

    IF (sPage <> "") THEN
        DspGetMouse(nX, nY);
        WinNewAt(sPage, nX, nY, 5);
    END
END
```

The above code will get called every time an equipment is selected in the workspace, and the operator right clicks on the object.

A menu with a single item “Configure” will be displayed. If the user selects “Configure”, then the faceplate popup page is displayed automatically. The modes used for WinNewAt will prevent Resize (4) and ensures the window is closed whenever the operator navigates away from the mimic page.

- 2. Save the file
- 3. Go to Citect Studio
- 4. Navigate to Standards Activity and the Labels view



4.1.3. Modify your Master Page to not include a Faceplate Zone

4.2. Displaying a single Popup faceplate Window

<picture>

Need to change WinNewAt to have 128 and modify pane to be autofill

Running it up

- 1. Run up Citect
- 2. Select a piece of equipment
- 3. Right click and select “Faceplate” from the context menu

5. Using and Customising Controls

5.1. Using and Customising the Tree Control

The tree control genie supplied as part of the SA_Controls library of the SA_Include project provides a fast way to render single or multi-column displays with hierarchical data. On top of the options available via the genie parameters, such as checkboxes or changing the presentation of the tree nodes, engineers can also customise the treeview behaviour and columns via code.

This example will show you how to customize the tree control to render data from a custom data source and display some of that data numerically and as a vector based genie.

Food	Sugar (g/100)	Fat (g/100)	Health Rating
✓ Breakfast			
Bran	10	2	★★★★★
Oats	15	1	★★★★★
Bubbles	20	2	★★★★
✓ Sometimes			
Works Burger	20	60	★
Hotdog	20	30	★
Violet Crumble	45	10	★★★★★

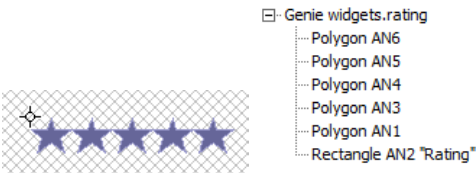
Note: For 2018 users, this requires Update 1 for Citect SCADA 2018

To create a custom treeview control with graphical representation of data, then it can be done using the following design steps:

- Create a genie to represent your data
- Create a data source which maintains a Citect Menu
- Configure Treeview Properties
- Implement the View Model Init Function
- Implement the Datasource Item Retrieve Function
- Implement the Datsource Watcher Function
- Implement Click handlers
- Trying it out!

5.1.1. Create a genie to represent your data

Cells within the treeview's grid can also be rendered as a genie rather than plain text. For this example we are going to render the numeric Rating attribute of a Food item, values of 1-5, as a sequence of stars as shown below.



The most important thing to understand when using genies to represent data with the treeview (and the listview) genies is **meta data** is used as the interface. When the treeview animates it will take the data from the view array and pass it automatically into your genie. You will see how you associate your data to your metadata in 5.1.5 Implement the Datasource Item Retrieve Function.
The 'Rating' rectangle is where the metadata for the rating genie is held.

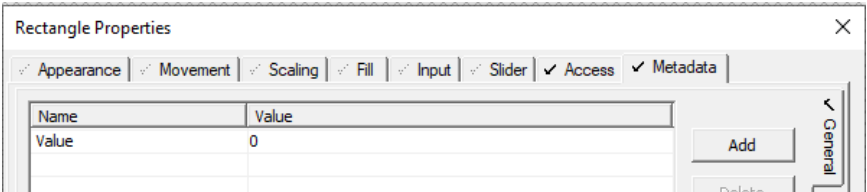


Figure 4 - Rating Widget Meta data

The individual stars hide and show based upon a visibility expression and referencing the metadata from the Rating rectangle.

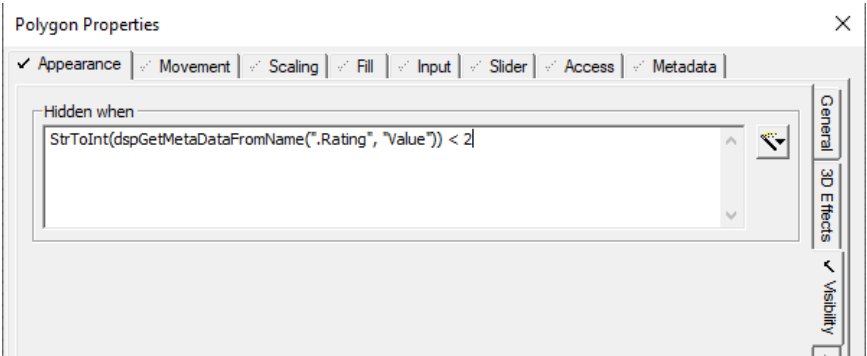


Figure 5 - The Visibility expression of Star 2 in the widget

5.1.2. Create a data source which maintains a Citect Menu

The treeview displays nodes based upon a Citect menu and uses them as a datasource for the **nodes**. Your data source is unlikely to be a Citect menu, it could be the file system, or some other external source and it likely contains

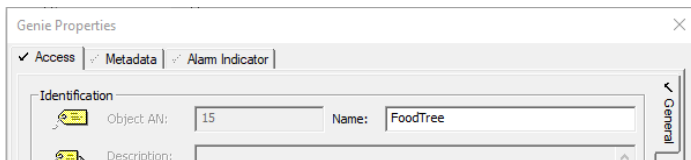
various attributes/metadata associated with each node that you might want to display in columns alongside the node

The example demonstrates how to use a hierarchical map as a datasource and use that as a source for a Citect menu which the treeview understands.

You will find the code to do this in the "FoodSource.ci" file.

5.1.3. Configure Treeview Properties

1. On a test page, paste the treeview genie from the [sa_controls] library
2. Right click on the genie and select "Properties..."
3. In the Access Tab, name the genie "FoodTree"



4. Display the "Genie Parameters" for the treeview genie and configure it as follows:

TreeView

Basic Setup:

Datasource

Food

Use Display Names

FALSE

Search:

Search Box Name

Search

Display:

TreeView Height

500

TreeView Width

500

Row Height

24

Display Checkboxes

FALSE

Auto-check Children

FALSE

Display Alarms

FALSE

Display Shelved

FALSE

Events:

On Init Complete Function

On Check Function

On Left Click Function

MyFoodTree_OnLeftClick

On Right Click Function

MyFoodTree_OnRightClick

Customization:

View Genie

sa_controls.treeview/item_hd1080

View Init Function

_TreeView_InitViewGenie_HD1080

Fill Function

_TreeView_Update

View Model Data Size

3

View Model Init Function

MyFoodTree_ViewModelInit

Datasource Item Retrieve Function

MyFoodTree_DataRetrieve

Is Source Ready Function

Datasource Watcher Function

MyFoodTree_ModelWatcher

Can Check Items Function

AN

15

OK

Cancel

Help

5. If you want search capability make sure the search controls are added and the name of the search box is configured. You can copy and paste these from the Trend Page in the sa_include project.

@(Search)

5.1.4. Implement the View Model Init Function

The treeview has a 3-dimensional array attached to its AN, called the View Model. The View Model is responsible for containing the data that will be rendered to the screen (e.g. the tree nodes), but also contains other information such as how to render the information, for example as text or a genie. The view array dimensions consist of column, row pairs (i.e. cell) and each cell has a Z dimension which allows us to store additional information for that cell. The “View Model Init Function” allows you to setup additional columns (multi-column treeview) and configure their behaviour and appearance.

In the code below we are adding 3 columns: Sugar, Fat, and Rating. “Row 0” is called the “Information Row” and is where we configure the column name, width and whether all rows of that column will display text or a genie.

```
FUNCTION MyFoodTree_ViewModelInit(INT hViewArray, INT nCustomDataStartIndex)

    // Get the total width of the tree control
    INT nTreeWidth = ArrayGetInt(hViewArray, TREEVIEW_INFORMATION_COLUMN, TREEVIEW_INFORMATION_ROW,
TREEVIEW_ZINDEX_ARRAY_XMAX) - 250;

    // Udate the Tree Column's width to be total tree control width - the total size of your columns
    ArraySetInt(hViewArray, nTreeWidth, 1, ARRAY_INFORMATION_ROW, ARRAY_ZINDEX_COLUMN_WIDTH);

    // Create the Sugar Column
    ArraySetInt(hViewArray, ARRAY_COLUMN_TYPE_STRING, nCustomDataStartIndex,
ARRAY_INFORMATION_ROW, ARRAY_ZINDEX_COLUMN_TYPE); // The Column will display Text
    ArraySetString(hViewArray, "Sugar (g/100)", nCustomDataStartIndex,
ARRAY_INFORMATION_ROW, ARRAY_ZINDEX_COLUMN_VALUE); // The heading of the column will be. We don't use
this is the treeview
    ArraySetInt(hViewArray, 50, nCustomDataStartIndex,
ARRAY_INFORMATION_ROW, ARRAY_ZINDEX_COLUMN_WIDTH); // The default width of the column

    // Create the Fat Column
    ArraySetInt(hViewArray, ARRAY_COLUMN_TYPE_STRING, nCustomDataStartIndex + 1,
ARRAY_INFORMATION_ROW, ARRAY_ZINDEX_COLUMN_TYPE);
    ArraySetString(hViewArray, "Fat (g/100)", nCustomDataStartIndex + 1,
ARRAY_INFORMATION_ROW, ARRAY_ZINDEX_COLUMN_VALUE); // The heading of the column will be. We don't use this
is the treeview
    ArraySetInt(hViewArray, 50, nCustomDataStartIndex + 1,
ARRAY_INFORMATION_ROW, ARRAY_ZINDEX_COLUMN_WIDTH); // The default width of the column

    // Create the Health Rating Column. Note the change in column type to signify a genie will be used to
represent the data.
    // The genie to display the value with is set per cell at render time.
    ArraySetInt(hViewArray, ARRAY_COLUMN_TYPE_NORMALGENIE, nCustomDataStartIndex + 2,
ARRAY_INFORMATION_ROW, ARRAY_ZINDEX_COLUMN_TYPE); // The Column will be display a genie containing "0-5
vector stars"
    ArraySetString(hViewArray, "Rating (1-5)", nCustomDataStartIndex + 2,
ARRAY_INFORMATION_ROW, ARRAY_ZINDEX_COLUMN_VALUE); // The heading of the column will be. We don't use
this is the treeview
    ArraySetInt(hViewArray, 150, nCustomDataStartIndex + 2,
ARRAY_INFORMATION_ROW, ARRAY_ZINDEX_COLUMN_WIDTH); // The default width of the column

END
```

5.1.5. Implement the Datasource Item Retrieve Function

Whenever the treeview needs data for the custom columns of a specific item, it will call the “Datasource Item Retrieve Function”.

In this function you retrieve the extra column data from your data source. If your column is a genie display type, you also set what genie you want the cell to display as. The treeview allows every cell to have a different genie. In our example we only use one genie which can animate between 1 and 5 stars. But you could for example have 5 different genies and choose a different one for each rating value.

Passing the rating value to the genie’s “Value” metadata is done via the z-dimension of the view array (ARRAY_ZINDEX_CELL_METADATA). It uses the format of: .<animation name>|<metadata name>|<value>

If your genie has grouped objects, and the object the metadata resides on is within a group you can still get to that as well by using dot syntax. e.g.

```
.Parent.Child.MyMetaData|44
```

You can also set multiple metadata at the same time by adding a |e.g.

```
.Parent.Child.MyMetaData1|44|.Parent.Child.MyMetaData2|Hello
```

```

FUNCTION MyFoodTree_DataRetrieve(INT hViewArray, INT nCustomDataColumnIndex, INT hTreeItem, INT nRenderItem)

    ErrSet(1);

    STRING sItemName = MenuNodeGetProperty(hTreeItem, MENU_PROP_NAME);
    STRING hItemMap;

    hItemMap = MapOpen(sItemName, 1);
    IF (IsError() = 0) THEN
        ArraySetInt(hViewArray, MapValueGet(hItemMap, "Sugar"), nCustomDataColumnIndex,
nRenderItem, ARRAY_ZINDEX_CELL_Value);
        ArraySetInt(hViewArray, MapValueGet(hItemMap, "Fat"), nCustomDataColumnIndex + 1,
nRenderItem, ARRAY_ZINDEX_CELL_Value);

        // Since the rating is being representing by a genie we need to tell it which genie to use.
        ArraySetString(hViewArray, "widgets.rating", nCustomDataColumnIndex + 2,
nRenderItem, ARRAY_ZINDEX_CELL_Value);

        ArraySetString(hViewArray, ".Rating|Value|" + MapValueGet(hItemMap, "Rating"):#,
nCustomDataColumnIndex + 2, nRenderItem, ARRAY_ZINDEX_CELL_META1);
    ELSE
        // Clear out the view array if there was an error
        ArraySetString(hViewArray, "", nCustomDataColumnIndex, nRenderItem,
ARRAY_ZINDEX_CELL_Value);
        ArraySetString(hViewArray, "", nCustomDataColumnIndex + 1, nRenderItem,
ARRAY_ZINDEX_CELL_Value);
        ArraySetString(hViewArray, "", nCustomDataColumnIndex + 2, nRenderItem,
ARRAY_ZINDEX_CELL_Value);
        ArraySetString(hViewArray, ".Rating|Value|0", nCustomDataColumnIndex + 2, nRenderItem,
ARRAY_ZINDEX_CELL_META1);
    END
    ErrSet(0);

END

```

5.1.6. Implement the Datasource Watcher Function

The “Datasource Watcher Function” allows engineers to trigger the treeview control to refresh itself if the datasource changes underneath it. Triggering the treeview to refresh is done via the internal function “_Treeview_Reload”.

For the example, whenever the datasource changes (i.e. the data that affects the core tree) we trigger a refresh. As we are rendering a multi-column display we are also watching to see if the meta data like Rating, Fat, and Sugar has changed. So why don't we just use that to also trigger a reload? A full reload is only required when the tree structure changes. For column data we don't need to rebuild the tree and only need to update the cells currently on display.


```

FUNCTION MyFoodTree_ModelWatcher (STRING sDatasource, INT nTreeviewAN)

    INT nOriginalKeys = MapKeyCount (g_hFoods);
    INT nKeys = nOriginalKeys;
    INT hViewArray;

    WHILE (TRUE) DO
        nKeys = MapKeyCount (g_hFoods);
        IF (nKeys <> nOriginalKeys) THEN
            _Treeview_Reload (nTreeviewAN);
            nOriginalKeys = nKeys;
        ELSE
            // Look for changes to the data of the nodes (e.g. rating, fat, sugar)
            IF (MapValueGet (g_hFoods, "Dirty")) THEN
                MapValueSet (g_hFoods, "Dirty", 0);
                _MyFoodTree_UpdateCustomData (nTreeviewAN);
            ELSE
                SleepMS (250);
            END
        END
    END
END
END

```

5.1.7. Implement Click Handlers

5.1.8. Trying it Out

The “Example_Treeview” project demonstrates the steps in this tutorial.

1. Run the computer Setup Wizard and select the “DEFAULT” profile.
2. Run Citect SCADA up and you will see the treeview demonstration page.
3. Try: expanding / collapseing and searching for food items.
4. Try: Left clicking on the node text.
5. Try: Right clicking on the Breakfast and Sometimes group nodes and adding a new food item.
6. Try: Right clicking on a food item like “Hotdog” and editing its values. Note how the tree updates.

5.2. Using and Customising the List Control

The Listview genie control can be used to render tables of data from custom data sources. The listview genie allows data to be drawn as text or graphically.

The “Example_CustomList” project demonstrates the steps in this tutorial. It contains two source files, one for a simple list using pure text presentation, and a second using a mixture graphical and text.

This walkthrough will focus on the second example as it is minor superset of the first example.

Simple Multi-Column List

FlightCode	Origin	Destination
CT0	Brisbane	Townsville
CT1	Brisbane	Mackay
CT2	Brisbane	Bundaberg
CT3	Brisbane	Rockhampton
CT4	Brisbane	Goldcoast
CT5	Brisbane	Sydney

Graphic Multi-Column List

Graphic	Value1	Value2
<div></div>	0	33
<div></div>	7	44
<div></div>	24	54
<div></div>	47	70
<div></div>	51	89
<div></div>	60	34

To use a custom listview control with custom data can be done using the following design steps:

- Create display formats
- Implement the Initialisation function
- Implement an Exit function
- Create a genie to represent your data
- Implement Fill function
- Implement Click Handlers
- Trying it out!

5.2.1. Create display formats

The width and number of columns is controlled by using a Format. The example defines two format parameters one for the simple list and one for the graphic list.

You will see in the SA_Include project we commonly provide a format for HD1080 and UHD4K to accommodate different column widths for the font sizes (See "AlarmPage_GetDefaultParameters" in [SA_Include]\AlarmPage.ci.
To this:

1. In Citect Studio, Navigate to Setup > Parameters activity
2. Enter two formats as shown below:

←

Setup

Alarming

Parameters

Screen Profiles

Devices

Events

Languages

Keyboard Keys

Custom Fil

Save

Discard

Copy

Paste

Delete Row(s)

Export All

Import All

Row	Section Name	Name	Value	Comment	Project
1	FORMAT	SimpleListFormat	{FlightCode,100}{Origin,100}{Destination,100}	Simple List Column Format	Example_CustomList
2	FORMAT	GraphicListFormat	{Graphic,120}{Value1,50}{Value2,50}	Graphic List Column Format	Example_CustomList

The names such as 'FlightCode' are the names of our columns which will be displayed at runtime.

3. [Save] the changes

5.2.2. Implement the Initialisation function

Listviews require an initialisation function which must be called from the `On Page Entry` event on the page they placed on.

The example uses a technique where a map is used to pass parameters controlling the presentation of the list. In this case the parameters such as list width and font size is changed based on the resolution.

1. Create a new cicode file in your project. E.g. "GraphicList.ci"
2. Create a new function with the following code

```

FUNCTION GraphicList_Init(STRING hParameters)

    INT nItemListAN = DspGetAnFromName("GraphicList");
    INT hListViewArray;
    INT nListWidth;
    INT nListRowHeight;
    INT nListHeight;
    INT hRowFont;
    INT nCol;
    STRING sListFormat;

    // Adjust the listview based upon the parameters
    nListWidth = MapValueGet(hParameters, g_sGraphicListParamListWidth);
    nListRowHeight = MapValueGet(hParameters, g_sGraphicListParamListRowHeight);
    nListHeight = MapValueGet(hParameters, g_sGraphicListParamListRowCount) * nListRowHeight;
    hRowFont = DspFontHnd(MapValueGet(hParameters, g_sGraphicListParamListFont));

    MapClose(hParameters);

    hListViewArray = GenericListCreate(nItemListAN, nListWidth - 14, nListHeight, nListRowHeight, TRUE,
    "GraphicListFormat", hRowFont); // -14 is for scrollbar

    // Setup the columns
    ArraySetString(hListViewArray, ARRAY_COLUMN_TYPE_NORMALGENIE, 1, ARRAY_INFORMATION_ROW,
    ARRAY_ZINDEX_COLUMN_TYPE); // Set Column 1 to be a genie
    ArraySetInt(hListViewArray, ARRAY_COLUMN_TYPE_NUMBER, 2, ARRAY_INFORMATION_ROW,
    ARRAY_ZINDEX_COLUMN_TYPE); // Set Column 2 to be a int
    ArraySetInt(hListViewArray, ARRAY_COLUMN_TYPE_NUMBER, 3, ARRAY_INFORMATION_ROW,
    ARRAY_ZINDEX_COLUMN_TYPE); // Set Column 3 to be a int

    // Create an array to store our model. Normally you wouldn't do this here, instead perhaps it is
    something you create on startup of citect
    // and pass in as a parameter to the Init function, or as a global variable.
    PageSetHandle(1_sDataModelPageHandle, ArrayCreate("GraphicList_Items", 1_nDataModelAttributeCount,
    1_nDataModelItemCount));

    _GraphicList_AddModelData();

    // Mark the view as dirty so it gets rendered the first time
    ArraySetIsDirty(hListViewArray, TRUE);

END

STRING
FUNCTION GraphicList_GetDefaultParameters(STRING sResolution)

    STRING sMap = MapOpen("", 0);

    IF (sResolution = "HD1080") THEN
        MapValueSet(sMap, g_sGraphicListParamListWidth, 450);
        MapValueSet(sMap, g_sGraphicListParamListRowHeight, 28);
        MapValueSet(sMap, g_sGraphicListParamListRowCount, 7);
        MapValueSet(sMap, g_sGraphicListParamListFont, "SA_ListRow");
    ELSE
        MapValueSet(sMap, g_sGraphicListParamListWidth, 651);
        MapValueSet(sMap, g_sGraphicListParamListRowHeight, 40);
        MapValueSet(sMap, g_sGraphicListParamListRowCount, 13);
        MapValueSet(sMap, g_sGraphicListParamListFont, "SA_ListRow4K");
    END

    RETURN sMap;

END

```

To simplify this example, the “datasource” is also initialised in line with the list. In practice you would separate this as you do not want your data source being duplicated everytime you create a list! See the treeview as an example.

5.2.3. Implement an Exit function

Listviews require an exit function which must be called from the On Page Exit event on the page they placed on.

This function performs critical resource cleanup via the GenericListDestroy call. In the code below, the ArrayDestroy is only required because we are creating our datasource in the Init.

In the same cicode file "GraphicList.c"

- 1. Create a new function with the following code:

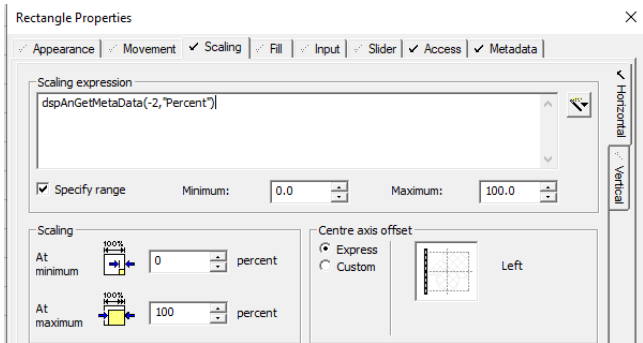
```
FUNCTION GraphicList_Exit()  
  
  INT nItemListAN = DspGetAnFromName("GraphicList");  
  INT hModel = PageGetHandle(1_sDataModelPageHandle);  
  
  // Destroy arrays (maps will be destroyed by PageObject)  
  GenericListDestroy(nItemListAN);  
  ArrayDestroy(hModel);  
  
END
```

5.2.4. Create a genie to represent your data

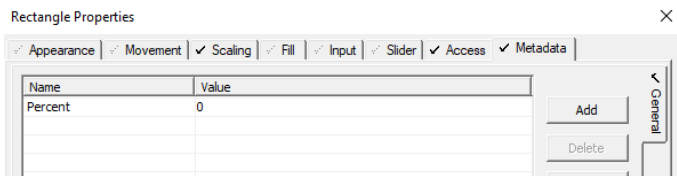
- 1. Switch to Graphics Builder
- 2. Create a new genie as follows and ensure the two rectangles are named as such:



- 3. Display the properties for 'Bar1'
- 4. Configure the scaling expression as follows:



- 5. Switch to the Metadata tab and configure the following metadata:



6. Repeat steps 4 and 5 for the Bar2' rectangle object
7. [Save] the genie in library 'example' with a name of 'bar'. This is important as we will be referencing the library and name in the next section.

5.2.5. Implement Fill function

The fill function is called by Citect Runtime when it needs to render the list. The purpose of the fill function is to fill the view array attached to the listview genie AN with the data that needs to be rendered.

It is called in response to scrolls etc.

The fill function is designed to gather data from a model for just the rows it needs to render. For example, you may have a data source of 1000 rows, but the listview only needs to show rows 500-520.

In the same cicode file "GraphicList.ci"

1. Create a new function with the following code

```

FUNCTION GraphicList_Fill(INT nListAN)

    INT hModel = PageGetHandle(l_sDataModelPageHandle);
    INT hView = ArrayGetArrayByAn(nListAN);
    INT nModelRows;
    INT nViewRows;
    INT nViewRow;
    INT nModelRow;
    INT nOldModelRow;
    INT nSelectedRow;
    INT nValue1;
    INT nValue2;

    IF (hModel = BAD_HANDLE OR hView = BAD_HANDLE) THEN
        RETURN;
    END

    nModelRows = ArrayGetInfo(hModel, 1);
    nViewRows = ArrayGetInfo(hView, 1);

    nViewRow = 1;
    nModelRow = _ArrayView_YOffset(nListAN);

    nSelectedRow = StrToInt(_ArrayView_Selected_RecID(nListAN));
    nOldModelRow = PageGetInt("__OldModelRow" + nListAN:);

    // Prevents the re-drawing when state is the same
    IF (NOT ArrayIsDirty(hView)) THEN
        RETURN;
    END

    // View has likely been scrolled so invalidate the view
    IF (nModelRow <> nOldModelRow) THEN
        ArraySetIsDirty(hView, TRUE);
    END

    ArraySetInt(hView, nModelRows, ARRAY_INFORMATION_COLUMN, ARRAY_INFORMATION_ROW,
    ARRAY_ZINDEX_ARRAY_YTOTAL);
    WHILE (nViewRow < nViewRows) DO

        IF (nModelRow < nModelRows) THEN

            nValue1 = ArrayGetInt(hModel, 1_iColumnValue1, nModelRow);
            nValue2 = ArrayGetInt(hModel, 1_iColumnValue2, nModelRow);

            // Column 0: Info
            ArraySetString(hView, nSelectedRow = nModelRow, ARRAY_INFORMATION_COLUMN, nViewRow,
            ARRAY_ZINDEX_ROW_SELECTED);
            ArraySetString(hView, nModelRow:#, ARRAY_INFORMATION_COLUMN, nViewRow,
            ARRAY_ZINDEX_ROW_ID);

            // Column 1: Genie
            ArraySetString(hView, "Example.Bar", 1, nViewRow, ARRAY_ZINDEX_CELL_VALUE);
            // The genie has two graphics objects on it. This allows us to set the meta data
            properties of each
            ArraySetString(hView, ".Bar1|Percent|" + nValue1:# + "|" + ".Bar2|Percent|" + nValue2:#, 1,
            nViewRow, ARRAY_ZINDEX_CELL_META1);

            // Column 2: Value
            ArraySetInt(hView, ArrayGetInt(hModel, 1_iColumnValue1, nModelRow), 2, nViewRow,
            ARRAY_ZINDEX_CELL_VALUE);

            // Column 3: Value
            ArraySetInt(hView, ArrayGetInt(hModel, 1_iColumnValue2, nModelRow), 3, nViewRow,
            ARRAY_ZINDEX_CELL_VALUE);

        ELSE

```

```
// No more items in model, clear remaining view rows. We don't use the meta data column in
this example, good practice to wipe it regardless. It is used for genie column type
// This would occur if you had a view which could display 10 rows, but only 5 lines of data
in the model.
    ArraySetString(hView, FALSE, ARRAY_INFORMATION_COLUMN, nViewRow,
ARRAY_ZINDEX_ROW_SELECTED);
    ArraySetString(hView, "-1", ARRAY_INFORMATION_COLUMN, nViewRow, ARRAY_ZINDEX_ROW_ID);
    ArraySetString(hView, "", 1, nViewRow, ARRAY_ZINDEX_CELL_VALUE);
    ArraySetString(hView, "", 1, nViewRow, ARRAY_ZINDEX_CELL_META1);
    ArraySetString(hView, "", 2, nViewRow, ARRAY_ZINDEX_CELL_VALUE);
    ArraySetString(hView, "", 2, nViewRow, ARRAY_ZINDEX_CELL_META1);
    ArraySetString(hView, "", 3, nViewRow, ARRAY_ZINDEX_CELL_VALUE);
    ArraySetString(hView, "", 3, nViewRow, ARRAY_ZINDEX_CELL_META1);

    END

    nViewRow = nViewRow + 1;
    nModelRow = nModelRow + 1;

    END
END
```

5.2.6. Implement Click Handlers

5.2.7. Trying it out!

The “Example_CustomList” project demonstrates the steps in this tutorial.

- 1. Run the computer Setup Wizard and select the “DEFAULT” profile.
- 2. Run Citect SCADA up and you will see the treeview demonstration page
- 3. Try: expanding / collapseing and searching for food items.
- 4. Try: Left clicking on the node text.

6. General Workspace Customisations

6.1. Adding an Alarm Banner to a Master Page

Alarm Banners are designed to display the most recent active alarms in the operators purview – irrespective of context of the workspace.

P...	On Date	On Time	Name
▲	12/09/2018	02:23:05 PM	PMP001 Overheated
▲	12/09/2018	02:23:00 PM	PMP002 Overheated

The size and position of your alarm banner is fully in your control. The out-of-the-box master pages of the Situational Awareness starter project show two ways to include an alarm banner:
Popup – as seen in the HD master page (Accessed by the Top 5 button in the header)
In place – as seen by the UHD master page (bottom left of the master page)

In this example we are going to show you how to include one n your page and customise its size. To do this we will need to:

- Create an Alarm Banner Page
- Insert an Alarm List on the Page
- Configure the Alarm List
- Set the Alarm Banner Page as the Default Page

Before we get started, this tutorial assumes you have already created yourself a master page and have allocated a pane to host your alarm banner.

6.1.1. Create an Alarm Banner Page

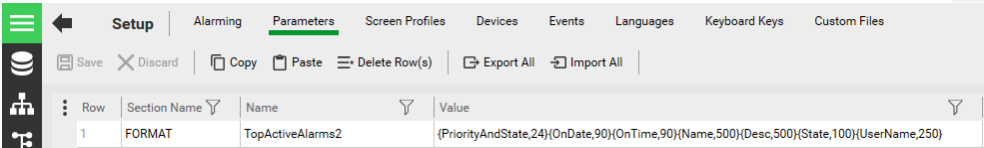
The Alarm Banner page consists of an Alarm List Control and an alternating list row genie to make readability of alarm data easier.

1. Create a new page based on the Blank HD template
2. Paste the “list_row_x10” genie from the sa_controls onto your page and postion at 0,0. (If your alarm list needs more than 10 rows, use the “list_row_x40” genie instead.)
3. Paste the Treeview genie from the sa_controls library onto your page at position at 0,0
4. Go to the File menu and select “Properties” and then click the Appearance Tab
5. Change the Width and Height properties to match your Alarm Banner pane in your Master Page
6. While still in the Properties Click on the “Events” Tab
7. Configure the “On Page Entry” as follows: (Change 720 to the width of your page and 260 to the height of your page)

```
_ArrayView_Initialize();  
_AlarmListCreate(DspGetAnFromName("AlarmList"), 0, 720-14, 260, 20,  
1, "", "TopActiveAlarms2", -1, DspFontHnd("SA_AlmHeaderB"))
```
8. Configure the “On page Exit” as follows

```
AlarmListDestroy(DspGetAnFromName("AlarmList"))
```
9. Configure the “On page Shown” as follows:

```
Alarm_SortList(DspGetAnFromName("AlarmList"), "", TRUE)  
AlarmSetInfo(DspGetAnFromName("AlarmList"), 6, DspFontHnd("SA_AlmRow"))
```
10. Note that in step 7 we used the value “TopActiveAlarms2” for the alarm row format. We need to define that. Go to Citect Studio and navigate to the Setup > Parameters Activity. Configure it as follows:

The screenshot shows the Citect Studio Parameters table. The table has four columns: Row, Section Name, Name, and Value. There is one row with Row number 1, Section Name 'FORMAT', Name 'TopActiveAlarms2', and Value '{PriorityAndState,24}{OnDate,90}{OnTime,90}{Name,500}{Desc,500}{State,100}{UserName,250}'. The 'Parameters' tab is selected in the top menu.

Row	Section Name	Name	Value
1	FORMAT	TopActiveAlarms2	{PriorityAndState,24}{OnDate,90}{OnTime,90}{Name,500}{Desc,500}{State,100}{UserName,250}

11. [Save] the page as “TopAlarms_HD1080”. (the name can be anything)

- 6.1.2. Insert an Alarm List on the Page
- 6.1.3. Configure the Alarm List
- 6.1.4. Set the Alarm Banner Page as the Default Page

6.2. Adding Alarm Limit Configuration to the Information Zone

ALARMTRENDINTERLOCKSETUP

Optimal / Practical Ranges

OR High0.00

OR Low0.00

Optimal Range

PR High0.00

PR Low0.00

Limit Adjustment

MyHH0.00

MyH0.00

MyL0.00

MyLL0.00

6.3. Adding a Tab to the Information Zone

The Information Zone can be customised to change the default tabs it displays.

The tabs that are displayed in the Information Zone are controlled by the ‘InformationZone’ menu in the Menu Configuration view of the Visualization Activity of Citect Studio.

VisualizationMenu ConfigurationPagesContent TypesKeyboard CommandsReports											
SaveDiscardCopyPasteDelete Row(s)Export AllImport All											
Row	Page	Level 1	Level 2	Level 3	Menu Command	Target Page	Comment	Order	Custom 1	Custom 2	
1	InformationZone	@(ALARM)				Info_Alarm	Shows the Alarm	1	1		
2	InformationZone	@(TREND)				Info_Trend	Shows the Trend	2	1		
3	InformationZone	@(INTERLOCK)				Info_Interlocks	Shows the Interlo	3	1		

Figure 6 - Information Zone Configuration

Additional tabs can be added by simply adding another row item for this menu as shown below:

Visualization										
Menu Configuration										
Pages										
Content Types										
Keyboard Commands										
Reports										
Save Discard Copy Paste Delete Row(s) Export All Import All										
Row	Page	Level 1	Level 2	Level 3	Menu Command	Target Page	Comment	Order	Custom 1	Custom 2
1	InformationZone	@(ALARM)				Info_Alarm	Shows the Alarm	1	1	
2	InformationZone	@(TREND)				Info_Trend	Shows the Trend	2	1	
3	InformationZone	@(INTERLOCK)				Info_Interlocks	Shows the Interlo	3	1	
	InformationZone	MyTab				MyPage	Shows the MyPag	4	1	

The ‘Order’ column can also be used to control the display order of the tabs in the user interface. In the example above, ‘MyTab’ has been assigned order ‘4’ which will place it at the end. To put it at the front change it to ‘1’ and then change the other tabs to be ‘2,3,4’ etc.

The ‘TargetPage’ page field indicates which page will be displayed when the operator selects the tab at runtime. If your system will support HD and UHD displays then you need to omit the resolution suffix from your page names. By default the Starter project for Situational Awareness contains Info Zone pages in two resolutions HD1080 and UHD4K. For example, “Info_Alarm_HD1080” and “Info_Alarm_UHD4K”. To support this capability just specify “Info_Alarm in the target page field.

The ‘Custom1’ field is used to control whether the tab can be closed at runtime. A value of ‘0’ means the tab can be closed, a value of ‘1’ means it cannot be closed. For the Information Zone, ‘Custom 1 should be set to ‘1’ – cannot close.

6.4. Adding an equipment Debug Info Zone Tab

ALARM	TREND	INTERLOCK	DEBUG
Item		Value	
AutoCmd		<input type="checkbox"/>	
Calib		<input type="checkbox"/>	
CasCmd		<input type="checkbox"/>	
CtrlMode		Automatic ▼	
CtrlModeDef		Automatic ▼	
EqStatus		0	

6.5. Collapsible Panes

The workspace can be expanded to support collapsible panes.

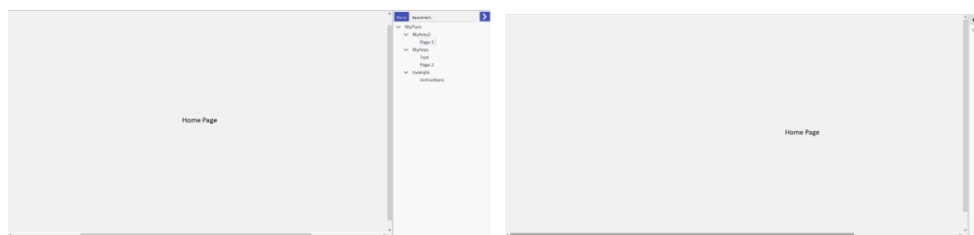


Figure 7 - A collapsible Pane

In the example above, the operator can click on the [>] button which will collapse the pane to a smaller version of itself. When it collapses, the mimic (content) pane to its left is expanded. The method that will be described here can be used to also hide/show panes.

Internally the workspace does not natively support automatic sizing of adjacent panes, instead you must manage that yourself.

To implement this design the following steps are required:

- Design your Master Page
- Write code to Toggle and Adjust Your Panes
- Bind your toggle function to a button within your Master Page

6.5.1. Design your Master Page

6.5.2. Write Code to Toggle Your Panes and Adjust the Panes

The panes must be manually adjusted. The order in which they are adjusted is critical as panes are NOT allowed to overlap each other at anytime. You must move your panes in an order that prevents overlap. The `WinMove`, `WinSize` functions all have protection for this and will error if an overlap occurs..

```

FUNCTION ToggleMenuPanel ()

    INT nWorkspace = Workspace_GetWorkspaceFromWindow(WinNumber());
    INT nWorkspaceWindow = Workspace_GetWindow();
    INT nNavigation = Workspace_GetPaneWindow(nWorkspace, "Navigation");
    INT nContent = Workspace_GetPaneWindow(nWorkspace, "Content");
    INT nMenubar = Workspace_GetPaneWindow(nWorkspace, "MenuBar");
    INT bCollapsed = FALSE;
    INT nPrevious;

    bCollapsed = PageGetInt("Collapsed", nWorkspaceWindow);
    PageSetInt("Collapsed", NOT bCollapsed, nWorkspaceWindow);

    IF (NOT bCollapsed) THEN
        nPrevious = WinSelect(nMenubar);
        WinMove(1870, 56, 1, 1);
        WinSelect(nNavigation);
        WinMove(1872, 100, 48, 856);
        WinSelect(nContent);
        WinSize(1868, 900);
        WinSelect(nPrevious);
    ELSE
        nPrevious = WinSelect(nContent);
        WinSize(1528, 900);
        WinSelect(nMenubar);
        WinMove(1532, 56, 340, 44);
        WinSelect(nNavigation);
        WinMove(1532, 100, 388, 856);
        WinSelect(nPrevious);
    END
END

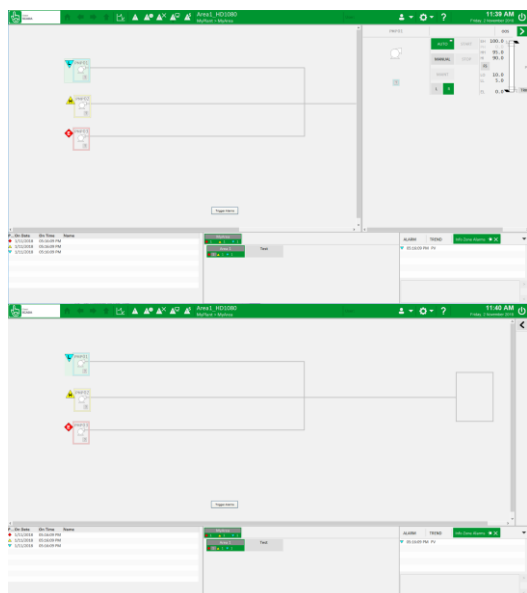
```

6.5.3. Bind your Toggle Function to a Command

In our example we are going to call the function by a button which is located in another pane within the Workspace. Open up the “MenuBarToggle_HD1080” page:

6.6. Auto Expanding and Collapsible Face Plate Pane

For some customers they have large and complex faceplates, they also do not need for them to always be on display. The workspace can be customised to dynamically display the faceplate on equipment selection, and allow the operator to close it at anytime.



In the example above, the operator can click on the [>] button which will collapse the pane to a smaller version of itself. When it collapses, the mimic (content) pane to its left is expanded and the Faceplate Header and Faceplate panes are 'hidden'.

If the operator clicks on any of the pump equipment that is not already selected, the faceplate will auto-expand.

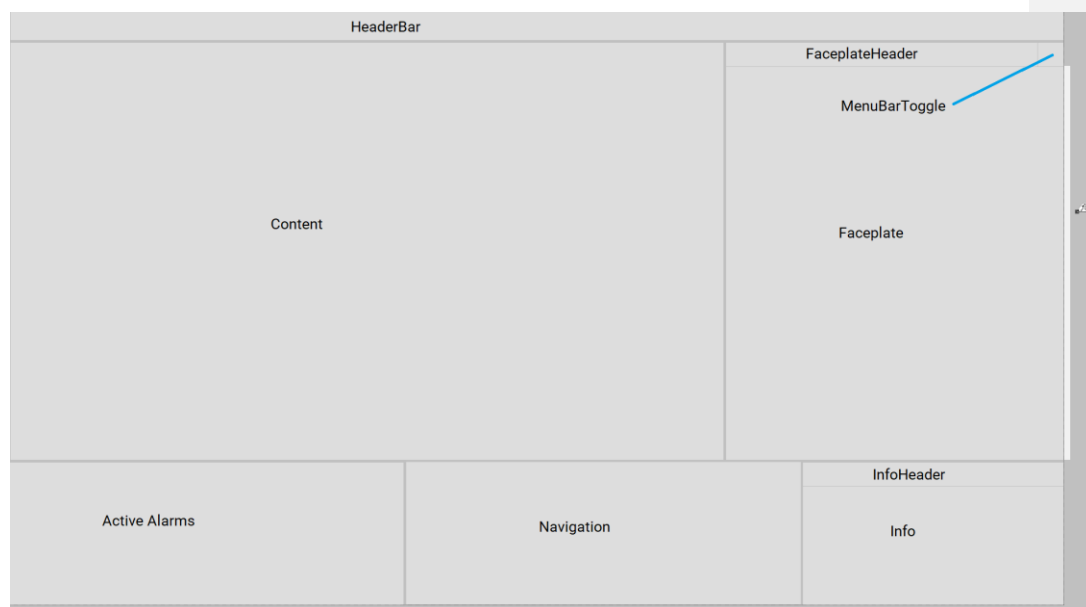
Internally the workspace does not natively support automatic sizing of adjacent panes, instead you must manage that yourself.

To implement this design the following steps are required:

- Design your Master Page
- Write code for your Master Page to monitor for context change and auto-expand the faceplate panes if not already expanded.
- Write code to Toggle and Adjust Your Panes
- Bind your toggle function to a button within your Master Page

6.6.1. Design your Master Page

To enable this behaviour a different layout Master Page needs to be created. The illustration below shows the new layout.



Note: It is an extremely important to ensure your panes **do not overlap**. If your panes do not display at any point. Turn on the [Debug]UserTraceMode =3 and check for pane creation errors or hardware alarms relating to overlapped windows.

The demonstration project contains an "Active alarms" page sized and configured for the Active Alarms Page. If you are creating this manually, you can use the "[SA_Include] TopActiveAlarms_UHD4K" page as the basis.

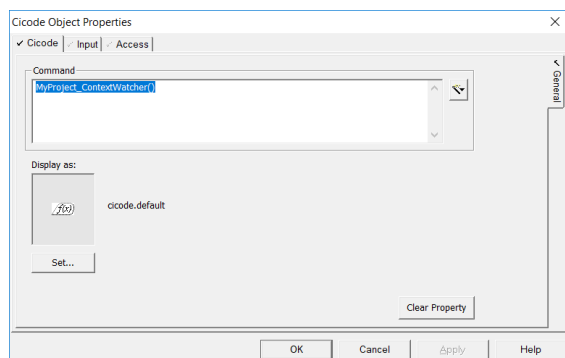
As the FacePlateHeader and FacePlate panes are sized differently to those for the standard Master Pages, you will need to ensure the content for these panes is optimially sized.

Document properties of MenuBarTogglePane

6.6.2. Monitor for Context Change

You can monitor for contextual changes on a Master Page using the "[Workspace_GetContext](#)" function. If you are within a pane you can use the more performant "[Workspace_GetSelContext](#)" function instead. To do this,

1. Add a Cicode Object to your Master Page
2. Configure the Command as follows



3. Write the function and add it to your project

```

FUNCTION MyProject_ContextWatcher ()

    INT bCollapsed = PageGetInt("Collapsed", WinNumber());
    STRING sEquipmentContext = Workspace_GetContext();
    STRING sLastEquipmentContext = PageGetStr("__CurEquipmentRef");

    IF (sEquipmentContext <> sLastEquipmentContext) THEN
        IF (bCollapsed) THEN
            // Don't expand if there is no equipment selected (e.g. transitioning to Alarm
Page)
            IF (sEquipmentContext <> "") THEN
                MyProject_ToggleFacePlatePanel();
            END
        ELSE
            // Collapse if there is no equipment selected. (e.g. transitioning to Alarm Page)
            IF (sEquipmentContext = "") THEN
                MyProject_ToggleFacePlatePanel();
            END
        END
        PageSetStr("__CurEquipmentRef", sEquipmentContext);
    END
END

```

The function works by storing a local copy of the current context and comparing it to the workspace's current selection.

When the selection differs, it will determine whether the faceplate should be expanded based upon whether the new context is blank or not. If blank it will keep it closed, or collapse it.

6.6.3. Bind your Toggle Function to a Command


```

FUNCTION MyProject_ToggleFacePlatePanel ()
    INT nWorkspace = Workspace_GetWorkspaceFromWindow(WinNumber());
    INT nWorkspaceWindow = Workspace_GetWindow();
    INT nFacePlate = Workspace_GetPaneWindow(nWorkspace, "FacePlate");
    INT nContent = Workspace_GetPaneWindow(nWorkspace, "Content");
    INT nFacePlateHeader = Workspace_GetPaneWindow(nWorkspace, "FacePlateHeader");
    INT bCollapsed = FALSE;
    INT nPrevious;

    bCollapsed = PageGetInt("Collapsed", nWorkspaceWindow);
    PageSetInt("Collapsed", NOT bCollapsed, nWorkspaceWindow);

    IF (NOT bCollapsed) THEN
        nPrevious = WinSelect(nFacePlateHeader);
        WinMove(1870, 56, 1, 1);
        WinSelect(nFacePlate);
        WinMove(1920, 100, 48, 716);
        WinSelect(nContent);
        WinSize(1868, 760);
        WinSelect(nPrevious);
    ELSE
        nPrevious = WinSelect(nContent);
        WinSize(1300, 760);
        WinSelect(nFacePlateHeader);
        WinMove(1306, 56, 566, 44);
        WinSelect(nFacePlate);
        WinMove(1306, 100, 616, 716);
        WinSelect(nPrevious);
    END
END

```

6.7. The Auto Close Popup Pattern

See Top 5 alarms / help Page

7. Creating Your Own Theme Combination

The workspace supports basic theming. By default we ship a number of themes out of the box such as Indigo, SEGreen, Black, Grey and in 2018 R2 "Blue".

A theme consists of:

- Header bar Colour
- Control colour

The set of colours for the header bar are fixed. They are Black, Green, Grey, Orange, Indigo and Blue

The set of colours for the control colours are also fixed. They are Green, Indigo, Orange and Blue

A custom theme involves selecting a Headerbar colour and a Control colour.

1. In Citect Studio, navigate to Standards > Labels
2. Change the label "_THEME_Default" to 5 or 6.
 - a. We only support 2 custom themes.
3. Create a Startup function for your Clients with the following code:

- a. Theme_SetHeaderColor(5, THEME_Header_Light)
 - b. Theme_SetHilightColor(5, THEME_Hilight_Blue)
- 4. [Save] the file
- 5. In your computers profile, ensure this function is called on startup of the client
- 6. Run up Citect

8. Printing the Content Pages in your System

The mimic content pages of your system can easily be printed out for operator review etc. by using a custom print function at runtime.

The following code loops through all pages in your project and prints them to a printer. There are 3 parameters which can be configured in your project:

Parameter	Allowable Values	Description
[Workspace]Printer	UNC printer name or port name	UNC printer name or port name of the printer to use
[Workspace]PrinterExcludeCT	Comma separated list of Content Types. Default is FP,__Internal,Master,General	List of Content Types to exclude. If a page has one of these content types it will not be printed.
[Workspace]PrinterSwapBW	0 (default) or 1	Swaps the colors black and white for the purpose of printing pages with a lot of black content. A value of 1 will swap black and white.

```

FUNCTION Utilities_PrintContentPages()
    INT hPagesItem;
    INT hRdb;
    INT nRecord;
    INT nItemCount = 0;
    INT nContentWindow;
    INT nPrevious;
    INT nWorkspace = Workspace_GetWorkspaceFromWindow(WinNumber());
    STRING sPageName;
    STRING sPrinter;
    STRING sContentTypeFilter;
    INT bSwapBlackWhite;

    sPrinter = ParameterGet("Workspace", "Printer", "PORTPROMPT:");
    sContentTypeFilter = ParameterGet("Workspace", "PrinterExcludeCT",
"FP, _Internal, Master, General");
    bSwapBlackWhite = ParameterGet("Workspace", "PrinterSwapBW", "0");

    nContentWindow = Workspace_GetPaneWindow(nWorkspace, "Content")
    nPrevious = WinSelect(nContentWindow);

    hRdb = RdbOpen(" _PAGES");
    IF (hRdb <> -1) THEN
        nRecord = RdbFirstRec(hRdb);
        WHILE (nRecord <> -1) DO
            sPageName = RdbGet(hRdb, "NAME");

            IF (_Utilities_IsContent(sContentTypeFilter, sPageName)) THEN
                _Utilities_NavigateAndPrint(sPrinter, bSwapBlackWhite, sPageName);
            END

            nRecord = RdbNextRec(hRdb);
        END
        RdbClose(hRdb)
    END

    WinSelect(nPrevious);
END

```

```

PRIVATE
INT FUNCTION _Utilities_IsContent(STRING sContentTypeFilter, STRING sPageName)

    STRING sContentType = PageFileInfoEx(sPageName, 2);

    // Omit excluded content type pages and the Include project pages
    IF (StrListContainsItem(sContentType, sContentTypeFilter) OR (sPageName = "!sgl_xpitune") OR
(sPageName = "!Startup") OR (sPageName = "PAGEMENU") OR (sPageName = "STANDARD") ) THEN
        RETURN FALSE;
    END

    // Page has not met any rules for exclusion, add it to the default menu
    RETURN TRUE;
END

```

```
FUNCTION _Utilities_NavigateAndPrint (STRING sPrinter, INT bSwapBlackWhite, STRING sPageName);  
    // Only allow a single thread to run this code, as it may effect multiple pages when clearing  
    the context  
    EnterCriticalSection("Navigation_ShowTargetPage");  
  
    Workspace_ShowContent (sPageName);  
  
    SleepMS (750);  
  
    WinPrint (sPrinter, 0, 0, bSwapBlackWhite);  
  
    LeaveCriticalSection("Navigation_ShowTargetPage");  
END
```



© 2018 AVEVA Group plc and its subsidiaries. All rights reserved.
AVEVA, the AVEVA logos and AVEVA product names are trademarks or registered trademarks of aveva group plc or its subsidiaries in the United Kingdom and other countries. Other brands and products names are the trademarks of their respective companies.

AVEVA Group plc
High Cross, Madingley Road
Cambridge CB3 0HB, UK
Tel +44 (0)1223 556655
Fax +44 (0)1223 556666

aveva.com