# finPOWER Connect 3
# Portals

Version 1.21

16th June 2021

# Table of Contents

# Disclaimer

This document contains information that may be subject to change at any stage.

All code examples are provided "as is".

This document may reference future functionality not currently available in the release version of finPOWER Connect.

# Version History

| Date | Version | Name | Changes |
|---|---|---|---|
| 21/09/2017 | 1.00 | PH | Created. |
| 11/01/2019 | 1.01 | PH | Updated for new Portal functionality. |
| 10/04/2019 | 1.02 | PH | Updated Partial Page section. |
| 15/05/2019 | 1.03 | PH | Client Sign-In Method added. |
| 08/08/2019 | 1.04 | PH | Updated portal.Navigate example. |
| 08/10/2019 | 1.05 | PH | Tag Overrides, Document Manager File, version 3.02.03 Login Theme options, Portal Host. |
| 06/11/2019 | 1.06 | PH | Page Tag1, Tag2 and Tag3 added. |
| 13/01/2020 | 1.07 | PH | Custom error pages added. |
| 14/01/2020 | 1.08 | PH | PWA properties and method for portal object. |
| 21/01/2020 | 1.09 | PH | Date-related tags added. |
| 29/01/2020 | 1.10 | PH | Updated 401 sample page code. |
| 07/02/2020 | 1.11 | PH | Added Using HTML Widgets in Portals section. |
| 25/02/2020 | 1.12 | PH | Version 3.03 additions such as Client Preferred Name. |
| 03/06/2020 | 1.13 | PH | portal.Navigate2 method added. |
| 29/07/2020 | 1.14 | PH | User Data plus Theme Logo Url tags added. |
| 17/08/2020 | 1.15 | PH | portal.GetPlatformCompatibilityWarning method, customising CSS for login form, defaults for Constant and UserData tags. PASSWORD_RESET_EMAIL special resource. |
| 12/10/2020 | 1.16 | PH | Theming enhancements. Partial Pages, Script Callbacks section. User Data Field for authentication. Performance and Best Practices. Portal authentication page changes. |
| 06/11/2020 | 1.17 | PH | Special Page Codes section. |
| 05/01/2021 | 1.18 | PH | Forced password change plus other changes for 3.03.05 |
| 08/04/2021 | 1.19 | PH | PartialPageId tag. |
| 08/06/2021 | 1.20 | PH | MFA added, Authentication page renamed to Security. |
| **16/06/2021** | **1.21** | **PH** | **Cookies section added plus enhanced hosting information.** |

# Introduction

This document discusses finPOWER Connect Portals.

It assumes a knowledge of the following:

- The finPOWER Connect business layer
- finPOWER Connect HTML Widgets, which includes:
  - HTML
  - JavaScript
  - Web development (client-side vs server-side Scripting)
  - CSS
  - jQuery

## External Hosting Considerations

When hosting a Portal, you should be aware of the following:

- Ideally, you will use the Portal Host application to access the Portal rather than having a direct link to Web Services.

## Resources

There is also limited intellisense on the `portal` and `page` JavaScript objects and a "finPOWER Connect Widgets and Portals" Help file available via Alt+F1 when editing the Portal's Master Page HTML or a Page's content.

## Cookies

Like most web applications, finPOWER Connect Portals uses cookies to some degree.

**IMPORTANT:** Our cookies are not used for tracking. They are used to provide a login mechanism and to enhance the user experience.

In Chrome or Microsoft Edge, you can see the cookies that we use by pressing F12, selecting the Application tab and looking at the "Cookies" entries:


Under **"Cookies"**, you may see the following:

- _highDpi
  - This will be set to "true" or "false" depending on whether a high-DPI (retina) display has been detected. This is sent with every request to ensure that higher resolution versions of images and icons are delivered where possible.
- __connected

- This simply holds a "1" or "0" and is used to provide a better user experience, e.g., to vary any 401 messages and to allow the Portal to be signed out automatically when a browser tab is closed.

- finCC_SessionId
  - This is simply the ASP.NET Session Id. By default, this cookie would be named "ASP.NET_SessionId". We simply assign it a custom name.
- *[PortalId]*.O, e.g. "CC.O"
  - This entry stores encrypted information relating to the current Client or User's login status.

**NOTE:** When we utilise external libraries such as Google Charts or Google Addressing, these may create their own cookies. These are nothing to do with finPOWER Connect Cloud.

# Overview

Portals represent Web applications that can be designed within finPOWER Connect and then used either internally (within finPOWER Connect) or, more importantly, externally via Web Services.

Examples or what Portals may be used for are:

- **Client Portal**
  - E.g., Clients can access the Portal via a Web Browser and view a list of their Accounts.
- **Dealer/ Broker/ Agent Portal**
  - External Party Users can sign in, add Account Applications and view their Tasks.
  - **NOTE:** It is strongly recommended that you consider finPOWER Connect Cloud before embarking on creating a Portal that targets finPOWER Connect Users.
- **Public Loan Application Portal**
  - Typically, a Loan Application form can be implemented as an HTML Widget. However, using a Portal to provide this functionality may be advantageous in certain situations.

**WARNING:** Portals are not designed to cope with the load that a public-facing Website may undergo, hence, they should be limited to the sub-set of functionality that requires finPOWER Connect, e.g., a Loan Application process.

# Portals Form

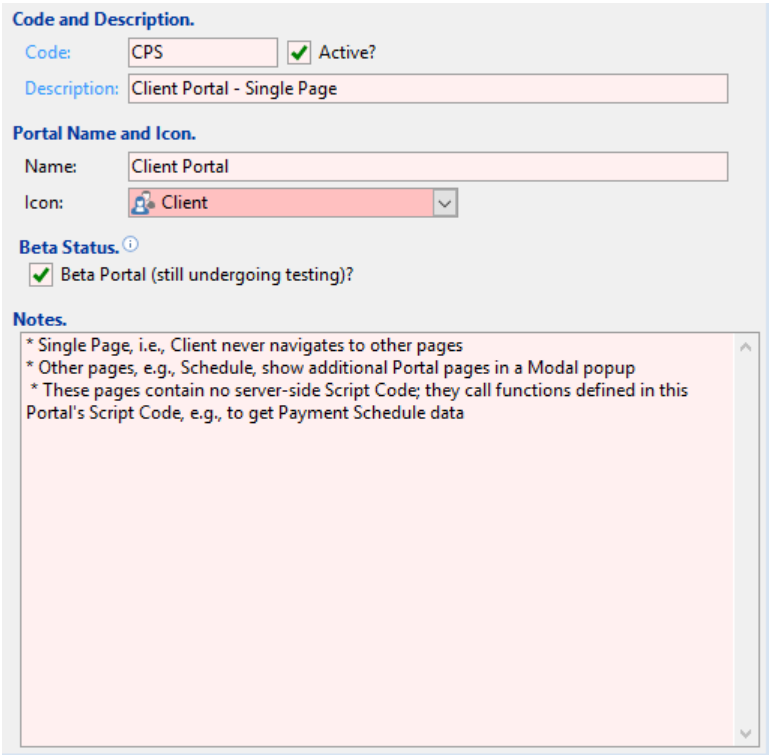
Portals are maintained via the Admin, Portals form.

Like most other Admin libraries, Portals can be Exported and Imported. However, a special "Export Package" action exists for Portals. This will create an export package that contains not only the portal but also any Scripts that are referenced such as HTML Widgets via `<%widget%>` tags or `portal.insertWidget` JavaScript calls.

**NOTE:** Packages must be imported via the File, Import/ Export Information, Import Information wizard.

## General

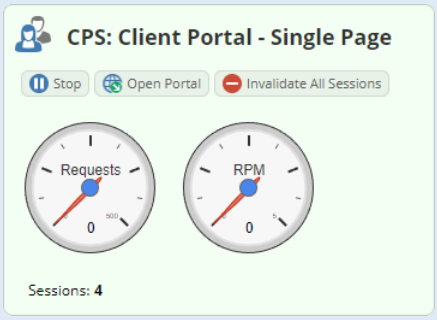
General options.




- **Code and Description**
  - Code:
    - A unique code up to 10 characters long.
  - Active:
    - Inactive Portals cannot be accessed via Web Services.
  - Description:
    - A description of the Portal, up to 50 characters long.
- **Portal Name and Icon**
  - Name:
    - The Portal Name if you want this to be different to the Descripiton.
  - Icon:
    - This Icon to display when running the Portal from within finPOWER Connect.
    - This will also be the Portal's Favourite Icon when running from Web Services unless a special Resource named 'favicon' is defined on the Resources page.

- **Beta Status**
  - Beta Portal (still undergoing testing)?
    - Just a way to flag that the Portal is not ready for production use.
- **Notes**
  - Notes regarding the Portal.

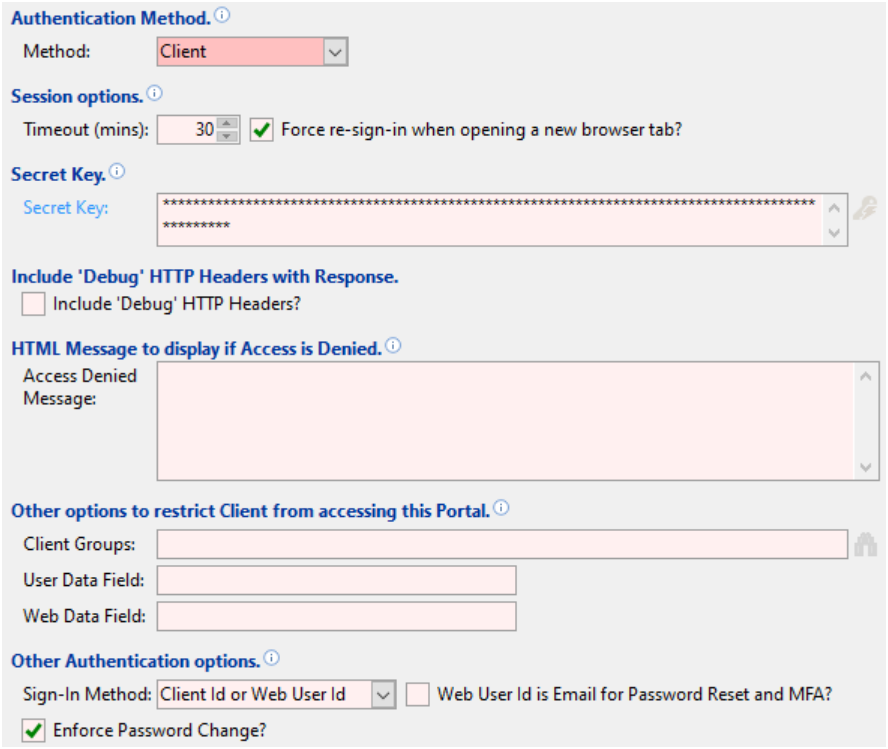
**NOTE:** Making a Portal inactive prevents Users from accessing it. This can also be achieved within Web Services Administration using the Portal's "Stop" button:

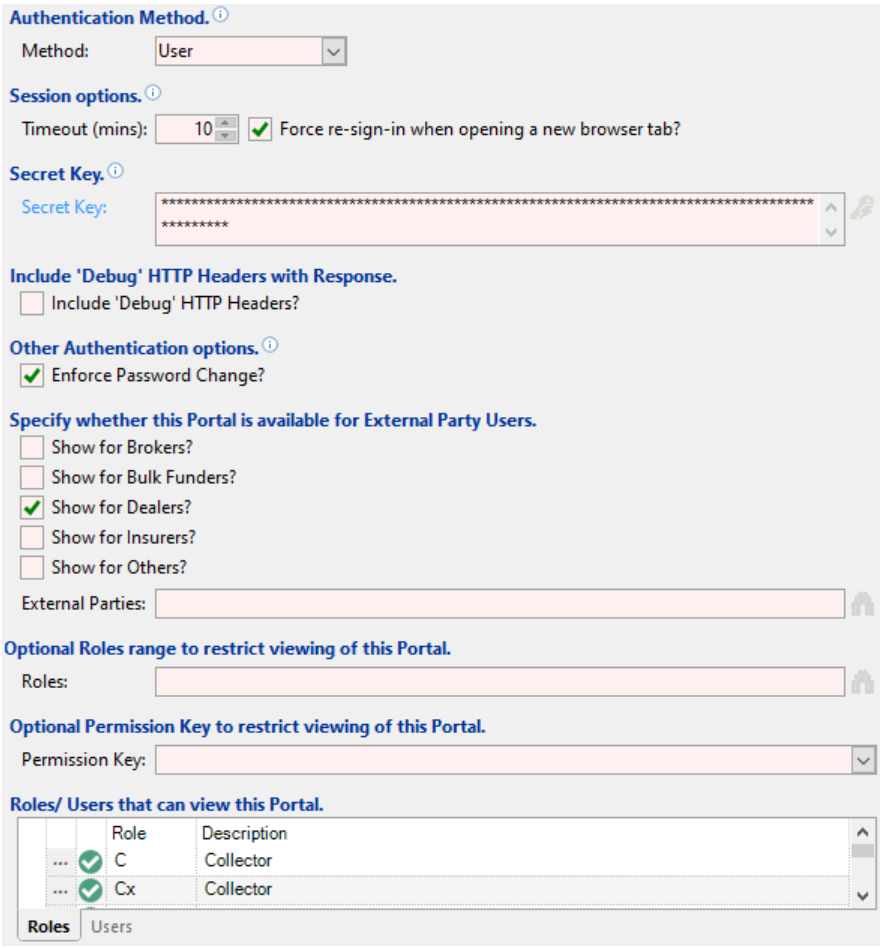


# Security

Defines the type of user that this Portal targets and other security related options.

The information available depends on what "Method" is selected.




- **Authentication Method**

- Method:
  - This determines who can access this Portal:
    - User – a finPOWER Connect User, including External Users such as Dealers.
    - Client – a finPOWER Connect Client.
    - Unauthenticated – anyone, e.g., a publicly available Loan Application wizard.

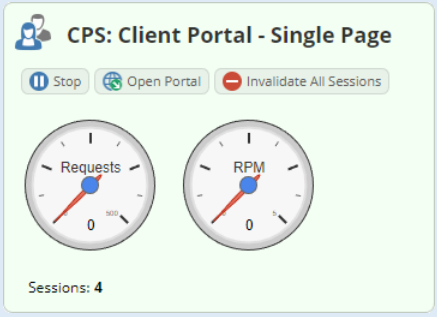
- **Session options**
  - Timeout:
    - The Session Timeout for Portals running outside of finPOWER Connect.
  - Force re-sign-in when opening a new browser tab?
    - Forcing re-sign-in when opening a new browser tab means that if a user closes their browser (or browser tab) then opening another tab to a Portal URL will redirect to the sign-in form, it is not related to the Session Timeout.

      **WARNING:** Browser behaviour such as Chrome's 'Duplicate' and 'Reopen closed tab' may not respect this setting since they preserve all of the original page's data when opening a tab.
- **Secret Key**
  - Secret Key:
    - A string of characters at least 30 characters long that is used when encrypting access to the Portal.

**NOTE:** Changing the Secret Key will invalidate any Clients or Users accessing this Portal outside of finPOWER Connect.

This will not sign the users out but will prevent any further interaction with the Portal without re-signing in.

This can also be achieved within Web Services Administration using the Portal's "Invalidate All Sessions" button:




## Client-Specific

- **HTML Message to display if Access is Denied**
  - Allows an HTML formatted message to be defined. This will be displayed to the Client if their credentials are correct, but they are denied access due to restrictions defined in the next section.

    **NOTE:** This message is only used when signing in to the Portal via Web Services.
- **Other options to restrict Client from accessing this Portal**
  - Client Groups:
    - A range of Client Groups that prevents Clients NOT in this range from accessing the Portal.

- User Data Field:
  - The name of a Boolean field stored in the Client's User Data. If this field does not exist or is not True then the Client will not be able to access this Portal.
- Web Data Field:
  - The name of a Boolean field stored in the Client's Web User Data. If this field does not exist or is not True then the Client will not be able to access this Portal.

- **Other Authentication options**
  - Sign-In Method:
    - Defines how a Client can Sign In.
      - Client Id or Web User Id
        - The Client can use either their Client Id or Web User Id to sign in.
      - Client Id
        - The Client can use only their Client Id to sign in.
      - Web User Id
        - The Client can use only their Web User Id to sign in.
          - **NOTE:** This may be desirable, e.g., where a Client's Web User Id is an Email address which a Client is much more likely to remember than their finPOWER Connect Client Id.
  - Web User Id is Email for Password Reset and MFA?
    - Indicates whether Client's are signing in using their Web User Id which is also an Email address that can be used to send Password Reset links and used for Multi-Factor Authentication.
  - Enforce Password Change?
    - When checked, the Portal will automatically display the Change Password form to the Client upon signing in if the Client's "Force Client to change their Web Password the next time they sign in" option is checked.

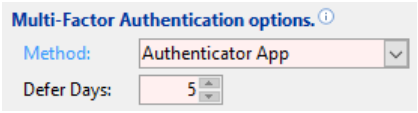
## User-Specific

Most of these options are identical to those on the finPOWER Connect Cloud Configuration form.

- **Other Authentication options**
  - Enforce Password Change?
    - When checked, the Portal will automatically display the Change Password form to the User upon signing in if the User's "Force User to change Password when they next login " option is checked.

# Multi-Factor

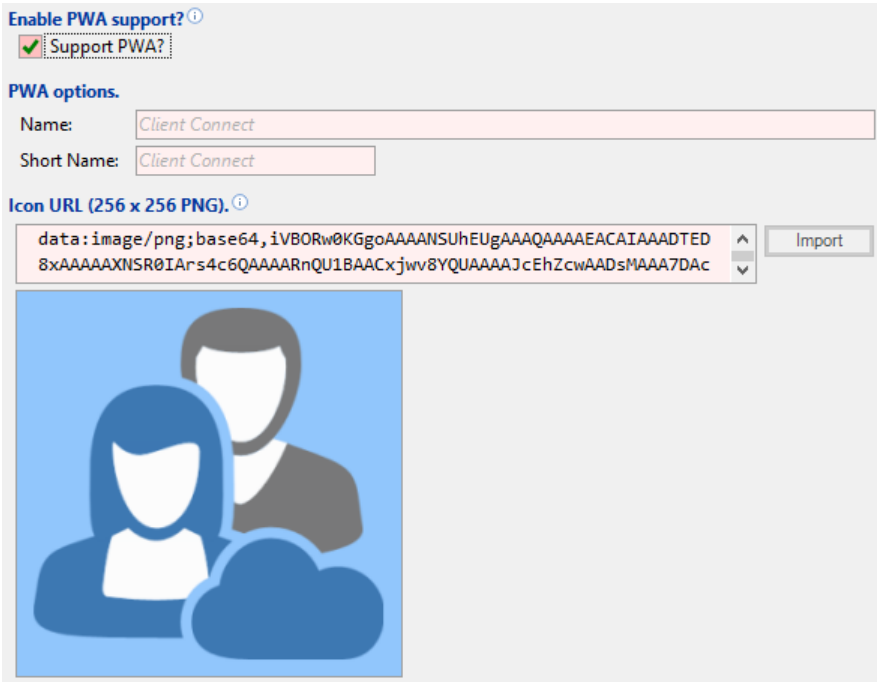
As of version 3.04 of finPOWER Connect, Portals support Multi-Factor Authentication (MFA) which is often referred to as Two-Factor Authentication (2FA).




- **Multi-Factor Authentication options**
  - Method:
    - The Multi-Factor Authentication Method to use for this Portal:
      - None
      - Email Code
        - Every time the User/ Client signs in they will be emailed a code that they must then enter to complete the sign-in process.
      - Authenticator App
        - Upon first signing in, the User/ Client will be prompted to use an Authenticator App (such as Google Authenticator) to scan a QR code. For future sign-ins, they will be prompted to enter the 6-digit code displays in this App.
      - SMS Code
        - Every time the User/ Client signs in they will be sent an SMS message containing a code that they must then enter to complete the sign-in process.
  - Defer Days:
    - If this is non-zero, Clients and Users logging into the Portal will have the option to "Skip this step for *X* days?"
      - If this option is checked then the Client or User will be able to sign in using just their User/ Client Id and Password for the specified period.

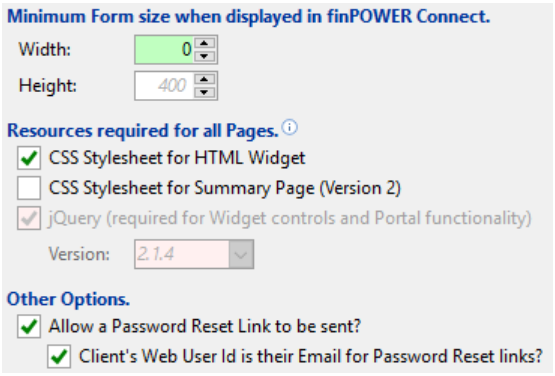
# PWA

Allow to Portal to support Progressive Web App functionality. This means that the Portal can be displayed as a shortcut on a device's home screen (e.g., an iPhone) and appear like a native App.



# Options

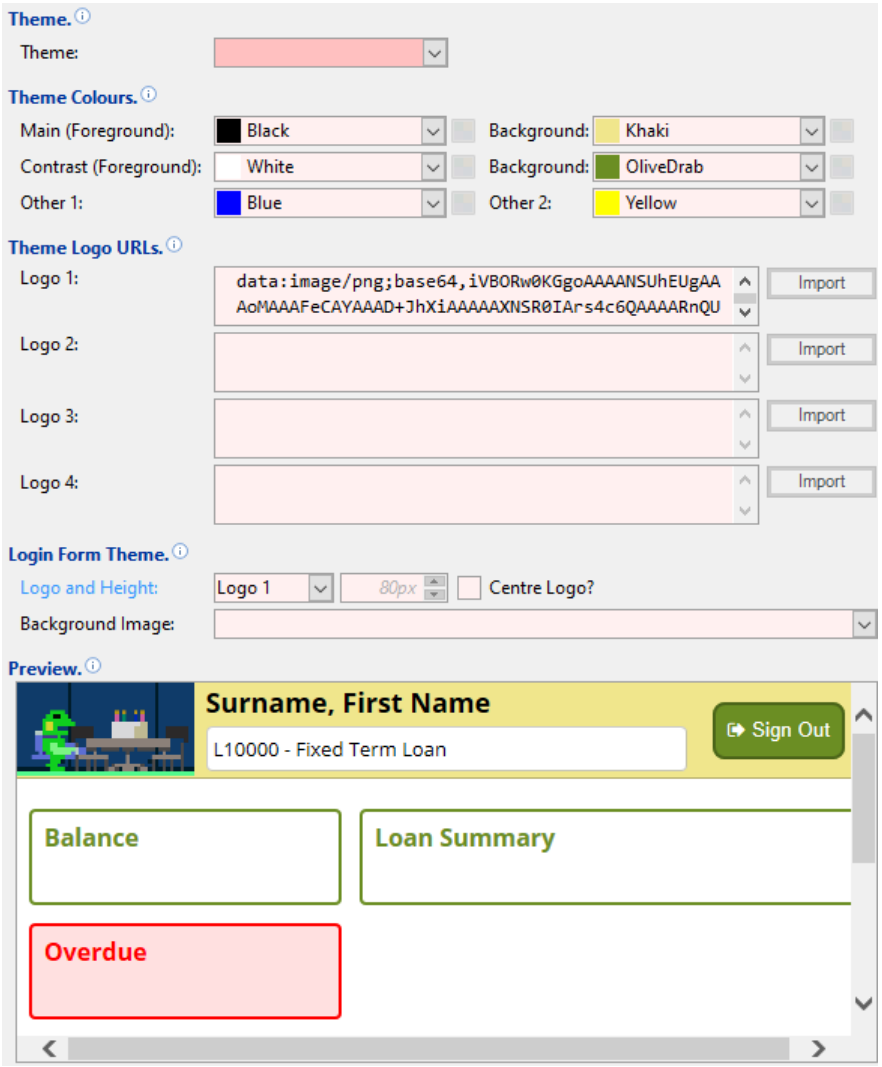
Miscellaneous options.




- **Minimum Form size when displayed in finPOWER Connect**
  - Width:
    - The minimum width (in pixels) of the inside area of the Portal form when the Portal is displayed within finPOWER Connect.
  - Height:
    - The minimum height (in pixels) of the inside area of the Portal form when the Portal is displayed within finPOWER Connect.
- **Resources required for all Pages**
  - CSS Stylesheet for HTML Widgets
    - Indicates whether to include the HTML Widgets Stylesheet for ALL Portal Pages.
  - CSS Stylesheet for Summary Page (Version 2)
    - Indicates whether to include the Summary Page (Version 2) Stylesheet for ALL Portal Pages.
  - jQuery (required for Widget controls and Portal functionality)
    - jQuery will ALWAYS in included since it is required for Portal and HTML Widget functionality.
- **Other Options**
  - Allow a Password Reset Link to be sent?
    - Indicates whether a Client or User Portal should allow a Password Reset Link to be sent via email.

# Theme

Allows colours and logo images to be defined that can be used via <%Theme%> tags, e.g., within a CSS file.




- **Theme**
  - The value defined here can be dynamically included within <%Resource%> tags using the special [THEME] tag, e.g.:

    <%Resource:[THEME].theme%>

    <%Resource:CSS_[THEME]%>
- **Theme Colours**
  - This allows various colours to be defined that can be used via <%Theme%> tags, e.g., within a CSS file or LESS file.
  - **WARNING:** Avoid using system colours such as "ActiveBorder", particularly when using LESS since the LESS compiler may fail and these colours are typically only supported under Windows Internet Explorer.
- **Theme Logo URLs**
  - This allows up to 4 logo files to be defined.
  - These can be URLs to an external Website or images that are embedded directly in the field via the button to the right of each field.
    - **NOTE:** There is a 64KB limit for embedded URLs (data URIs). Also, it is more efficient to link to an external URL where possible.

- As per the Theme Colours, these can be used via <%Theme%> tags.

- **Login Form Theme**
  - You can specify one of the Logos defined in the Theme Logo URLs section to appear at the top of the Login form in place of Portal Name.
    - You can optionally specify the height at which to fix the Logo size.
      - If unspecified, this will be 80px (the default).
    - You can also opt to centre this logo.
    - **NOTE:** Logos are scaled to 80px high when displayed on the Login form.
  - A Background Image for the Login form can be entered as either the URL of an externally hosted image or as an Image-type Resource defined within the Portal.
    - NOTE: If using a Portal Resource, it must be configured to "Allow direct download via a URL" and not require Authenticated access.

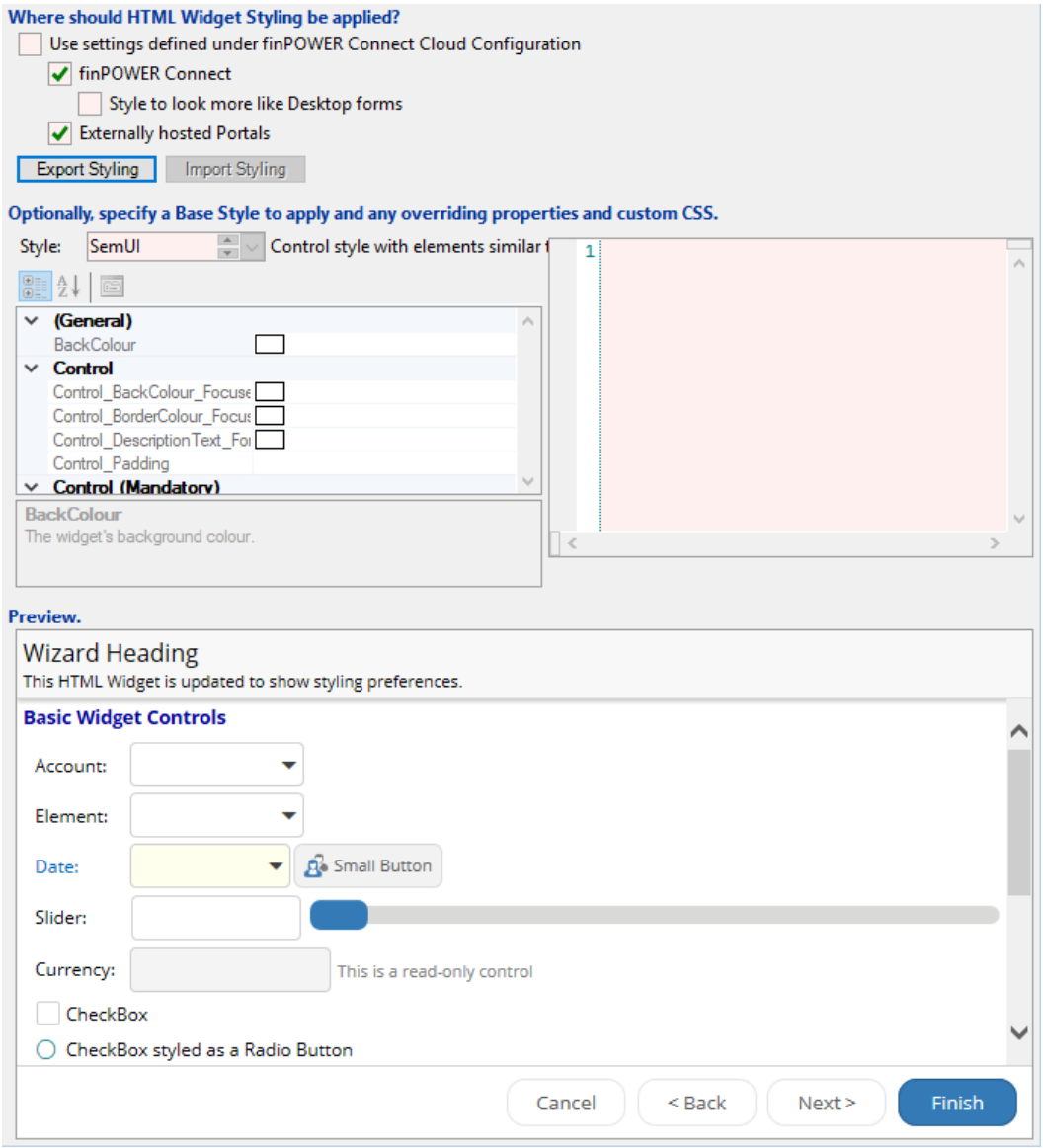
**NOTE:** By default, this page shows a generic preview of the specified Theme information.

However, adding a special Resource named "portalPreview" HTML resource allows a custom preview of the Portal to be defined as shown in the screen-shot above.

# HTML Widgets

Allows theming to be defined for any HTML Widgets hosted in this Portal and for any HTML Widget Controls (created via the `widget.UI` or `page.UI` objects) used in Portal pages.

This is almost identical to the corresponding page on the finPOWER Connect Cloud Configuration form.



## Master Page

Defines a template for the Portal.

This would typically be an HTML document with placeholders representing where the content of other pages should appear, e.g., the Master Page would contain headers, footers, menus etc and a section where the individual page content should be shown.

**NOTE:** Although not necessary, use of a Master Page is recommended since it allows centralisation of standard site layout such as Headers, Menus and Footer areas.

## Script Code

The Master Page Script code.

This has the same function signature as HTML Widget Scripts and Portal Page Scripts.

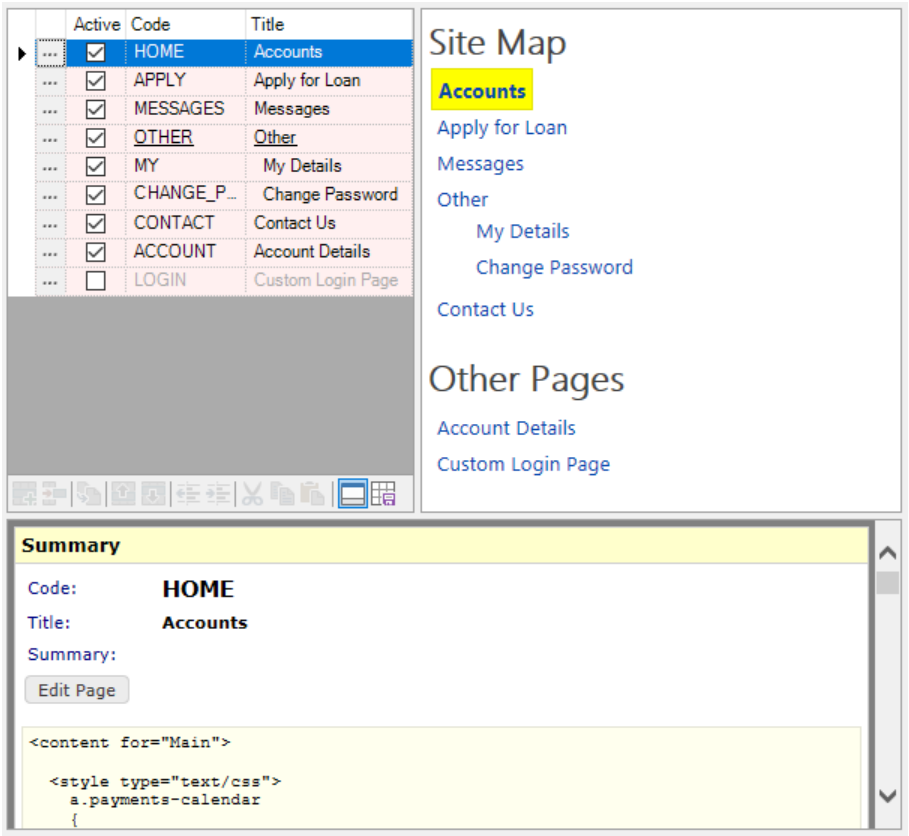
**NOTE:** If you have functions that you wish to call client-side from more than one page then they can be added to the Master Page Script Code and the `portal` rather than `page` object used to call them.

Alternatively, you could create a "Script Function Library" type Resource which means then functions would be callable client-side from any page and would also be available server-side.

# Pages

Defines Portal Pages.

A Portal must contain at least one Page to be useful.




The order of pages in the grid can be used to determine a site map which can be used to automatically generate a menu.

**NOTE:** On the IDE page, Pages are shown in alphabetical order to make them easier to locate.

## Resources
Defines a list of Resources (such as images) used by the Portal.

## Constants
Defines a list of Constants that can be used by the Portal Script, Portal Page Scripts or inserted into HTML and CSS via `<%Constant%>` tags.

## State
Allows Portal State to be viewed and cleared.

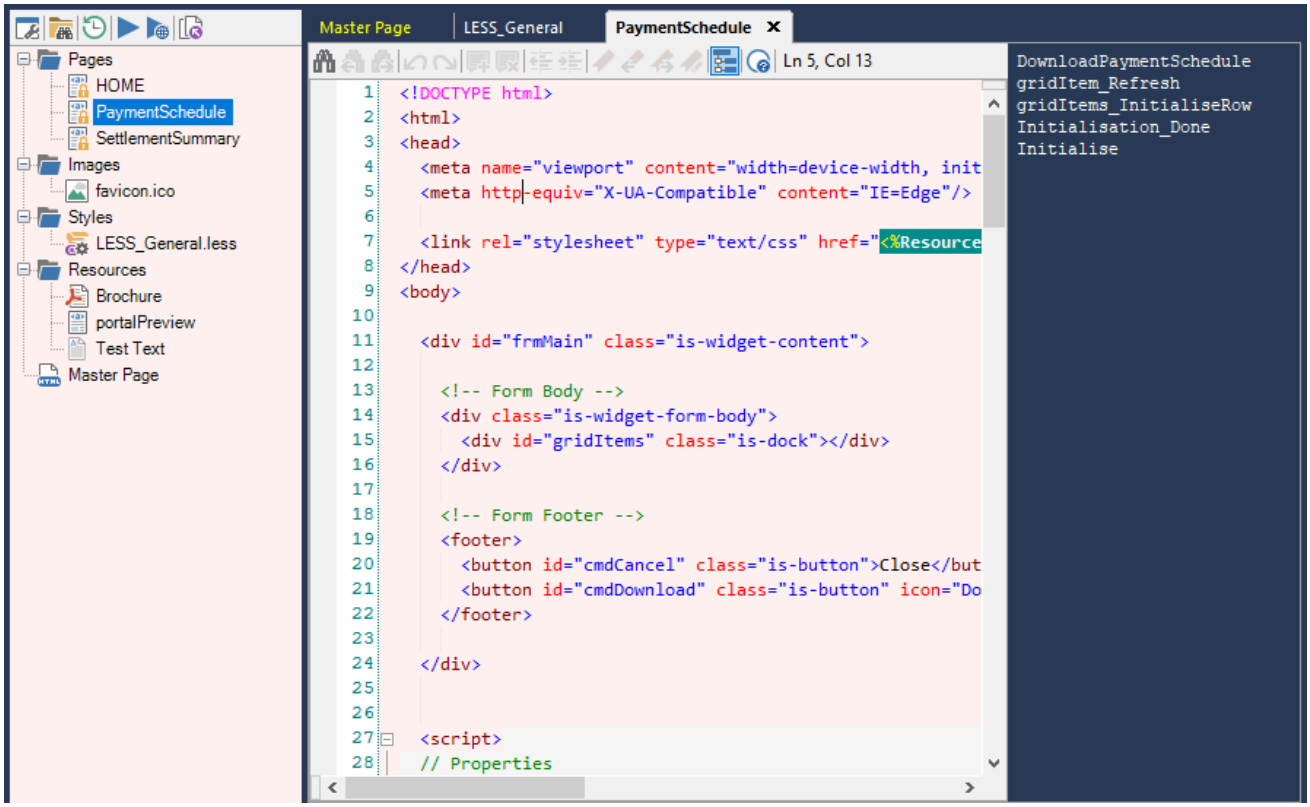
## Test
Allows the Portal to be tested within finPOWER Connect or in an External Web Browser.

## Publish
Allows the Portal to be published to a remote database at the click of a button.

**WARING:** Publishing a Portal copies only the Portal record to the remote database. It does not include any supporting Scripts, Documents etc and so should be used with caution.

# IDE

Allows the Portal's Scripts, Pages and Resources to be edited in an IDE-style environment where multiple tabs can be opened at once.




Open tabs (and their positions) are preserved on a per Portal, per User basis.

Tabs can be moved about in the Tab well by dragging them.

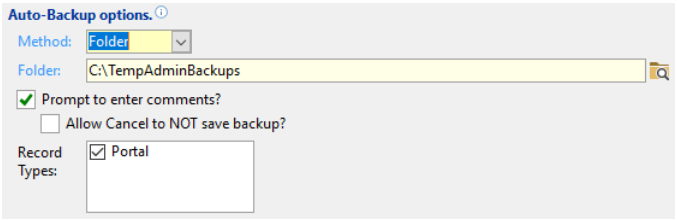
By default, double-clicking a Page will open its HTML view. Right-clicking the Page in the explorer allows you to select between HTML and Script Code.

## History

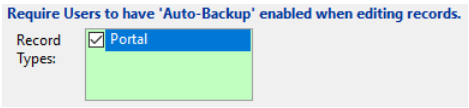
A Portal history can be maintained on a per-User basis.

**NOTE:** This is NOT source control, it is simply a way of keeping backups so that developers can record progress and, if necessary, view older copies of code.

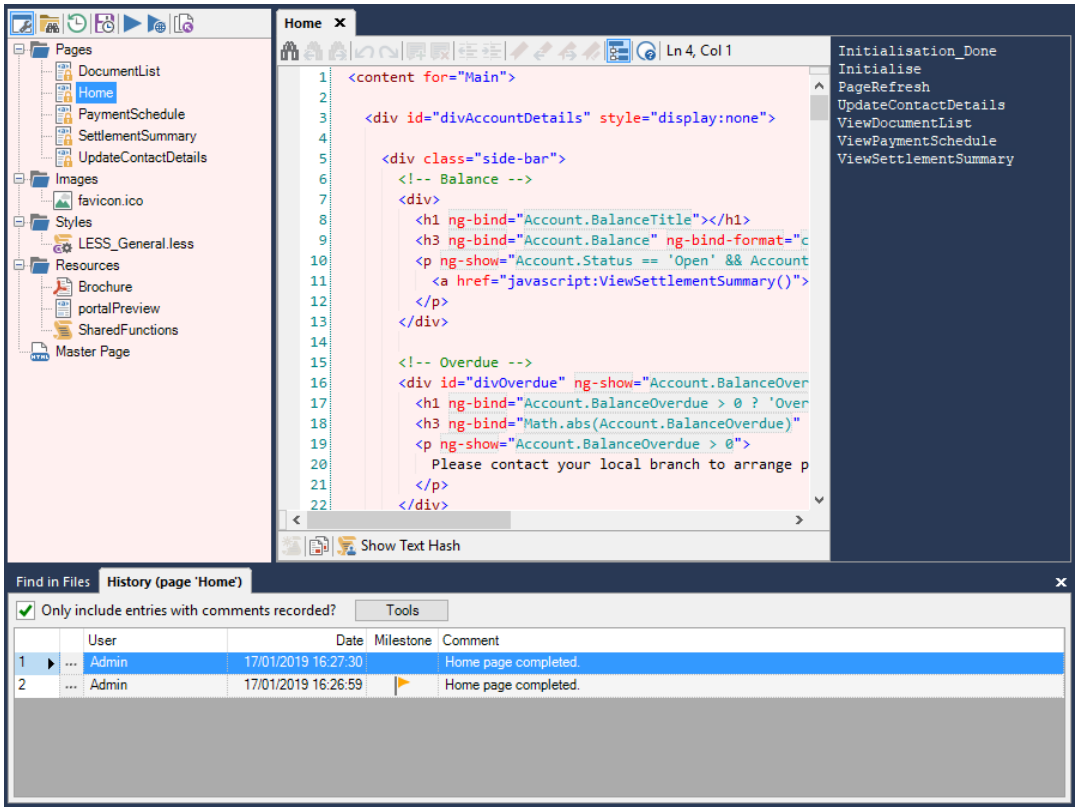
Under User Prefenreces, Developer, General, you can opt to create a backup copy of a Portal prior to each save:




You can also enforce that, before editing Portals, a User has their auto-backup settings turned on under Global Settings, Developer, General:



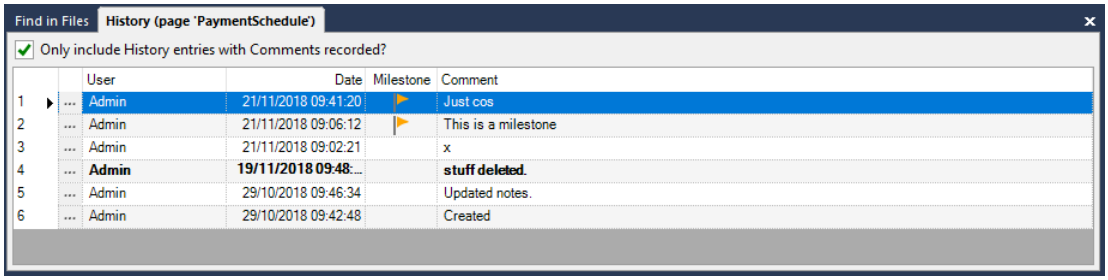

The "History" button at the top of the IDE page shows a list of these backups:



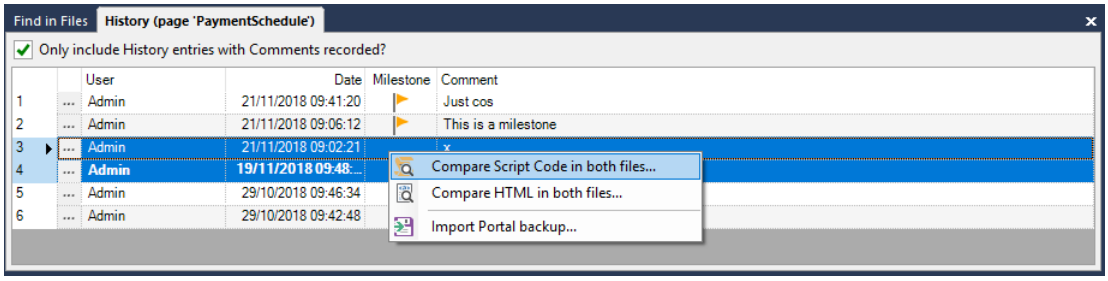

**NOTE:** By default, only entries against which comments are recorded will be shown.

This improves performance and also means that the grid only shows the more important saves.

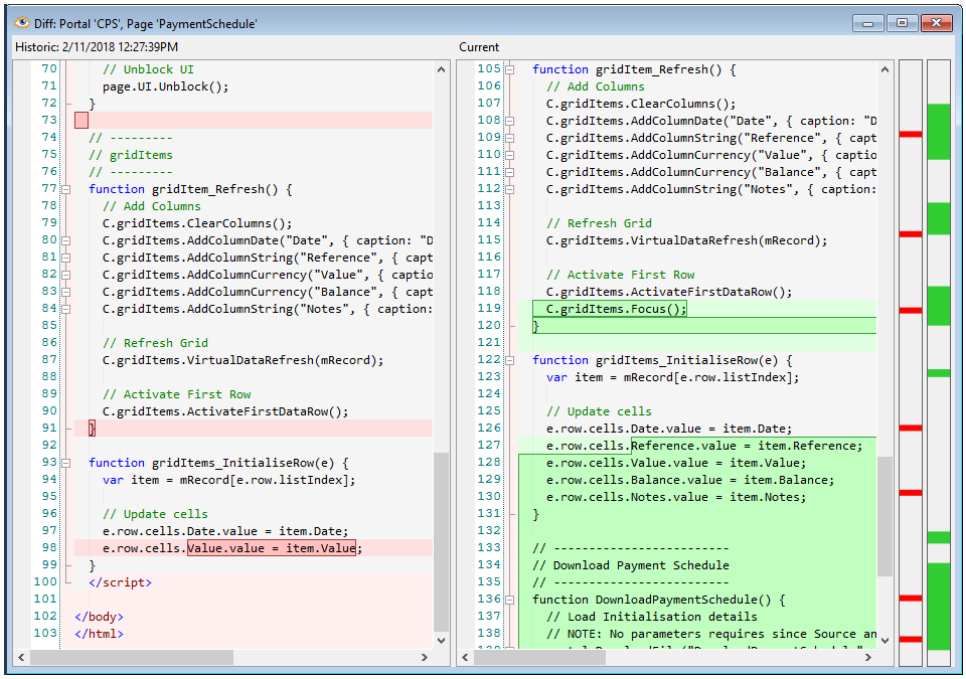
Selecting a Page or Resource in the explorer highlights entries in the History where that Page or Resource was changed:




You can then right-click and item in this grid to compare it against the current version or, select two rows in the History grid and compare them:



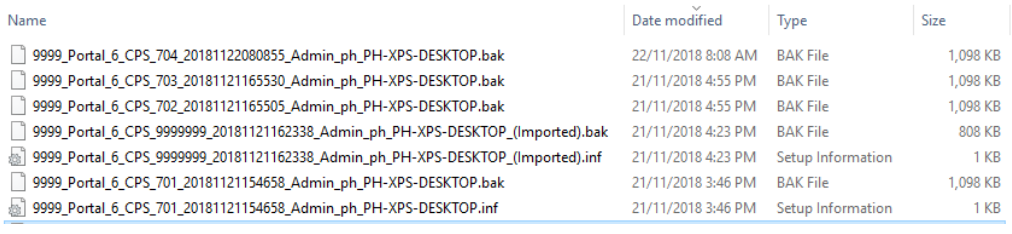

This will display a "Compare" form:




Right-clicking an entry in the History grid also gives you the option of restoring that version of the Portal as a new, unsaved record, i.e., the current Portal record WILL NOT BE TOUCHED IN ANY WAY.

From here, you can copy and paste items or text between the backup and the current Portal.

Portal History records are stored in the folder configured under User Preferences, Developer, General and can be viewed from Window Explorer:



### Manually Adding History Entries

Entries (XML files) can be manually added to this history using the "Tools" button to display the full Portal Auto-Backup History form and clicking the "Add" button.

This is useful, for example if, as a developer, you receive a file that someone else (e.g., a Client or Dealer) has modified and you want to keep track of it in your history.

Any files added in this way have a special '_(Imported)" suffix, as shown in the screenshow above.

**NOTE:** Of course, you would still need to update your own Portal record to reflect these changes.

### Purging History Entries

Over time, you may accumulate hundreds of history files so it is recommended you clean up this folder (i.e., delete older files) every so often; not just for the sake of storage but having a large history list will slow down the building of the History grid, particularly when a particular Page or Resource is selected in the IDE explorer.

Each file it a complete backup of the Portal so deleting files will not corrupt the backup in any way.

The "Purge" button on the Portal Auto-Backup History form allows you to quickly remove all non-commented files that are older than a given number of days. By default, this is 32 days but can be changed under Global Settings, Developer, General:

# Authentication and Security

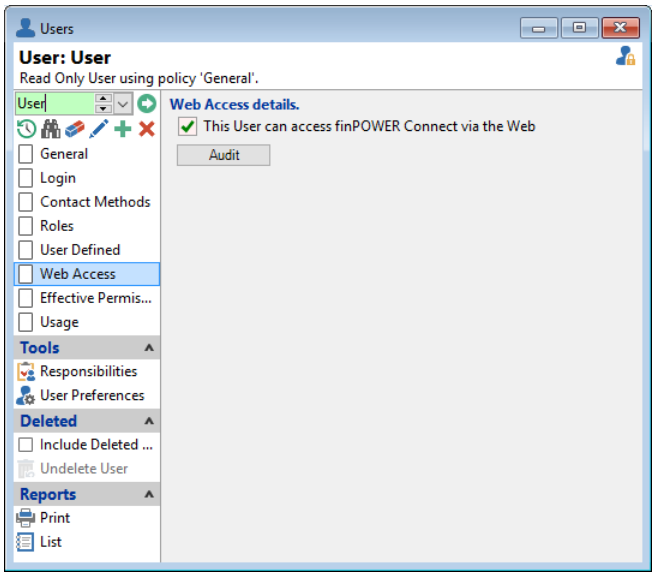
## Authentication Method

The Authentication page of the Portals form allows an Authentication Method to be specified.

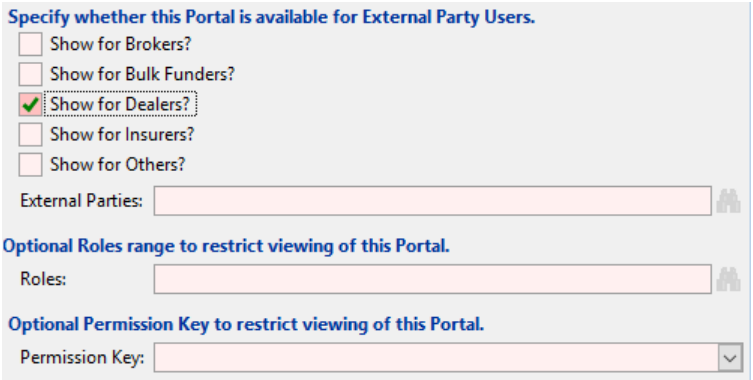
This determines which type of user can access the Portal and, if necessary, a sign-in form will be presented for the user to enter their credentials.

### User

- A finPOWER Connect User.
- When run from within finPOWER Connect, the current User is assumed and no sign-in page is used.
- When running from Web Services, the User must have "Web Access" enabled:



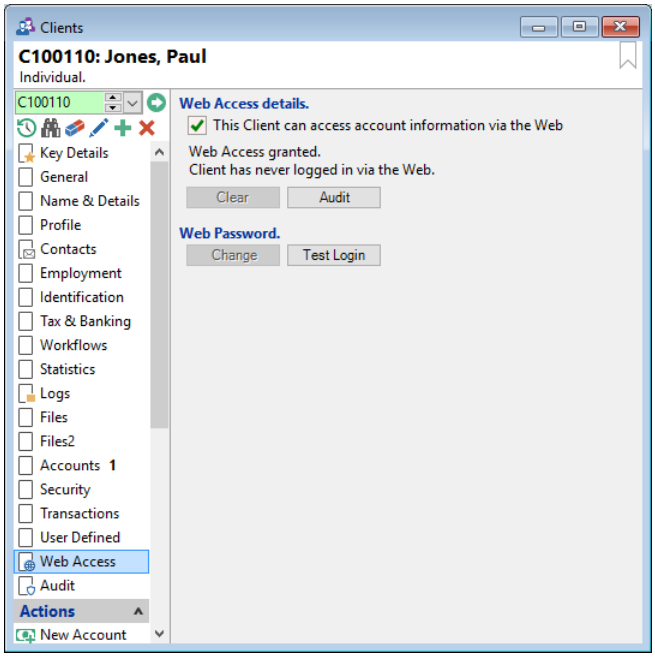

- Access to the Portal can be restricted to a certain type of External User or based on a range of External Parties or Roles or, on a Permission Key. This is done via the "Authentication" page of the Portals form:



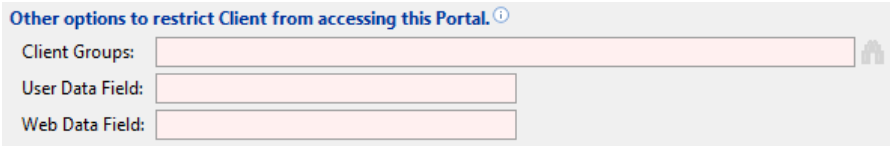

**NOTE:** By default, External Users cannot access any User-based Portals.

## Client

- A finPOWER Connect Client.
- The Client must have "Web Access" enabled:




- Access to the Portal can be restricted, e.g., to Clients within a range of Client Groups. This is done via the "Authentication" page of the Portals form:




## Unauthenticated

- No authentication is required to use this Portal.

# Session Timeout and Secret Key

## Session Timeout minutes

The Session Timeout defines how many minutes of inactivity before a Client or User is locked out of their Portal Session.

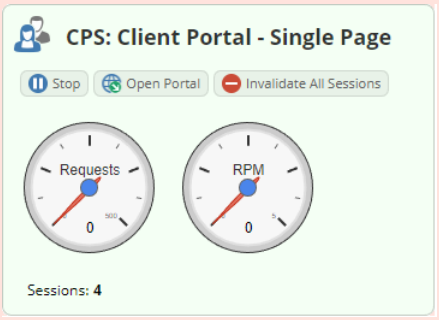
When talking about "Inactivity" with regard to Web applications, this typically means the time since the last request to the Web server was made. In the case of Portals, this can be interpreted as meaning the last time the Portal or Portal Page Script was called.

## Secret Key

The Secret Key is a random string of characters used when encrypting a Portal Session.

**IMPORTANT:** Changing the Secret Key will invalidate all currently signed in Portal User sessions and force the Client or User to sign in again.

The Portals page of the Web Administration facility has an "Invalidate All Sessions" button for each Portal that changes the Secret Key:



# Additional Authentication Checks

Once a User or Client has successfully entered their credentials, you may wish to perform additional checks such as restricting a Portal only to Clients that belong to a specific Client Group.

## Portal

You can perform additional authentication checks to prevent Users or Clients from accessing the entire Portal.

These checks are scripted in the Portal's Script Code page using a special `eventId` of "_Authenticate", e.g.:

```
' Objects
Private mPortalHandler As finPortalHandlerBL

Public Function Main(eventId As String,
                     parametersJson As String,
                     startUpParametersJson As String,
                     hostingContext As isefinHtmlWidgetHostingContext,
                     requestInfo As finScriptRequestInfo,
                     ByRef returnValue As String) As Boolean

  Dim Client As finClient

  ' Assume Success
  Main = True

  ' Initialise
  mPortalHandler = DirectCast(ScriptInfo.Properties.GetObject("PortalHandler"),
finPortalHandlerBL)

  ' Handle Events
  Select Case eventId
    Case ""
      ' Main event, i.e., return initial Master Page content
      returnValue = ScriptInfo.TemplateText

    Case "_Authenticate"
      ' Additional Authentication
      If mPortalHandler.Client.ClientId = "PAUL" Then
        Main = False
        finBL.Error.ErrorBegin("Client 'PAUL' is not allowed to use this Portal.")
      End If

    Case Else
      If eventId.StartsWith("_") Then
        ' Ignore system events
      Else
        Main = False
        finBL.Error.ErrorBeginFormat("Unhandled event '{0}'.", eventId)
      End If
  End Select

End Function
```

**NOTE:** All system events such as "_Authenticate" start with an underscore hence the `Case Else` code ignores these if they are unhandled to prevent any future issues if more system events are added.

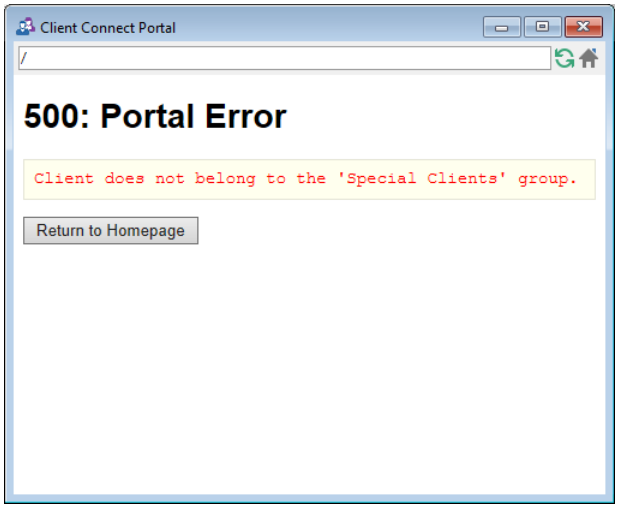
## Page

If a Page requires special Authentication then the main event that fetches the Page's HTML can be used in the Page's Script Code, e.g.:

```
' Objects
Private mPortalHandler As finPortalHandlerBL

Public Function Main(eventId As String,
```

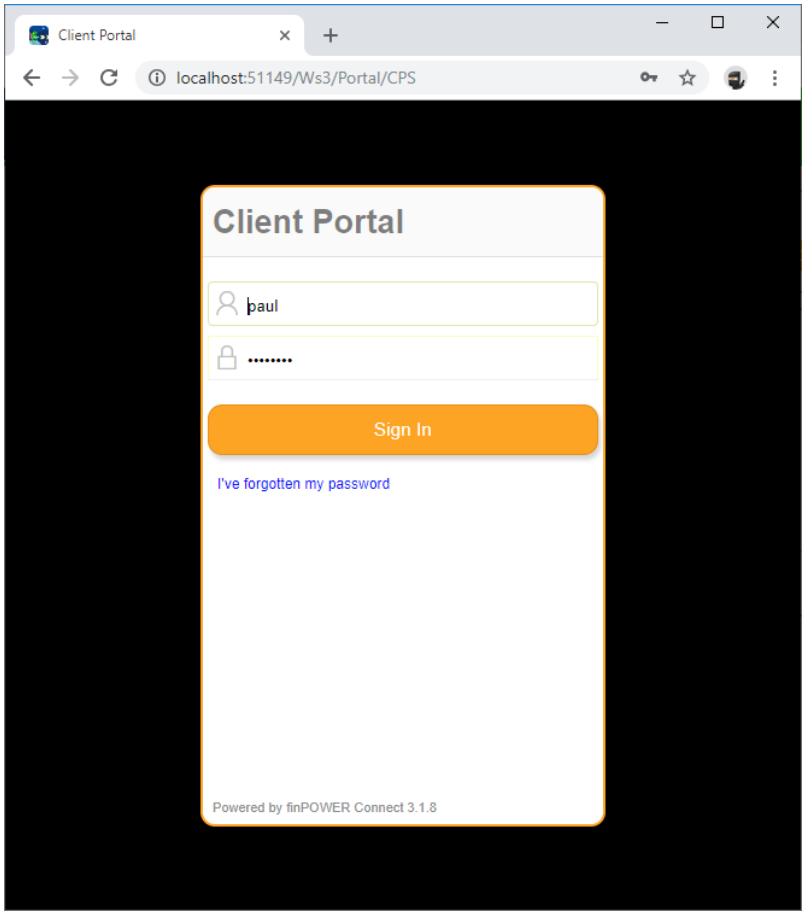
```
                    parametersJson As String,
                    startUpParametersJson As String,
                    hostingContext As isefinHtmlWidgetHostingContext,
                    requestInfo As finScriptRequestInfo,
                    ByRef returnValue As String) As Boolean

  ' Assume Success
  Main = True

  ' Initialise
  mPortalHandler = DirectCast(ScriptInfo.Properties.GetObject("PortalHandler"),
finPortalHandlerBL)

  ' Handle Events
  Select Case eventId
    Case ""
      ' Additional Authentication
      If mPortalHandler.Client.ClientGroupId <> "SC" Then
        Main = False
        finBL.Error.ErrorBegin("Client does not belong to the 'Special Clients' group.")
      End If

      ' Main event, i.e., return initial HTML content (excluding html and body tags)
      If Main Then returnValue = ScriptInfo.TemplateText

    Case Else
      Main = False
      finBL.Error.ErrorBeginFormat("Unhandled event '{0}'.", eventId)
  End Select

End Function
```

This will result in an unauthorised Client or User seeing the following:



# Custom Login Form

By default, User or Client Portals use a built-in Login (sign-in) form (this has a special URL of "@LOGIN"), e.g.:

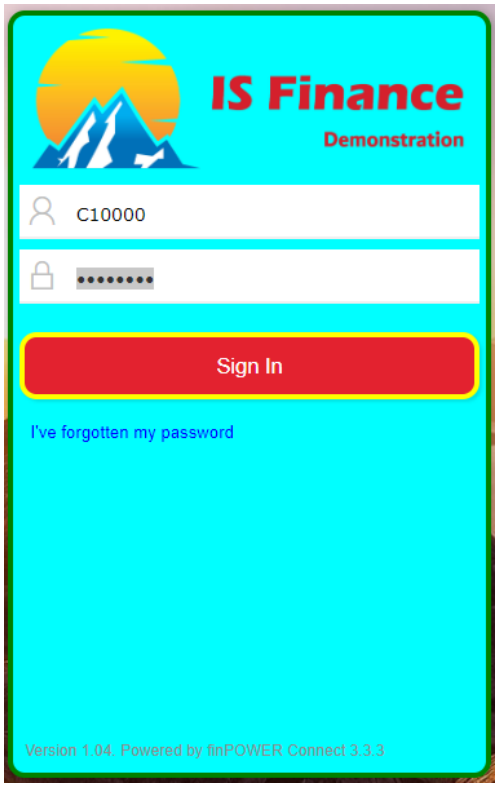



## Simple Style Changes

The "HTML Widgets" page of the Portals form allows custom CSS to be entered.

Using this, simple changes can be made to the page styling, e.g., the "Sign In" button:

```
.is-builtin-page-login {
  .is-portal-form {
    background-color:cyan !important;
    border:4px solid green !important;
  }

  #cmdLogin {
   border:4px solid yellow !important;
  }
}
```




**WARNING:** Be very careful when making CSS changes to avoid accidentally styling other parts of the Portal unintentionally.

## Custom Page

However, by defining a page with a special code of "LOGIN", a custom sign-in page can be defined.

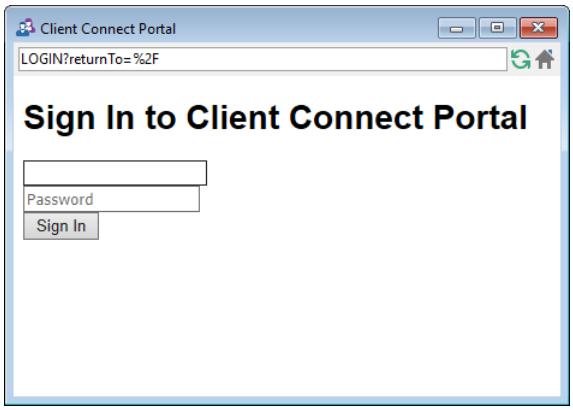
**WARNING:** This page will be ignored if it is not "Active" or does not "Allow Unauthenticated access".

The following example shows HTML for a custom Login form (no Script Code is required):

```
<h1>Sign In to <%PortalName%><h1>

<div id="divWarning" style="display:none"></div>


<form onsubmit="return SignIn()">
  <input id="id" placeholder="Client Id" maxlength="50" autocorrect="off" autofocus/>
  <br/>
  <input id="password" type="password" placeholder="Password" maxlength="50" autocorrect="off"/>
  <br/>
  <button id="cmdLogin" type="submit">Sign In</button
</form>


<script>
function SignIn() {
  // Validate
  var id = $("#id").val();
  var password = $("#password").val();
  if(!id) { $("#id").focus(); return false; };
  if(!password) { $("#password").focus(); return false; };

  // Authenticate
  portal.AuthenticateClient(id, password, SignIn_Fail);
```

```
  // Prevent Form from submitting
  return false;
}
function SignIn_Fail(data) {
  // Show Error
  $("#divWarning").html(page.UI.HtmlEncode(data));
  $("#divWarning").show();
  $("#id").focus();
}
</script>
```

This presents a simple, custom Login form:




**NOTE:** This sample does not include functionality to reset the password (as provided by the "I've forgotten my password" link on the built-in page.

As over version 3.03.04, pasting the template HTML for a Page with a code of "LOGIN" will generate a fully functional page based on the built-in form.

# Custom Error Pages

The following error pages can be replaced with custom pages:

- `401` (Unauthenticated)
- `404` (Not Found)

The Code dropdown on the Portal Page wizard contains special codes of ERROR_401 and ERROR_404.

Giving a page one of these codes will replace the built-in error page.

**WARNING:** This page will be ignored if it is not "Active" or does not "Allow Unauthenticated access".

## 401 (Unauthorised)

This page is displayed when the user is not authenticated or their session has expired.

In the example below, unremarking the Auto Sign-In JavaScript code forces the page to return to the sign-in page automatically.

```
<html>
<body>

  <img src="<%ResourceDataUri:LOGO%>"/>
  <h1>Not authorised.</h1>

  <script>
   var returnToUrl = "<%Literal:ReturnToUrl%>";

   // Auto Sign-In
   //portal.SignIn(returnToUrl);
  </script>

  <a href="javascript:portal.SignIn(returnToUrl)">Sign in and return to this page.</a>

</body>
</html>
```

**NOTE:** A special literal named **`ReturnToUrl`** is provided to this page.

## 404 (Not Found)

This page is displayed when the user attempts to navigate to a page that does not exist.

The example below shows a simply custom 404 page:

```
<html>
<body>

  <img src="<%ResourceDataUri:LOGO%>"/>
  <h1>Oops, this page could not be found!</h1>

</body>
</html>
```

# Navigation and Hyperlinks

Although Portals, at least when hosted outside of finPOWER Connect, are Websites, it is important to understand how to navigate between Portal pages.

The safest way is to use the `portal.Navigate()` method although, for convenience, anchor tag "href" attributes are handled automatically.

**WARNING:** Never use `window.location.href` to attempt to navigate between Portal pages.

## portal.Navigate

The portal JavaScript object has a `Navigate` method which is used to move between Portal pages or to reload the current page with URL parameters, e.g.:

```
portal.Navigate("Home");

portal.Navigate("Account?id=L10000");
```

This can also be called from hyperlinks, e.g.:

```
<a href="javascript:portal.Navigate('Home')">Home</a>
```

## portal.Navigate2

As of version 3.03.02 of finPOWER Connect, a `Navigate2` method is available.

This differs from the `Navigate` method in that you do not have to URL encode any parameters. Instead, a JavaScript object containing parameter values can be passed to the method, e.g.:

```
portal.Navigate2("Apply", { term: "2w", amount: 50 });
```

## portal.Open

A different Portal can be opened, e.g., when moving between a public Portal to a Client Portal, e.g.:

```
portal.Open("CP");

portal.Open("CP", true);
```

**NOTE:** The second parameter indicates whether to open the Portal in a new browser window (or tab).

## Anchor Tags "href" Attribute

Any anchor tags with an href attribute that starts with either "/" or "portal://" will automatically be directed to the respective Portal page, e.g.:

```
<a href="/Test">Test</a>

<a href="portal://Test?id=123">Test with Parameter</a>
```

## portal.OpenPageInModal

If you are creating a Single Page Application (SPA) or just want to show another page in a Modal form then the portal JavaScript object's `OpenPageInModal` method allows this, e.g.:

```
portal.OpenPageInModal("Settlment",
                       { accountId: "L10000" });

portal.OpenPageInModal("Settlement",
                       { accountId: "L10000" },
                       { title: "Settlement Summary" });
```

**NOTE:** The second parameter is a JavaScript object containing parameters to pass to the page (instead of using the URL).

The third parameter is the options to pass to the widget.UI.Modal form.

## portal.ShowPasswordChange

This is a special method that shows either the built-in or a custom Password Change page and allows a return URL to be specified to return to once the Password has been changed, e.g.:

```
portal.ShowPasswordChange("/Home");

portal.ShowPasswordChange("/", true,
                       { title: "Update Your Password" });
```

**NOTE:** The second parameter of this method allows the page to be shown in a Modal form and the third parameter is the options to pass to the `widget.UI.Modal` form.

# Master Page

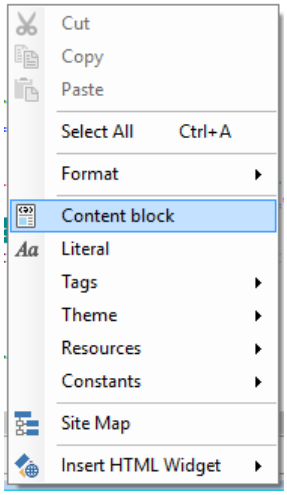
A Master Page is used to define an overall template for the Portal.

Via "Content blocks", a Portal Page can inject its content into one of more places on the Master Page.

**NOTE:** This is a similar concept to ASP.NET Master Pages, but with a slightly different syntax.

Pages can opt to not use the Master Page, e.g., a custom Login page may not suit the Master Page template.

The following sections cover the various tags that can be used in Master Pages or inserted via right-clicking in the Master Page field:



## Content blocks

Special <%Content%> tags are used in the Master Page to define regions that can be populated by a Portal Page, e.g.:

```
<!DOCTYPE html>
<html>
<head>
</head>
<body>

  <!-- Header Begin -->
  <h1><%PortalName%></h1>
  <!-- Header End -->


  <!-- Content Begin -->
  <h1><%PageTitle%></h1>

  <%Content:Main%>
  <!-- Content End -->



  <!-- Footer Begin -->
  <footer>
    Standard text to appear on each page.
  </footer>
  <!-- Footer End -->

</body>
</html>
```

A Master Page can contain one or more content blocks.

In the above example, a single content block with a name of "Main" is defined.

# Literals

Literals are a special type of Tag that can exist in the Master Page or a Portal Page.

Once the Page has been fetched, all Literals will be replaced by values that have been set in the Script (this could be the Master Page Script or a Portal Page Script), e.g.:

```
<!DOCTYPE html>
<html>
<head>
</head>
<body>

  Hello, your name is:

  <%Literal:PersonsName%>

</body>
</html>
```

Literals are populated via a special `ScriptInfo.Properties` Key/ Value List. They would typically be populated when fetching either the Master Page or the Portal Page's content, i.e., in the "" event, e.g.:

```
' Handle Events
Select Case eventId
  Case ""
    ' Main event, i.e., return initial Master Page content
    returnValue = ScriptInfo.TemplateText

    ' Set Literals
    With ScriptInfo.Properties
      .SetString("PersonsName", "Paul")
    End With

End Select
```

## Constants

Inserts a Constant into the page.

```
<%Constant:MyColour%>
```

Constants are defined on the "Constants" page of the Portals form.

You can also specify a default value in parentheses, e.g.:

```
<%Constant:MyColour(blue)%>
```

This is particularly useful when using tags within JavaScript where having a missing Constant (i.e., a name that does not exist in the Constants grid) may produce unpredictable results, e.g.:

```
var mAccounts_ShowAccountActions = <%Constant:Accounts_ShowAccountActions(false)%>;
```

**NOTE:** From version 3.03.03, if an unknown Constant name is specified then, instead of just a blank value being inserted (which would break JavaScript), the value `null` is inserted.

This may look odd if the tag is being used simply to insert text into HTML or CSS but has the benefit of not preventing JavaScript code from compiling (or being invalid).

## User Data

Inserts a User Data value into the page.

```
<%UserData:MyValue%>
```

Constants are defined in the `finPortal.UserData` property and can only be created and updated via Script code.

Typically, User Data can be used as an alternative to Constants by a Configuration utility.

You can also specify a default value in parentheses, e.g.:

```
<%UserData:MyValue('hello world')%>
```

This is particularly useful when using tags within JavaScript where having a missing User Data item (i.e., a name that does not exist in the Key/ Value list when viewing User Data from the Audit page of the Portals form) may produce unpredictable results, e.g.:

```
var mAccounts_ShowAccountActions = <%UserData:Accounts_ShowAccountActions(false)%>;
```

**NOTE:** From version 3.03.03, if an unknown User Data name is specified then, instead of just a blank value being inserted (which would break JavaScript), the value **`null`** is inserted.

This may look odd if the tag is being used simply to insert text into HTML or CSS but has the benefit of not preventing JavaScript code from compiling (or being invalid).

# SiteMap

This is a special tag that inserts a hierarchical list (HTML unordered list) into the document.

```
<%SiteMap%>
```

This creates HTML in the following format:

```
<ul menu-id='root' menu-level='0'>
  <li class='current'><div><a href="javascript:portal.Navigate('HOME')">Accounts</a></div></li>
  <li><div><a href="javascript:portal.Navigate('APPLY')">Apply for Loan</a></div></li>
  <li><div><a href="javascript:portal.Navigate('MESSAGES')">Messages</a></div></li>
  <li class='has-sub-menu'><div><a>Other</a></div>
    <ul menu-id='MY' menu-level='1'>
      <li><div><a href="javascript:portal.Navigate('MY')">My Details</a></div></li>
      <li><div><a href="javascript:portal.Navigate('PWD')">Change Password</a></div></li>
    </ul>
  </li>
  <li><div><a href="javascript:portal.Navigate('CONTACT')">Contact Us</a></div></li>
</ul>
```

This list can then have CSS applied to form a menu, e.g.:

```
TODO:
```

# Resources

These are special tags that allow Resources to be used in the page.

Generally, you will be using a URL to a resource, e.g.:

```
<link rel="stylesheet" type="text/css" href="<%ResourceUrl:CSS_Menu%>"/>

<img src="<%ResourceUrl:Logo%>"/>
```

However, there may be occasions where you want to use a Data URI instead. A Data URI embeds the entire resource in the Web page rather than referencing an external file.

A common use for using a Data URI would be to embed a small icon, such as a warning icon, within a CSS resource, e.g.:

```
.warning
{
  border: 2px solid red;
  background-image: url(<%ResourceDataUri:WarningIcon%>);
}
```

## Replaceable [THEME] Tag

A "Theme" can be defined on the Theme page of the Portals form.

This can then be substituted dynamically into Resource tags, e.g.:

```
<%Resource:[THEME].theme%>
```

If the "Theme" on the Theme page of the Portals form was set to "TEST", this would resolve to the following:

```
<%Resource:TEST.theme%>
```

## Resource Bundles

Each CSS, Less or JavaScript Resource can be given a "Bundle Name". A Resource Bundle tag can then be used to include all of these resources into a Master Page or Page instead of listing each of the Resources.

This generates the appropriate <link> and <script> HTML tags, e.g.:

```
<head>

<%ResourceBundle:Common %>

</head>
```

# Tags

Tags are replaced by information relating to the Portal, Page or Client/ User, e.g.:

```
<%ClientFirstName%>
```

Tags can be typed in manually or inserted by right-clicking the Master Page or Portal Page HTML. The following Tags are available to both the Master Page and Portal Pages:

| Tag | Description |
|---|---|
| PortalId | The Portal Id |
| PortalName | The Portal Name |
| PageId | The Portal Page Id |
| PageTag1 | Custom Tag 1 for this Portal Page. |
| PageTag2 | Custom Tag 2 for this Portal Page. |
| PageTag3 | Custom Tag 3 for this Portal Page. |
| PageTitle | The Portal Page Title |
| PageSummary | The Portal Page Summary |
| ParentPageId | If used in a Partial Page, this will return the Id of the Page hosting the Partial Page. Otherwise it simply returns the current Portal Page Id. |
| ClientId | The Client Id or blank if there is no authenticated Client |
| ClientName | The Client Name or blank if there is no authenticated Client |
| ClientFirstName | The Client First Name or blank if there is no authenticated Client |
| ClientLastName | The Client Last Name or blank if there is no authenticated Client |
| ClientPreferredName | The Client Preferred Name (or their First Name if unspecified) or blank if there is no authenticated Client |
| UserId | The User Id or blank if there is no authenticated User |
| UserName | The User Name or blank if there is no authenticated User |
| ExternalPartyId | The External Party Id or blank if there is no authenticated User or the authenticated User is not an external User |
| ExternalPartyName | The External Party Name or blank if there is no authenticated User or the authenticated User is not an external User |
| CurrentDate | The current database date. |
| CurrentDateTime | The current database date and time. |
| CurrentDateTimeZone | The current database time zone. |
| CurrentYear | The current database year. |

## Overriding Tags

Tag values can be overridden.

For example, you may have a page showing Account details where you want to include the Account Id as part of the Page Title.

When fetching the Page's initial HTML (the "" event), you can set overriding values for tags as follows:

```
Select Case eventId
  Case ""
    ' Main event, i.e., return initial HTML content
```

```
    returnValue = ScriptInfo.TemplateText

    ' Include Account Id with Page Title
    mPortalHandler.TagOverrides.Add("PageTitle", "Account " & mAccountId)
```

# Theme

These are special tags that allow the values defined on the "Theme" page of the Portals form to be inserted, e.g.:

```
<%Theme:MainBackground%>
```

These would typically be used in a CSS or LESS resource, e.g.:

```
.page-header
{
  border: 2px solid red;
  background-color: <%Theme:MainBackground%>;
  color: <%Theme:MainForeground%>;
  background-image: url(<%Theme:Logo1%>);
}
```

The following Theme Tags are supported:

| Theme Tag | Description |
|---|---|
| MainForeground | The Main Foreground colour. |
| MainBackground | The Main Background colour.<br>Typically, this would be used for the site's header area. |
| ContrastForeground | The Contrast Foreground colour. |
| ContrastBackground | The Contrast Background colour.<br>Typically, this would be used for elements such as tabs or headings (e.g., H1) tags. |
| Other1 | Other Colour 1. |
| Other2 | Other Colour 2. |
| Logo1 | The content of Logo 1 (a URL, or a data URI). |
| Logo2 | The content of Logo 2 (a URL, or a data URI) |
| Logo3 | The content of Logo 3 (a URL, or a data URI) |
| Logo4 | The content of Logo 4 (a URL, or a data URI) |
| Logo1Url | Logo 1 (if this is a data URI then it will NOT be inlined when accessed from Web Services and will be retrieved as a cacheable image) |
| Logo2Url | Logo 2 (if this is a data URI then it will NOT be inlined when accessed from Web Services and will be retrieved as a cacheable image) |
| Logo3Url | Logo 3 (if this is a data URI then it will NOT be inlined when accessed from Web Services and will be retrieved as a cacheable image) |
| Logo4Url | Logo 4 (if this is a data URI then it will NOT be inlined when accessed from Web Services and will be retrieved as a cacheable image) |

**NOTE:** Constants can also be used for theming in the same way as Theme tags.

# Widget

This is a special tag that inserts an HTML Widget into the document.

```
<%Widget:TEST:{ param1: "value1", param2: 123 }%>
```

The Widget tag is divided into three parts, each separated by a colon:

1. **Widget**
   The tag type identifier.
2. **Widget Id**
   The Id of the HTML Widget Script, e.g., "TEST".
3. **Startup Parameters**
   A JSON formatted String of Startup Parameters to be passed to the Widget.

**NOTE:** Behind the scenes, this simply inserts HTML and JavaScript blocks and uses the `portal.InsertWidget()` JavaScript method to add the HTML Widget.

## Styling

An HTML Widget is represented by an IFRAME element. When inserted into the page via a Widget tag, this IFRAME will be enclosed in a DIV element with the id of the Widget, e.g.:

```
<div id="divWidgetABC" widget="ABC"><iframe/></div>
```

The DIV tag contains a "widget" attribute which can be used as an alternative to the "id" tag in CSS to stye the content. The inner IFRAME which contains no style information and so will just appear as a small rectangular box containing the Widget. CSS can be used to style the Widget, e.g.:

```
#divWidgetABC > iframe
{
  border:none;
  height:280px;
  width:100%;
}
```

## Security

The HTML Widget will run in the same security context as the Portal, e.g., if the Portal is a "Client" portal then the HTML Widget's `requestInfo` parameter will reflect this, e.g.:

```
If requestInfo.AuthenticatedUserType = isefinwsAuthenticatedUserType.Client Then
  ' Authenticated Client
  ClientId = requestInfo.AuthenticatedClientId
End If
```

# Portal Pages

Portal Pages can use all of the various tags such as `<%Constant%>` and `<%Literal%>` in the same way as Master Pages. The only exception is `<%Content%>` tags which only apply to Master Pages.

## Content blocks

If the Portal Page is to use the Master Page as a template, it must define all of its content (HTML, Script and CSS if required) within one of more content blocks, e.g.:

```
<content for="Main">

  <div id="tabGeneral" class="tabs">
    <ul>
      <li tabId="Inbox">Inbox</li>
      <li tabId="Sent">Sent</li>
      <li tabId="Deleted">Deleted</li>
    </ul>

    <div id="pageInbox"><div id="gridInbox"></div></div>
    <div id="pageSent"><div id="gridSent"></div></div>
    <div id="pageDeleted"><div id="gridDeleted"></div></div>
  </div>

  <script>
    // Script code for this Portal page
  </script>

</content>
```

In the above example, all of the content between the content tags will be inserted into the Master Page at the **`<%Content:Main%>`** tag.

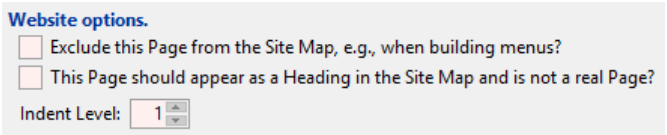
**NOTE:** If the HTML does not contain any `<content>` blocks then the Master Page will not be used; instead, the Portal Page should define its own `<html>`, `<head>` and `<body>` tags.

This allows Portal Pages to exist that are styled completely different from the rest of the Portal, e.g., a custom Login page.

## Site Map

Portal Pages can build up a Site Map.

The Website options section of the Portal Page wizard determines where (and if) the Page will appear in the Site Map, e.g.:



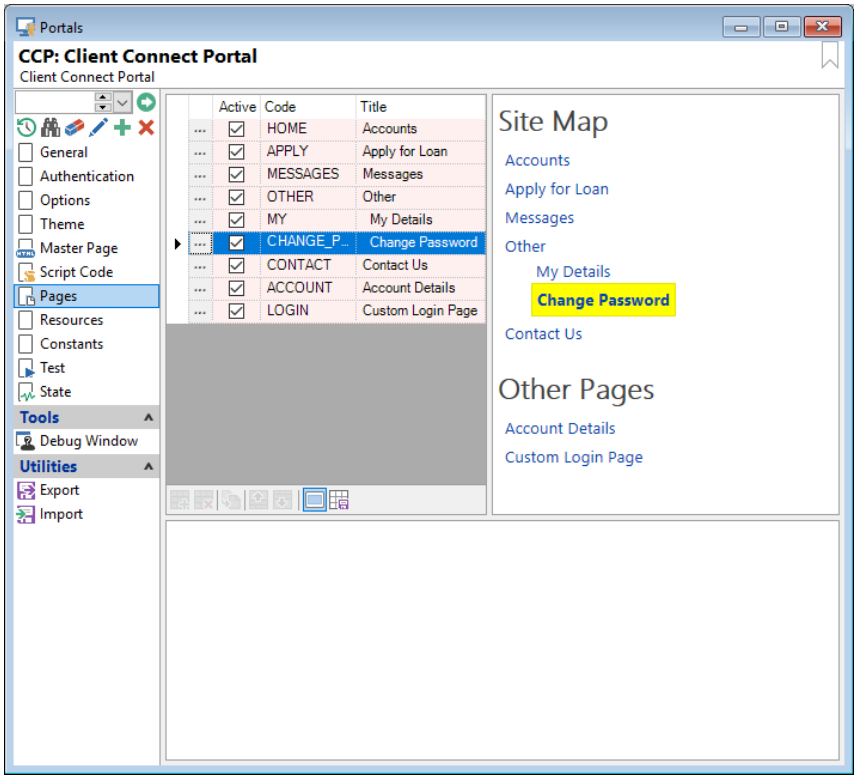

These options affect the Site Map as follows:

- Exclude this Page from the Site Map
  - This flags that the page should not appear in the Site Map, e.g., it can only be accessed from another page; not from the menu.

- See the "ACCOUNT" and "LOGIN" pages in the screenshot below.

- This Page should appear as a Heading
  - Useful for creating menus where the main menu item is not really a page, just a header for the items below it.
  - See the "OTHER" page in the screenshot below.
- Indent Level
  - This allows you to build a site hierarchy.
  - Indenting a page indicates that it is a child or sub-page of the page above.
  - See the "OTHER", "MY" and "CHANGE_PASSWORD" pages in the screenshot below.

The Portals form displays an outline of the Site Map, e.g.:

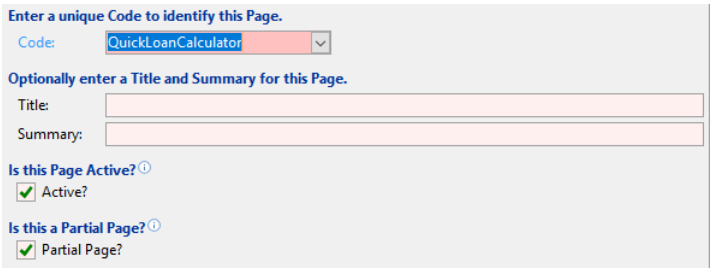


# Partial Pages

Partial Pages are designed to represent reusable blocks that can be included in other pages, e.g., a Loan Calculator or pages in a Single Page Application (SPA).

**NOTE:** Conceptually, Partial Pages are very similar to HTML Widgets. The main difference is that they are not hosted within an IFRAME and are included within the Portal record itself.

A Page is flagged as a Partial Page via the Portal Page wizard:




It can be inserted into other pages using the `<%PartialPage%>` tag, e.g.:

```
<%PartialPage:QuickLoanCalculator%>
```

## Partial Page Guidelines

**NOTE:** If the Partial Page contains no Script code or JavaScript and it simply represents a static piece of HTML, it would be served using an "HTML" type Resource instead.

### HTML Elements

Give all HTML element ids a prefix that is unlikely to clash with other page elements, e.g.:

```
<li>
  <label for="qlc_Term">What is your preferred repayment term?</label>
</li>
<li>
  <input id="qlc_Term" type="range"/>
</li>
<li>
  <div>
    <label id="qlc_TermMin" class="min"></label>
    <label id="qlc_TermMax" class="max"></label>
    <label id="qlc_TermSelected" class="selected"></label>
  </div>
</li>
```

### JavaScript

Use a jQuery initialisation block and nest ALL code within this so it does not clash with other code in the page, e.g.:

```
$(function () {
  // Controls
  var C = {
    rsTerm: null,
    lblTermMin: null,
    lblTermMax: null,
    lblTermSelected: null,
  }
```

```
  // --------
  // Controls
  // --------
  C.rsTerm = widget.UI.RangeSlider("qlc_Term", {
    minValue: 2,
    maxValue: 10,
    step: 1,
    width: "100%",
  });
  C.rsTerm.Value(C.rsTerm.MinValue());
  C.lblTermMin = $("#qlc_TermMin");
  C.lblTermMax = $("#qlc_TermMax");
  C.lblTermSelected = $("#qlc_TermSelected");

  // ---------
  // Calculate
  // ---------
  function CalculationInvalidate() {
  }
});
```

## Inline CSS and LESS

You can easily inline CSS or LESS-based CSS (which is automatically parsed by the Portal handler if you use a `type="text/less"` attribute) using HTML `<style>` tags, e.g.:

```
<style type="text/less">
.quick-loan-calculator {
  border:1px solid blue;
  display:inline-block;

  > ul {
    margin:0;
    list-style:none;
    padding:0;

    > li {
      margin:0;
      padding:0;

      /* Slider labels */
      > div {
        min-height:24px;
        position:relative;
        margin:-6px 0 8px 0;

        label {
          display:inline-block;
          font-size:13px;
          position:absolute;
          width:auto;
          left:unset;
          right:unset;
          white-space: nowrap;

          /* Min */
          &.min {
            left:0;
          }

          /* Max */
          &.max {
            right:0;
          }

          /* Selected */
          &.selected {
            border:1px solid rgba(0, 0, 0, 0.2);
            border-radius:8px;
            background-color:<%Theme:ContrastColourBackground%>;
```

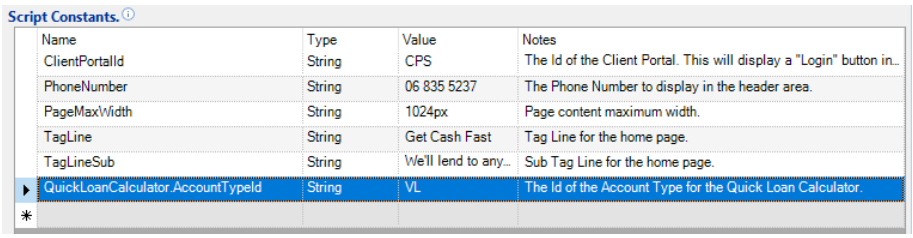
```
            color:<%Theme:ContrastColourForeground%>;
            padding:4px;
            top:-48px;
            right:unset;
          }
        }
      }
    }
  }
}
</style>
```

**NOTE:** The advantage of inlining styles is that its helps make the entire Partial Page self-contained without any reliance on other Resources.

The advantage of using LESS is that you can nest styles in a more readable fashion which is ideal for Partial Pages.

## Constants

If the Partial Page requires any configuration in the main Portal, it is recommended that you name all constant using a prefix and dot separator, e.g. "QuickLoanCalculator.AccountTypeId":




| Name | Type | Value | Notes |
|---|---|---|---|
| ClientPortalId | String | CPS | The Id of the Client Portal. This will display a "Login" button in... |
| PhoneNumber | String | 06 835 5237 | The Phone Number to display in the header area. |
| PageMaxWidth | String | 1024px | Page content maximum width. |
| TagLine | String | Get Cash Fast | Tag Line for the home page. |
| TagLineSub | String | We'll lend to any... | Sub Tag Line for the home page. |
| QuickLoanCalculator.AccountTypeId | String | VL | The Id of the Account Type for the Quick Loan Calculator. |

# Multi-Use Partial Pages and Avoiding Element ID Conflicts

If you follow the above guidelines, then your Partial Page should be fully self-contained.

However, it will only be self-contained if it is used only once in a page. If you want to use the Partial Page more than once, then you will need to make element ids unique. This is where the special <%PartialId%> tag can be used. It is given a unique code which can then be applied to ids, e.g.:

```
<li>
  <label for="<%PartialId%>_qlc_Term">What is your preferred repayment term?</label>
</li>
<li>
  <input id="<%PartialId%>_qlc_Term" type="range"/>
</li>
<li>
  <div>
    <label id="<%PartialId%>_qlc_TermMin" class="min"></label>
    <label id="<%PartialId%>_qlc_TermMax" class="max"></label>
    <label id="<%PartialId%>_qlc_TermSelected" class="selected"></label>
  </div>
</li>


<script>
$(function () {
  // Controls
  var C = {
    rsTerm: null,
    lblTermMin: null,
    lblTermMax: null,
    lblTermSelected: null,
```

```
  }

  // --------
  // Controls
  // --------
  var prefix = "<%PartialId%>";
  C.rsTerm = widget.UI.RangeSlider(prefix + "_qlc_Term", {
    minValue: 2,
    maxValue: 10,
    step: 1,
    width: "100%",
  });
  C.rsTerm.Value(C.rsTerm.MinValue());
  C.lblTermMin = $("#" + prefix + "_TermMin");
  C.lblTermMax = $("#" + prefix + "_TermMax");
  C.lblTermSelected = $("#" + prefix + "_TermSelected");
});
</script>
```

**NOTE:** The use of the `<%PartialId%>` tag does make the HTML and JavaScript slightly more difficult to read but does mean that there should never be any control or element naming conflicts on included pages.

## Script Callbacks

Since a Partial Page has access to both the `portal` and `page` JavaScript objects, it can communicate with Script Code, e.g., to fetch chart data.

However, calling the `page.GetString` method will call back to the main page that is hosting the Partial Page which is probably not desirable.

Using `portal.GetString` will call back to the Master Page's Script Code, i.e., the Script Code page on the Portal form. This may be desirable but means that the Partial Page's Script Code and its HTML Code are not self-contained in the Page which is not ideal.

A special syntax can be used in `portal.GetString` to ensure that the callback occurs to the Partial Page's Script code. The sample below assumes that you have a Partial Page with a code of Portfolio and therefore prefixes the `eventId` parameter with "Portfolio:", e.g.:

```
portal.GetString("Portfolio:GetData", {},
  function (data) {
    window.alert(data);
  },

  function (error) {
    window.alert("ERROR: " + error);
  });
```

Rather than hard-coding the Page Id, you can use a replaceable tag, e.g.:

```
portal.GetString("<%PageId%>:GetData", {},
```

# Special Page Codes

The Custom Login Form section details how to create a custom Login form by using a special Page Code of "LOGIN".

Below is a list of all special Page Codes and a brief description of how they work:

**ERROR_401**

Defines a custom 401, "Unauthenticated " page.

**ERROR_404**

Defines a custom 404, "Page Not Found" page.

**LANDING**

If a Page with this Code exists, is active and "Allows Unauthenticated access" then this will be the initial page that the Client/ User sees when navigating to the Portal.

Typically, this page would show a button to go to the Login form which is achieved by calling the `portal.SignIn()` method.

**LOGIN**

Defines a Custom Login Form.

**PASSWORD_CHANGE**

Defines Custom Password Change page that will be displayed instead of the built-in page when the `portal.ShowPasswordChange()` method is called.

**PORTAL_CONFIG**

Defines a special page which is in fact an HTML Widget definition rather than a true Portal page.

This allows a configuration utility to be defined as per the system-supplied Client Connect Portal sample.

**TERMS**

If a Page with this Code exists, is active and "Allows Unauthenticated access" then this will provide a link on the Login form. If the Page has as title, this will used as the link text, otherwise the caption "Terms & Conditions" will be used.

Clicking this link calls the `portal.ShowTerms()` method.

# Resources

Resources are items, either text or binary files, that can be used by the Portal.

Each resource must be given a unique code (or Resource Id).

## Special Resources

The "Code" dropdown contains a list of special Resource Ids that are used by the system. These are:

### favicon

- The Favourite Icon used by Web Browsers.
- This is an "Image (ICO)" type Resource.
- **NOTE:** Even though the Resource Type will be locked at "Image (ICO)", the actual image can be any format, e.g., PNG.

### PASSWORD_RESET_EMAIL

- Used to provide a custom email for Password Request Requests.
- This is an "HTML" type Resource.
- The following special tags are available:
  - `<%ResetLink%>`
    - An HTML anchor tag containing a link to the password reset page.
  - `<%ResetUrl%>`
    - A URL that can be used to provide a link to the password reset page.

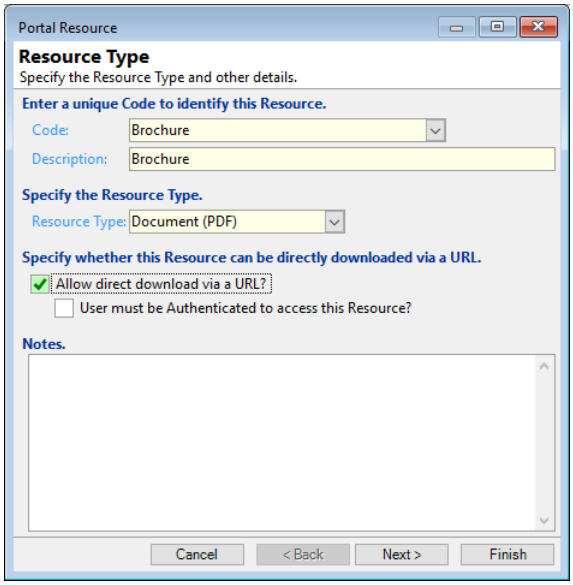
**NOTE:** Formatting HTML emails so they work across a variety of email clients is outside of the scope of this document.

### portalPreview

- A preview of this Portal to display on the 'Theme' page of the Portals form.
- This is an "HTML" type Resource.
- Any constants of theme tags contained in the HTML will be resolved however, there will not be a reference to the HTML Widgets JavaScript or CSS files.

## Securing Resources

The first page of the Portal Resource wizard allows a Resource to be secured, i.e., to NOT allow the Resource to be downloaded (unless specifically downloaded via Script Code):




The following types of Resource cannot be secured, i.e., they can always be downloaded since they are necessary to render Portal Pages:

- StyleSheet (CSS)
- StyleSheet (LESS)

The following types of Resource are always secured, i.e., they can NEVER be downloaded:

- Script Function Library

All Image type Resources are not secured by default but can be if necessary.

## High DPI Resources

Duplicate resources can be added with a "@2" suffix. When a resource is requested and high DPI mode is detected, the "@2" resource will be returned instead.

For example, say you have a resource with a code of "LOGO" which is your Portal's Logo. This is an image that is 320 pixels wide.

However, high DPI (or Retina) displays benefit from displaying high resolution versions of images such as Logos which may appear blurry at lower resolutions such as 320 pixels.

Therefore, we can create a second resource with a code of "LOGO@2" that is 640 pixels wide. When a resource is requested and High DPI mode is detected, a check will be made for a resource with a suffix of "@2" and, if one is found, this will be used instead of the requested resource.

**NOTE:** When using alternate, High DPI image resources, always specify the image size in your HTML or CSS so that the image is displayed consistently.

For example:

```
<img style="width:320px" src="<%Resource:Logo%>"/>
```

This way, even if the 640 pixel wide image is requested, it will be displayed at the correct size since Web browsers use "virtual" pixels.

# Resource Types

## HTML

HTML type Resources would generally be used to include a common block of HTML on multiple pages.

It can also be used to include self-contained control-type functionality, e.g., CSS and the corresponding JavaScript to implement a Tabs control.

## Images

The following types of Image resources are supported:

- GIF
- JPEG
- PNG
- ICO

**NOTE:** See the High DPI Resources section for details on how to include different image resolutions.

## Documents

Document Type resources would generally be used to allow a static document file to be downloaded, e.g., a brochure.

The following types of Document resources are supported:

- PDF

## StyleSheet (CSS)

Allows a Cascading Style Sheet (CSS) to be defined.

## StyleSheet (Less)

Allows a Cascading Style Sheet (CSS) to be defined but one which can use Less-preprocessing, e.g., to nest styles and perform functions such as lightening or darkening theme colours.

The following site gives more details on Less:

http://lesscss.org/features/

**WARNING:** The LESS compiler can be quite pick and fail to compile to CSS if there are errors such as the use of named system colours or certain formatting mistakes in the source.

## JavaScript

Allows a JavaScript file to be defined.

## Script Function Library

These Resources will be included when compiling the Portal's Script, and also, each of the Portal Page Scripts.

This is the same concept as "Function Library" type Scripts (which can still be used via the #include directive) but there they do not need to be explicitly referenced in the Scripts.

## Text

Allows a plain Text file to be defined.

# State Data

Portals act like Web applications, even when hosted within finPOWER Connect.

They therefore support the concept of Application and Session state.

A special "User Data" state is also supported so that data can be saved between User's sessions, e.g., you may give the user the ability to choose their own background colour for the site.

Each of the `Application`, `Session` and `UserData` properties of the Portal Handler are Key Value Lists (`ISKeyValueList` objects).

## Application

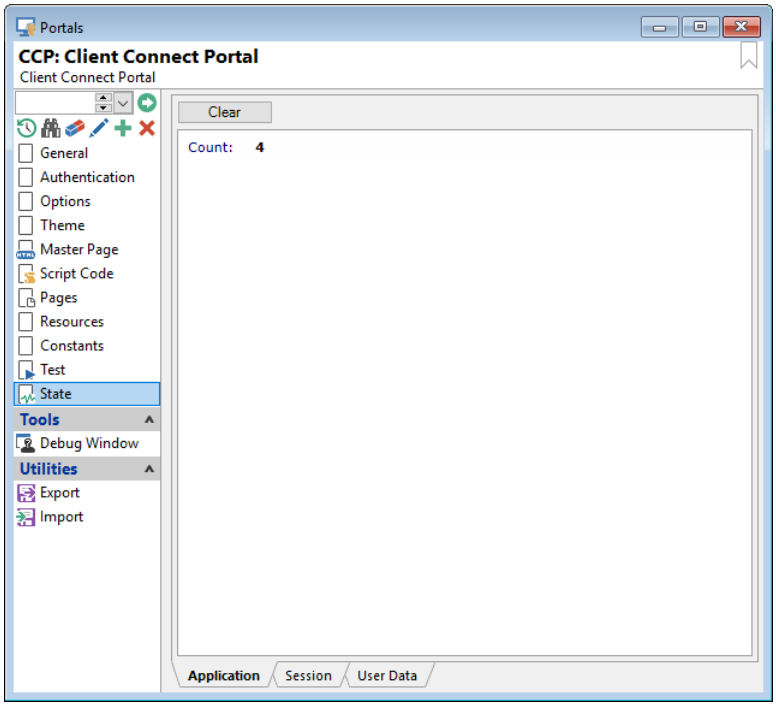
This is used to save and retrieve information for the entire Portal, e.g., a counter that is incremented every time a user requests a certain page.

Since Application state is shared between all Portal users, it must be locked before making any changes to it and then unlocked to commit those changes, e.g.:

```
' Increment Fetch Count
mPortalHandler.ApplicationLock()
With mPortalHandler.Application
  .SetInteger("Count", .GetInteger("Count") + 1)
End With
mPortalHandler.ApplicationUnlock()
```

**IMPORTANT:** Application state should be locked for as brief a period as possible since it will block other requests to the Portal that need to access the Application object.

Application state can be viewed (and cleared) on the "State" page of the Portals form, e.g.:

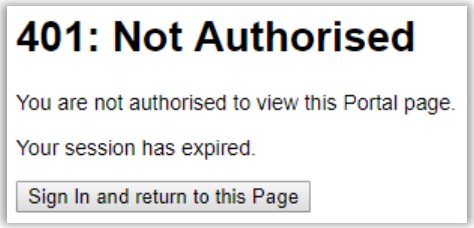


# Session

Session state is used to store information for a current user session (either a finPOWER Connect User, a Client or an Anonymous user).

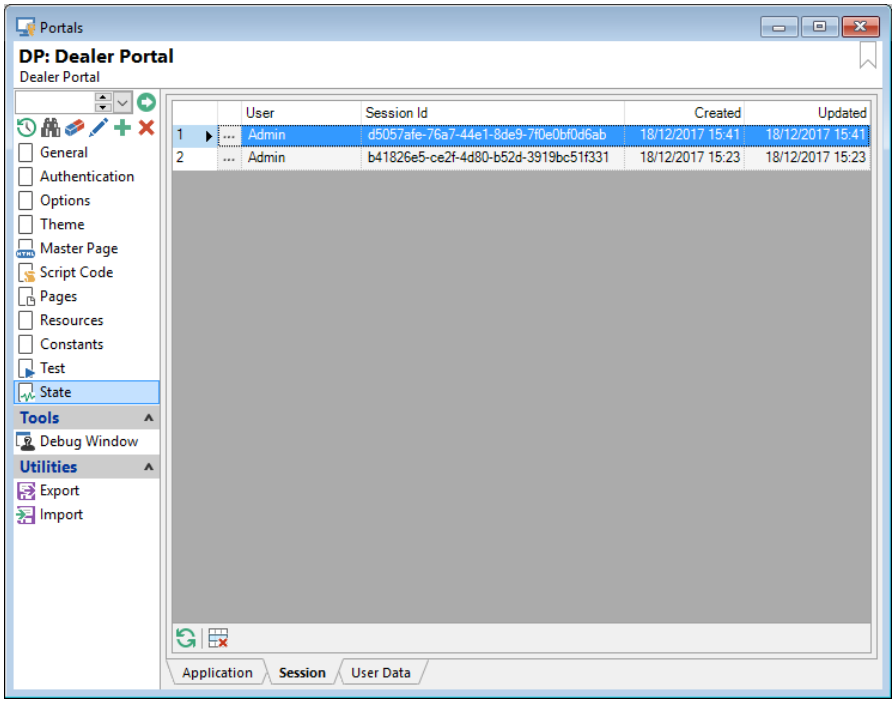
When the Session expires or the user signs out, their session data will be lost.

By default, Session Timeout for a Portal is set to 15 minutes. This means that if no requests have been made for 15 minutes (e.g., moving between Portal Pages) then the Session will be expired and, if a Client or User is signed in, they will receive the following message:




**NOTE:** Unlike Application state, Session state does not need to be locked and unlocked and is saved automatically.

Session state can be viewed (and deleted) on the "State" page of the Portals form, e.g.:




Drilling down on a Session record allows its data to be viewed.

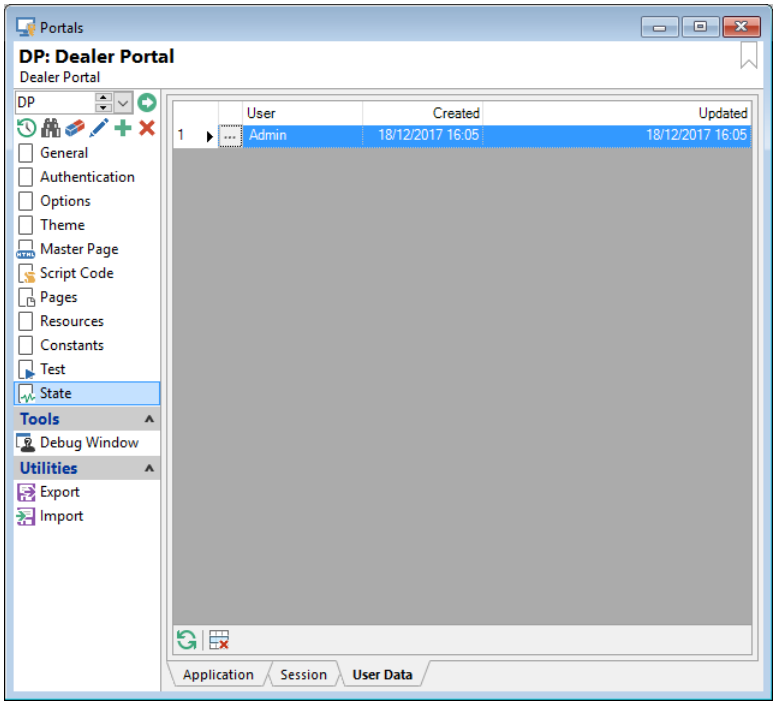
**IMPORTANT:** Deleting a User or Client's Session record will NOT cause the User or Client to be signed out; it will simply remove any Session data they have recorded.

# User Data

User Data is used to store information for a current user (either a finPOWER Connect User or a Client).

**NOTE:** Unlike Application state, User Data state does not need to be locked and unlocked and is saved automatically.

User Data state can be viewed (and deleted) on the "State" page of the Portals form, e.g.:




Drilling down on a User Data record allows its data to be viewed.

**NOTE:** Unlike Session state which stores information only for the User or Client's current session, User Data is stored indefinitely.

# JavaScript objects

The following special JavaScript objects are available to Portals:

- `page`
  - This represents the current page and is a shortcut to a slightly modified `widget` object (the `widget` object can still be used in most circumstances, e.g., when copying and pasting code between HTML Widgets and Portals.
- `portal`
  - The represents the Portal.

## page

As mentioned above, this is a shortcut to the `widget` object but has the following additional properties and methods that are Portal-specific:

| Member | Description |
|---|---|
| widget.PageId | The Id of the currently loaded Page. |
| widget.UI.Modal.LoadPortalPage | Load a Portal Page into a Modal form.<br><br>NOTE: Used by portal.OpenPageInModal which is the preferred way to achieve this. |
| widget.UI.Forms.ShowHtmlWidgetModal | Show an HTML Widget in a Modal form.<br>NOTE: See the Using HTML Widgets in Portals section for more information. |

## portal

All interaction with the Portal is performed via this object.

| Member | Description |
|---|---|
| AllowPasswordResetLink | Indicates whether this Portal is configured to allow a Password Reset Link to be emailed to a Client or User. |
| AuthenticateClient | Authenticate a Client and sign in.<br>NOTE: Used by the Login page for Client-based Portals. |
| AuthenticateUser | Authenticate a User and sign in.<br>NOTE: Used by the Login page for User-based Portals. |
| ChangePassword | Change the password for a signed-in User or Client. |
| DownloadFile | Download a File via a call to the Portal's (rather than the current Page's) Script.<br><br>NOTE: Consider using DocumentManagerFile Application Shortcuts for a non-coding solution to download existing files in the Document Manager. |
| GetPlatformCompatibilityWarning | Get a warning message relating to platform compatibility, e.g., to display on the Sign-In form. |
| GetResourceUrl | Get a URL to a Resource such as an image or a CSS file. |
| GetString | Get a String via a call to the Portal's (rather than the current Page's) Script. |
| HidePwaPrompt | Hide the PWA prompt if it is showing. |
| InsertWidget | Insert an HTML Widget. |

| | |
|---|---|
| IsInModal | Returns `true` if this Portal Page is being shown in a Modal form. |
| PortalId | The Portal Id. |
| Navigate | Navigate to a different Portal page. |
| OpenPageInModal | Open a different Portal page in a Modal form. |
| OpenPortal | Open a different Portal. |
| ParentModalClose | Closes the parent Modal if this Portal Page is hosted in a Modal.<br>NOTE: Does nothing if page is not in a Modal. |
| PasswordResetFromToken | Reset a Client or User Password from the supplied token. |
| PwaEnabled | Indicates whether this Portal has PWA (Progressive Web Application) functionality enabled. |
| PwaName | The PWA application name. |
| SendPasswordResetLink | Send a Password Reset link to a Client or User. |
| SignIn | Go to the Login page.<br>NOTE: Will go to the system-based @LOGIN page unless a custom page named "LOGIN" exists. |
| SignOut | Sign Out of the Portal and return to the Login page. |
| ShowPasswordChange | Shows the Password Change page, optionally, in a Modal form.<br>NOTE: Will go to the system-based @PASSWORD_CHANGE page unless a custom page named "PASSWORD_CHANGE" exists. |
| ShowPwaPrompt | Show PWA prompt providing existing Web browser supports PWAs and this Portal is not already running as a PWA. |
| Version | The current Portals version. |

# Launching a Portal

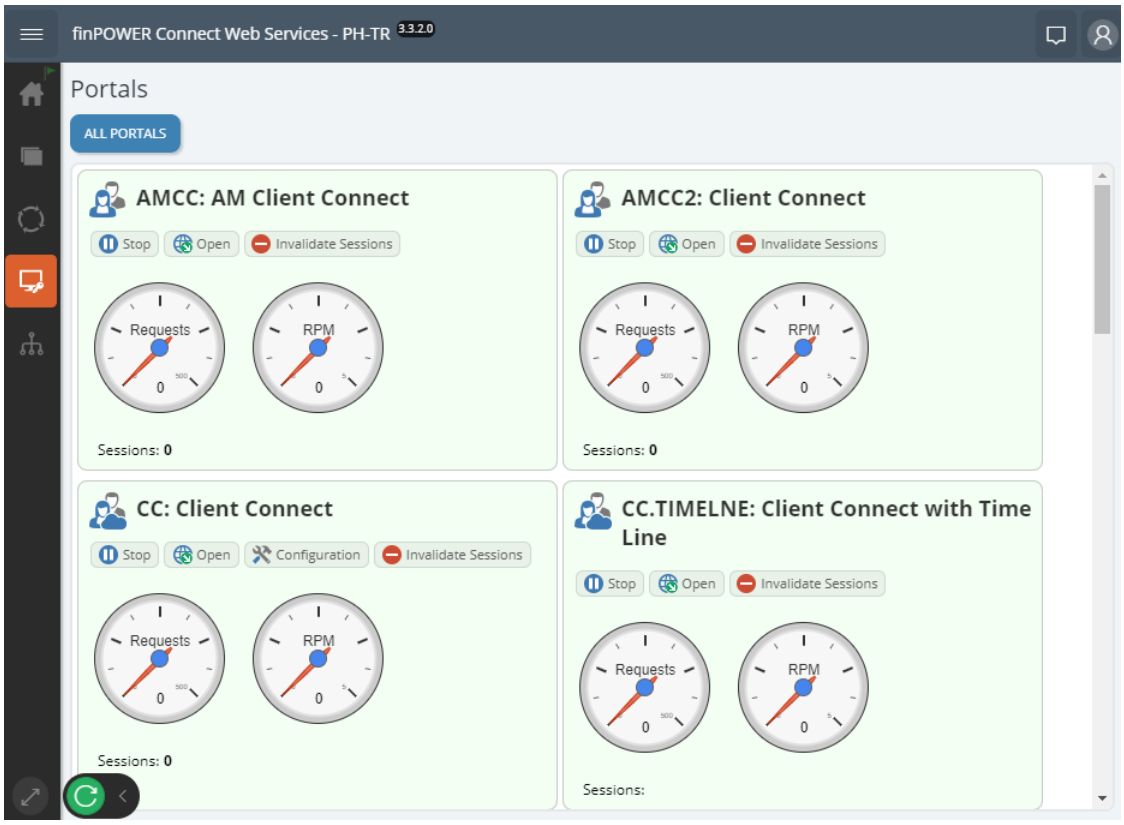
## From within finPOWER Connect

Portals can be run from within finPOWER Connect via Application Shortcuts, e.g.:

```
app://Portal?id=MYPORTAL
```

## From Web Services Administration

The "Portals" view of the Web Services Administration facility allows Portals to be opened:




## In a Web Browser

### Via Web Services

Web Services handle a special "Portal" folder from which all Portals are accessed.

Accessing a Portal is as simple as specifying the Portal Id in the URL, e.g.:

```
https://web-services-url/Portal/MYPORTAL
```

### Via Web Services in an IFRAME

Directing users to a Web Services URL may look a little odd, e.g., if the Portal is hosted within a public company Website.

Hosting the Portal in an IFRAME via a simple HTML page on the company Website makes the Portal appear to be part of that Website.

Below is the complete HTML for a page that does the following:

- Hosts the Portal in an IFRAME.
- Uses the URL hash to display the current Portal page.
  - The allows users to bookmark pages within the Portal.

```
<!DOCTYPE html>
<html>
<head>
  <title>Hosted Portal Sample</title>
</head>
<body>

  <iframe id="portal"
          src="http://localhost:51149/Ws3/Portal/CCP/"
          style="position:absolute; left:0; top:0; right:0; bottom:0;
                 width:100%; height:100%; box-sizing:border-box;">
  </iframe>

  <script>
    // Initialise
    var iframe = document.getElementById("portal");
    var inHashChange = false;

    // Trap changes to iframe URL and propogate to this document's URL Hash
    window.addEventListener("message", function (event) {
      if (event.data && event.data.message) {
        switch (event.data.message) {
          case "portal-url-changed":
            inHashChange = true;
            window.location.hash = event.data.url;
            window.setTimeout(function () { inHashChange = false; }, 10);
             break;

           case "portal-loaded":
             // Navigate now?
             if (window.location.hash && window.location.hash != "#") {
               PortalHavigateHash();
             }
             break;
           }
         }
       }, false);

    // Trap changes to this document's URL Hash and navigate Portal
    window.onhashchange = function () {
      if (inHashChange) return;
        PortalHavigateHash();
      };

    // Navigate the Portal based on the URL Hash
    function PortalHavigateHash() {
      var hash = window.location.hash;
      if (hash && hash.startsWith("#")) hash = hash.substr(1);
      iframe.contentWindow.postMessage({ message: "portal-navigate", url: hash }, "*");
    }
  </script>

</body>
</html>
```

**NOTE:** See the `WebServices/Samples/PortalIframe/PortalHost.htm` file for a sample page.

**WARNING:** Browser security prevents this working if the HTML page containing the IFRAME is loaded from the local file system, i.e., via the "`file://`" protocol.

# Portal Hosting

Portals can be accessed in the following three ways:

1. Directly
2. Directly but in an IFRAME
3. Using the Portal Host Web Application

These options are described in the following sections.

**NOTE:** The recommended access method is using the Portal Host Web Application.

## Directly

Directly via a URL to Web Services, e.g.:

https://demo.intersoft.co.nz/finPOWERConnectWS3/Portal/CC

This is the simplest method since, if your Web Services are exposed publicly, you can simply provide a URL directly to the Portal with no additional work.

This method is great for testing a Portal however it has the following drawbacks:

- You have no control over the URL:
  - i.e., the URL always points to Web Services when you might want to provide Clients with a friendlier URL such as http://clientconnect.mycompanywebsite.com
- Web Services must be externally visible to the Internet.
  - This means that anyone can access your Web Services.
    - Although Web Services do require authenticated access, this is not ideal and finPOWER Connect Cloud using a proxying mechanism so that Web Services can exist on a different server and potentially behind a firewall.

**WARNING:** This method is not recommended since it requires Users or Clients having a direct URL to Web Services and therefore Web Services being publicly available over the internet.

## Directly but in an IFRAME

This method still accesses the Portal directly via Web Services however it disguises this fact from the user by using an HTML IFRAME to make it look as if the Portal is hosted on your own website.

See the Via Web Services in an IFRAME section for more information.

**WARNING:** This method is not recommended since it requires Users or Clients having a direct URL to Web Services and therefore Web Services being publicly available over the internet.

## Using the Portal Host Web Application

Intersoft Systems supply a Portal Host Web Application.

This is installed just like finPOWER Connect Cloud and provides a simple proxying mechanism which allows the Portal to be hosted on your own Website but to access Web Services in a similar manner to finPOWER Connect Cloud.

Once installed, this Application must be configured by directly editing the App_Data/config.xml file to specify the URL of your Web Services and the ID of the Portal to host, e.g.:

```
<ISKeyValueList version="1.00">
  <Item type="String" key="PortalId">CC</Item>
  <Item type="String" key="WebServicesUrl">http://localhost:51149/Ws3/</Item>
```

```
</ISKeyValueList>
```

**WARNING:** Always take a backup copy of your config.xml file before installing a new version of the Portal Host.

# Testing a Portal

There are three different modes a Portal can be accessed from and all should be tested (unless the Portal is only going to be used from a Web Browser via Web Services):

1. From within finPOWER Connect
    a. e.g., via the "Test" button on the Test page of the Portals form.

2. Testing in an External Browser launched from within finPOWER Connect
    a. Via the "Test in Web Browser (Experimental)" button on the Test page of the Portals form.
    b. **NOTE:** This mode is primarily designed for trouble-shooting JavaScript and HTML issues and may behave slightly differently in certain situations; Users will never access the Portal via this method.

3. From Web Services
    a. i.e., as a fully-fledged Website.

You should also ensure that, if the Portal is to be hosted in a Web Browser, that you test:

1. All target browsers, e.g.:
    a. Chrome
    b. Internet Explorer
    c. Microsoft Edge
    d. Firefox
    e. Safari
2. On mobile devices such as:
    a. iPads
    b. iPhones
    c. Android Phones and Tablets

# Performance and Best Practices

Every call to retrieve a Page or a Resource such as an image requires Web Services and therefore involves overhead that can be minimised or avoided completely.

## Inlining Resources vs Retrieving via a URL

Inlining Resources (e.g., expanding a stylesheet so it is included as part of the HTML page) avoids making additional Web Services calls. However, this makes the page size larger in the initialise request, e.g.:

```
<link rel="stylesheet" type="text/css" href="<%ResourceDataUri:CSS%>"/>
```

**NOTE:** In the above example, the content of the CSS resource is encoded into a Data URI and therefore becomes part of the page.

Retrieving a Resource via a URL means that the overall page size is smaller but an additional Web Service call is made t, e.g.:

```
<link rel="stylesheet" type="text/css" href="<%ResourceUrl:CSS%>"/>
```

Using the `ResourceDataUri` tag rather than the `ResourceUrl` tag means that there is less load put on the Web Server. This can be useful for optimising performance but may impact a user's page load times. However, you should bear in mind that most Resources should be cached by the user's web browser rather than being fetched with every page request.

## Embedded vs External Resources

Resources such as images can be embedded within the Portal, e.g., from the Resources page or by creating a Data URI for a Logo on the Theme page, e.g.:


If an image or resource is available at an external location then referencing this instead of embedding the image means that no Web Services call is made and also that no large data URI is embedded in the page.

# Using HTML Widgets in Portals

There are occasions where you may want to use an HTML Widget within a Portal.

For example, you may have an HTML Widget to perform a Loan Quote that you want finPOWER Connect Users to access (via finPOWER Connect Cloud) but also want Clients to be able to access (e.g., from a Client Portal).

## Identifying the Portal

The HTML Widget Script code can use the following to determine the Id of the Portal that it has been called from:

```
mPortalId = requestInfo.HostedInPortalId
```

## Identifying the Signed-In Client

For Client-based Portals, checking the Current User, e.g., via `finBLCurrentUser.UserId` is meaningless.

The HTML Widget Script code can use the following to determine the Id of the Portal's signed-in Client and act accordingly:

```
mClientId = requestInfo.AuthenticatedClientId
```

**NOTE:** When running a Client-based Portal from within finPOWER Connect, `requestInfo.AuthenticatedClientId` will always be a blank String.

An HTML Widget could vary its behaviour if it detects a Client is signed in, e.g.:

- Hiding data-capture pages from an Account Application since the Client is already known.
- Hiding information that a Client should never be able to select, e.g., a Dealer or Disbursement details.

## Using Portal Styling

When an HTML Widget is displayed in a Client Portal, you may wish for it to be styled differently to how a finPOWER Connect Cloud User would see it.

Currently, there is no automatic way to do this. However, if you have a CSS (or LESS) resource in you Portal that you wish to include in you HTML Widget, you can access the resource as shown below:

```
PortalCss = finBL.HtmlWidgetUtilities.GetPortalResourceCss(mPortalId, "MyCSS")
```