Infor System i Workspace AnyWhere

5250 AnyWhere Emulator Extensions Guide

Publication Information

Release: Infor System i Workspace AnyWhere

Publication date: June 6, 2023

Contents

About this guide

This guide covers the programmatic features available to Infor System i Workspace AnyWhere 5250 AnyWhere Emulator users.

Caution: The Infor System i Workspace AnyWhere screen shots within this guide were obtained with the *Infor Design UI Version* option set to **Classic**. If the System Administrator has changed the *Infor Design UI Version* option to **New**, the appearance of the product may differ to the screen shots within this guide, but the documented functionality of Infor System i Workspace AnyWhere will be the same.

Intended audience

Advanced users who have web-programming experience, specifically JavaScript, and a good understanding of the client/server architecture used by Infor System i Workspace AnyWhere.

Some of the terms and concepts used in this guide are covered within the Infor System i Workspace AnyWhere – 5250 AnyWhere Emulator Designer Guide, which should be read first, before reading this guide.

Organization

This table shows the chapters of the guide:

Section	Description
Introduction	Overview of this document.
Events	How events are called.
Methods	The API methods available within JavaScript.
Variables	System information variables that you can use within your Workspace Extensions.

Section	Description
Re-using a Jacada Extension	Information on how to convert a Jacada extension written for a previous release of System i Workspace to use the 5250 AnyWhere Emulator instead.
Standards for Infor System i Workspace AnyWhere Extensions	Information on how to convert Emulator Extensions written for older web-browsers and technology standards to work with Infor System i Workspace AnyWhere.
Re-using a System i Emulator Extension	Information on how to convert a System i Emulator extension written for a previous release of System i Workspace to use the 5250 AnyWhere Emulator instead.
Replacing an Entire Screen	For when you want to replace an entire Application Screen with your own user interface, this chapter explains the new features that Infor System i Workspace AnyWhere provides to make this process as simple as possible.
Examples	Examples of using Infor System i Workspace AnyWhere Extensions.

Related documents

You can find the documents in the product documentation section of the Infor Support Portal, as described in "Contacting Infor" on page 8.

Contacting Infor

If you have questions about Infor products, go to Infor Concierge at https://concierge.infor.com/ and create a support incident.

The latest documentation is available from docs.infor.com or from the Infor Support Portal. To access documentation on the Infor Support Portal, select **Search > Browse Documentation**. We recommend that you check this portal periodically for updated documentation.

If you have comments about Infor documentation, contact documentation@infor.com.

Chapter 1 Introduction

Welcome

The 5250 AnyWhere Emulator allows you to change the standard UI layout and content of a Infor System i Workspace AnyWhere application using a GUI tool called the 5250 AnyWhere Emulator Designer. It also provides a set of JavaScript Extension Functions for more advanced users.

The 5250 AnyWhere Emulator can be used within a wide range of browsers and platforms to provide day-to-day access to your Infor ERP and IBM i tasks. It can also be used to modify screens, without RPG changes, using the built-in 5250 AnyWhere Emulator Designer tool. No additional software is required on the client to use the 5250 AnyWhere Emulator

For brevity, within this guide, we will use the term Designer to reference the 5250 AnyWhere Emulator Designer.

If you wish to add or change your IBM i applications to have custom validation, screen handling, links to 3rd party applications or even replace an entire IBM i application screen with your own, then you will need to use the 5250 AnyWhere Emulator Extensions.

The 5250 AnyWhere Emulator Extensions provide a set of JavaScript events and methods that can be used to add advanced function to your screens.

This guide covers the features and usage of the 5250 AnyWhere Emulator Extensions.

For brevity, within this guide, we will use the term Emulator Extensions to reference the 5250 AnyWhere Emulator Extensions.

This guide assumes that the reader has a good understanding of JavaScript and HTML and the Document Object Model (DOM) within Web Browsers, and XSL/XML. No guidance is provided for these technologies.

Chapter 2 Events

Overview

There are four main Emulator Extension events

- Control Screen Visibility
- Screen Updated (new data received from the server)
- User Action Performed (that will be sent to the server)
- User Requested Help

Each of these events corresponds to a JavaScript function within the Emulator Extensions file.

The Emulator Extensions file

The Emulator Extensions file is called `5250-extensions.js` and is located in the `customScripts` folder of your Workspace static content.

By default, for a Tomcat installation, this file is located in the `C:\Program Files\Infor\SiWAnywhere\tomcat\webapps\systemi\static\customScripts` folder on your Infor System i Workspace AnyWhere server.

By default, for an IBM WebSphere installation, this file is located in the `C:\Program Files\IBM\WebSphere\AppServer\profiles\AppSrv01\installedApps\{server}Node01Cell\SiWAnywhere.ear\Workspace.war\static\customScripts` folder on your Infor System i Workspace AnyWhere server.

This file should be edited with a text editor or suitable IDE (such as Eclipse). All changes should be applied within this file.

After changing this file, you should always restart your Infor System i Workspace AnyWhere server.

Caution: Changes made within this file will affect all Infor System i Workspace AnyWhere profiles where the 5250 AnyWhere Emulator is used. Development of Emulator Extensions should only be carried out on a non-business critical Workspace installation to ensure that no data corruption occurs.

Once your Emulator Extension changes have been thoroughly reviewed and tested, they can be copied to your live Infor System i Workspace AnyWhere server.

Caution: Any code you insert into the Event methods below should **not** block and/or prevent the method from completing in a timely manner which could cause instability within 5250 AnyWhere Emulator tasks. Use the `window.setTimeout` method to execute any code that may take time to complete (such as a synchronous web-request) after the Event method has completed.

Caution: We recommend that you do **not** use the JavaScript `alert` method, or any other JavaScript method that displays a modal dialog, within your Extension code. Modal dialogs may block other executing threads within the application and cause instability within the 5250 AnyWhere Emulator tasks.

Additional Extension files

If you wish to import any additional JavaScript files into a 5250 AnyWhere Emulator webpage, such as 3rd party JavaScript libraries, you can place these files in a sub-folder of the `customScripts` folder within Infor System i Workspace AnyWhere called `5250`. Any file, within this folder, that has the `.js` extension, will be loaded into the page automatically.

By default, for a Tomcat installation, the `customScripts` folder is located in the `C:\Program Files\Infor\SiWAnywhere\tomcat\webapps\systemi\static` folder on your Infor System i Workspace AnyWhere server.

By default, for an IBM WebSphere installation, the `customScripts` folder is located in the `C:\Program Files\IBM\WebSphere\AppServer\profiles\AppSrv01\installedApps\{server}Node01Cell\SiWAnywhere.ear\Workspace.war\static` folder on your Infor System i Workspace AnyWhere server.

Caution: You must ensure that any 3rd party JavaScript files you import do not interfere with or replace code required for either Infor System i Workspace AnyWhere, or the 5250 AnyWhere Emulator, to function correctly. Before reporting problems to Infor Xtreme Support, ensure that the problem is not caused by any additional files you have added to the standard Infor System i Workspace AnyWhere installation.

The Control Screen Visibility event

This event is fired by the 5250 AnyWhere Emulator when a new screen arrives. It allows the visibility of the screen, to the user, to be controlled programmatically.

To enable processing of this event, you need to add the following JavaScript method to `5250-extensions.js`…

```
function CustomHideEmul(strScreenId, strLastScreenId) {
}
```

There is a commented-out example of this method within the standard `5250-extensions.js` shipped with Infor System i Workspace AnyWhere. Remove the comments from the start of each line to enable the function.

The function receives two parameters…

Parameter	Description
`strScreenId`	A string value containing the unique identifier for the current application screen (also known as a Magic Number).
`strLastScreenId`	A string value containing the unique identifier for the previous application screen.

You can find the Screen ID of a screen using the 5250 AnyWhere i Emulator via the context menu *Screen Details* option. The current Screen ID will be displayed in a popup dialog.

Using the two Screen ID/Magic Number parameters, you can decide whether you want to show the screen. If you wish to hide the current screen, you should return a value of **1** from the method; otherwise, if you wish to use the standard Emulator display logic, you should return a value of **0**.

Here is an example of using this event…

```
function CustomHideEmul(strScreenId, strLastScreenId) {
var result = 0;
switch(strScreenId) {
// Hide the first screen of the option 1/OEE
case "OE02X99OE03699":
result = 1;
break;

// Show all other screens
default:
break;
}
return result;
}
```

We recommend that you keep any processing within this function to a minimum. Any screen processing should be handled within the Screen Updated event.

The Screen Updated event

This event is fired by the 5250 AnyWhere Emulator when a new screen arrives at the client, but user-interaction is not yet enabled. It is within this event you should carry out any screen modifications or processing.

To enable processing of this event, you need to add the following JavaScript method to `5250-extensions.js`…

```
function CustomScreenChanged(strScreenId, strLastScreenId,
                             blnPopup, obj5250) {
}
```

There is a commented-out example of this method within the standard `5250-extensions.js` shipped with Infor System i Workspace AnyWhere. Remove the comments from the start of each line to enable the function.

The function receives four parameters…

Parameter	Description
`strScreenId`	A string value containing the unique identifier for the current application screen (also known as a Magic Number).
`strLastScreenId`	A string value containing the unique identifier for the previous application screen.
`blnPopup`	A Boolean value, set to true if the current application screen is within a Popup Window otherwise it will be false.
`obj5250`	A handle to the current Emulator object. It is against this object you will call methods to alter the screen. These methods are covered in a subsequent section of this document.

This function has no return value.

Using the two Screen ID/Magic Number parameters, you can decide whether you want to perform additional processing.

Here is an example of using this event…

```
function CustomScreenChanged(strScreenId, strLastScreenId,
                             blnPopup, obj5250) {
switch(strScreenId) {
// Navigate past the first screen of the option 1/OEE
case "OE02X99OE03699":
obj5250.SetFieldValueByIndex(1, 1, "CUSTOMER");
obj5250.SetFieldValueByIndex(2, 1, "000");
obj5250.SendKey(42);
```

```
break;
// All other screens
default:
break;
}
}
```

We recommend you create JavaScript methods for each screen you wish to process to avoid the main event method getting difficult to read and understand.

Caution: You should make sure that any processing you do always completes in a timely manner or gives the user some feedback that processing is in process, otherwise the end-user may think the application has locked up or is in error.

The User Action Performed event

This event is fired by the 5250 AnyWhere Emulator when the user carries out an action that will send data back to the server. It is within this event you should carry out any additional validating processing, and optionally block the user's action, if necessary.

To enable processing of this event, you need to add the following JavaScript method to `5250-extensions.js`…

```
function CustomActionPerformed(strScreenId, intAction, obj5250) {
}
```

There is a commented-out example of this method within the standard `5250-extensions.js` shipped with Infor System i Workspace AnyWhere. Remove the comments from the start of each line to enable the function.

The function receives three parameters…

Parameter	Description
`strScreenId`	A string value containing the unique identifier for the current application screen (also known as a Magic Number).
`intAction`	An integer value for the user's action (i.e., the aid or function key they have activated via the keyboard or mouse).
`obj5250`	A handle to the current 5250 AnyWhere Emulator object. It is against this object you will call methods to alter the screen. These methods are covered in a subsequent section of this document.

The possible values for the `intAction` parameter are…

intAction Value	User's Action
`42`	Enter
`43 to 66`	Command F1 to F24
`73`	Page Up
`74`	Page Down
`75`	Host Print
`76`	Reset

Using the Screen ID/Magic Number parameter, you can decide whether you want to perform additional validation or block the user's action.

If you wish to block the user's action on the current screen, you should return a value of **1** otherwise, if you wish to use the standard 5250 AnyWhere Emulator processing, you should return a value of **0**.

Here is an example of using this event…

```
function CustomActionPerformed(strScreenId, intAction, obj5250) {
var result = 0;
switch(strScreenId) {
// Block F4 on the 1st screen of 1/ARE
case "SL07K99":
if (intAction == 46) result = 1;
break;
default:
break;
}
return result;
}
```

The User Requested Help event

This event is fired by the 5250 AnyWhere Emulator when the user requests help by pressing/activating the F1 command key or clicking the Question Mark button on the page heading. You can use this function to carry out your own help processing.

To enable processing of this event, you need to add the following JavaScript method to `5250-extensions.js`…

```
function CustomProcessHelpKey(strScreenId, strLastScreenId, blnPopup) {
}
```

The function receives three parameters…

Parameter	Description
`strScreenId`	A string value containing the unique identifier for the current application screen (also known as a Magic Number).
`strLastScreenId`	A string value containing the unique identifier for the previous application screen.
`blnPopup`	A boolean value, set to true if the current application screen is within a Popup Window otherwise it will be false.

If you wish to carry out your own help processing, you should return a value of **1** from this function**.** If you wish to use the 5250 AnyWhere Emulator help processing within the Application Help display, you should return a value of **0**. If you want the 5250 AnyWhere Emulator to perform its default processing based on the current Screen ID, do not return any value from the function (e.g., omit the `return` keyword). If you want to block the standard 5250 AnyWhere Emulator help processing and always send the Host Help command key directly to the IBM i host, you should return a value of **2**.

Here is an example of using this event…

```
function CustomProcessHelpKey(strScreenId, strLastScreenId, blnPopup) {
  if (strScreenId == "ABCDE99") {
    // Use custom help processing for bespoke screen ABCDE99
    doCustomHelp();
    return 1;
  } else if (strScreenId == "SL07K99") {
    // Use Host Help for 1st panel of 1/ARE
    return 2;
  }
}
```

Caution: If this function is available, and 5250 AnyWhere Emulator Scripting is enabled, it will be called when Application Help is activated across all 5250 AnyWhere Emulator screens, whether they have a Screen ID or not. Unless you wish to add your own Help processing for screens without a Screen ID, we recommend that you either explicitly return a value of **2** for those applications (so the standard IBM i Host Help processing is performed), or do not return a value from this function (so the default 5250 AnyWhere Emulator processing is applied – as shown in the example above).

Chapter 3 Methods

Overview

The 5250 AnyWhere Emulator object (the `obj5250` parameter that is passed into the Screen Updated and User Action Performed events) has several methods on it that can be used to control the current application screen. There are also some additional JavaScript objects and methods that can be used to alter the display.

5250 AnyWhere Emulator methods

All these methods are public to the 5250 AnyWhere Emulator object (the `obj5250` parameter) that is passed into Screen Updated and User Action Performed events.

Caution: The 5250 AnyWhere Emulator object has many other public methods which are not documented here. You should not attempt to use these methods under any circumstance.

GetFieldValue

This method retrieves the content of a field within the current IBM i application screen, using the row and column position of the field. This method takes two parameters…

Parameter	Description
`intHostRow`	Row position of required field (0 – 23/27)
`intHostCol`	Column position of required field (0 – 79/132)

This method returns the string value of the field (or a blank string if no field is found at the passed row/column position). Even if a field is hidden, you can still retrieve its value.

The 5250 AnyWhere Emulator display is divided up into a grid of eighty columns and twenty-four rows (or one hundred and thirty-two columns and twenty-seven rows on extended screens). Each field will have a unique row and column parameter position.

This example will return the value from the field on row ten (i.e. the eleventh screen row), column position one (i.e. the second screen column)…

```
var value = obj5250.GetFieldValue(10, 1);
```

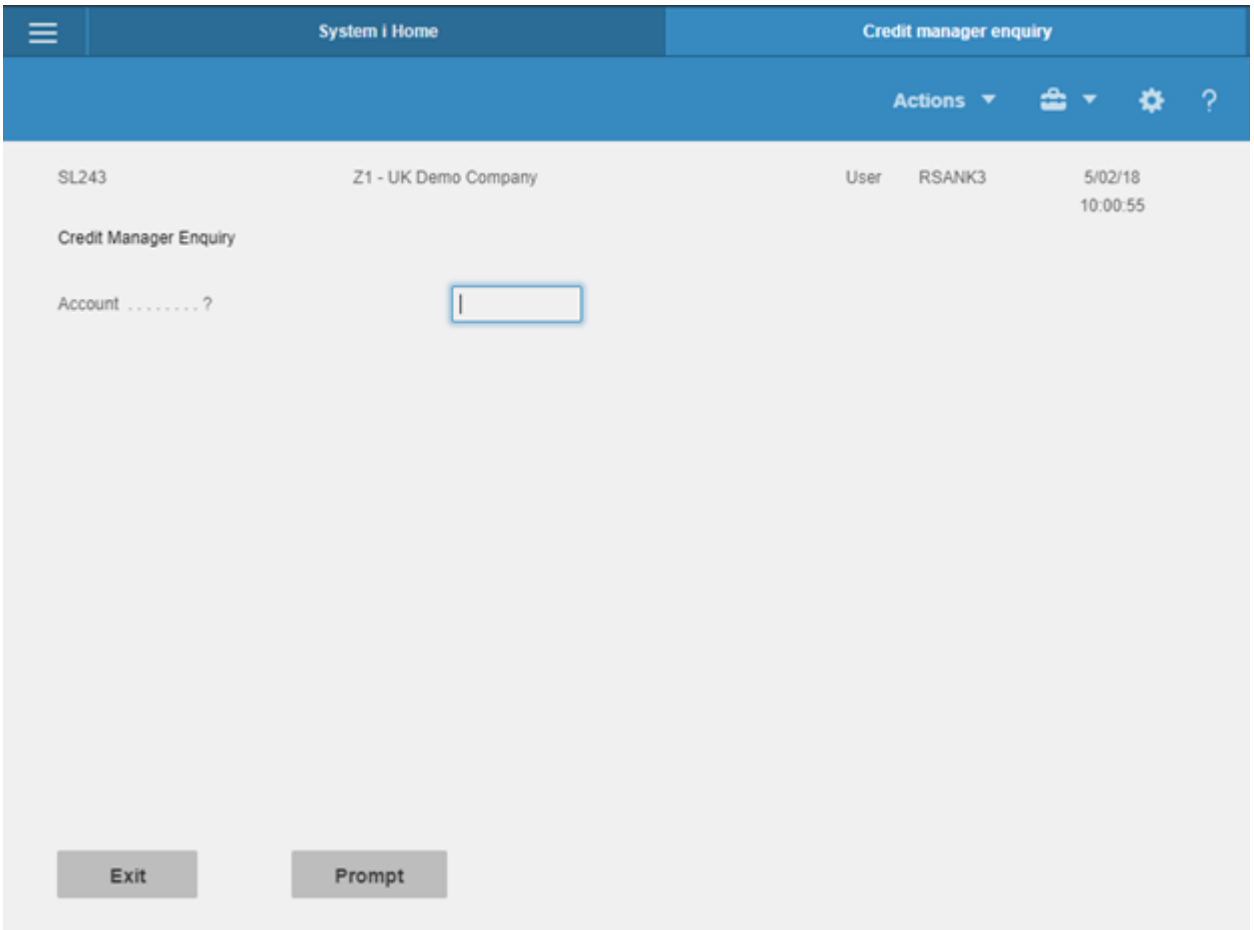
GetFieldValueByIndex

This method retrieves the content of a field within the current IBM i application screen, using the field index. This method takes two parameters…

Parameter	Description
`intIndex`	The field index (starts at 0).
`intFieldType`	The field type (1 = edit, 2= static)

This method returns the string value of the field (or a blank string if no field is found at the passed index). Even if a field is hidden, you can still retrieve its value.

Each field within the 5250 AnyWhere Emulator screen is given a unique index. The indexes are assigned to both static and edit fields starting at the top left corner of the screen and going left to right, then down the screen, for example:



In this screen, the static text *SL243* would be given an index of 0, the text *Z1 – UK Demo Company* would be given an index of 1 and so on. Any buttons will be counted as static text and assigned an index accordingly. As there is only one edit field that will be given an index of 0.

This method returns the string value of the field (or a blank string if no field is found at the passed index).

This example will return the value from the text field at index six...

```
var value = obj5250.GetFieldValueByIndex(6, 2);
```

…which in the above screen is the text `Credit Manager Enquiry`.

SetFieldValue

This method updates the content of a field within the current IBM i application screen, using the row and column position of the field. This method takes three parameters…

Parameter	Description
`intHostRow`	Row position of required field (0 – 23/27)
`intHostCol`	Column position of required field (0 – 79/132)
`strValue`	New value to assign to this field

This method does not return a value. Even if a field is hidden, you can still assign it a value.

This example will set the field value for the field on row twelve, column position ten to *New Value*…

```
obj5250.SetFieldValue(12, 10, "New Value");
```

SetFieldValueByIndex

This method updates the content of a field within the current IBM i application screen, using the field index (see `GetFieldValueByIndex` for more information on indexes). This method takes three parameters…

Parameter	Description
`intIndex`	The field index (starts at 0).
`intFieldType`	The field type (1 = edit, 2= static)
`strValue`	New value to assign to this field

This method does not return a value. Even if a field is hidden, you can still assign it a value.

This example will set the edit field at index two to *New Value*…

```
obj5250.SetFieldValueByIndex(2, 1, "New Value");
```

GetIndexCount

This method retrieves the number of edit or static fields currently available within the current System i application screen. This method takes one parameter…

Parameter	Description
`intFieldType`	The field type (1 = edit, 2= static)

This method returns an integer value containing the count of edit or static fields. The count will contain hidden/non-display fields.

For example, to get the number of edit fields on the current screen…

```
var count = obj5250.GetIndexCount(1);
```

SendKey

This method sends the supplied action to the IBM i application. This method takes one parameter…

Parameter	Description
`intAction`	The action key code

The `intAction` parameter should be set to one of the following values…

intAction Value	User's Action
`42`	Enter
`43 to 66`	Command F1 to F24
`73`	Page Up
`74`	Page Down
`75`	Host Print
`76`	Reset

For example, to simulate the user pressing the *Enter* key…

```
obj5250.SendKey(42);
```

AddTooltip

This method adds a tooltip to the field, regardless of type, at the supplied host row/column position. The tooltip is shown when the user moves the mouse over the field. This method takes three parameters…

Parameter	Description
`intHostRow`	Row position of required field (0 – 23/27)
`intHostCol`	Column position of required field (0 – 79/132)
`strValue`	New tooltip text

This method does not return a value.

This example will add a tooltip for the field at row two, column one…

```
obj5250.AddTooltip(2, 1, "New tooltip");
```

SetFieldColour

This method allows you to apply one of the pre-defined IBM 5250 Terminal colour styles to the field, regardless of type, at the supplied host row/column position. This method takes five parameters…

Parameter	Description
`intHostRow`	Row position of required field (0 – 23/27)
`intHostCol`	Column position of required field (0 – 79/132)
`intColour`	The color to apply to this field (see below)
`blnUnderline`	If true, apply the underline style to the field
`blnReverse`	If true, apply the reverse-image style to the field

The value `intColour` must be one of the following JavaScript variables…

intColour Value	Description
`Emulator_Colour_Green`	Apply the IBM 5250 Green color
`Emulator_Colour_White`	Apply the IBM 5250 White color
`Emulator_Colour_Red`	Apply the IBM 5250 Red color
`Emulator_Colour_Red_Highight`	Apply the IBM 5250 Red Highlight color
`Emulator_Colour_Cyan`	Apply the IBM 5250 Turquoise color
`Emulator_Colour_Yellow`	Apply the IBM 5250 Yellow color
`Emulator_Colour_Pink`	Apply the IBM 5250 Pink color
`Emulator_Colour_Blue`	Apply the IBM 5250 Blue color

This method does not return a value.

This example will apply the Red colour to the field at row five, column ten…

```
obj5250.SetFieldColour(5, 10, Emulator_Colour_Red, false, false);
```

SetFieldColourByIndex

This method allows you to apply one of the pre-defined IBM 5250 Terminal colour styles to the field at the supplied index (see `GetFieldValueByIndex` for more information on indexes). This method takes five parameters…

Parameter	Description
`intIndex`	The field index (starts at 0).
`intFieldType`	The field type (1 = edit, 2= static)
`intColour`	The color to apply to this field (see `SetFieldColour` method for values)
`blnUnderline`	If true, apply the underline style to the field
`blnReverse`	If true, apply the reverse-image style to the field

The value `intColour` must be one of the JavaScript variables described for the `SetFieldColour` method.

This method does not return a value.

This example will apply the Blue colour to the edit field at index six …

```
obj5250.SetFieldColourByIndex(6, 1, Emulator_Colour_Blue,
                              false, false);
```

SetFieldEnabled

This method allows you to enable/disable an Edit field at the supplied host row/column position. This method takes three parameters…

Parameter	Description
`intHostRow`	Row position of required field (0 – 23/27)
`intHostCol`	Column position of required field (0 – 79/132)
`blnEnable`	Set to true to enable the Edit field or false to make the Edit field read-only

This method does not return a value.

This example will make the Edit field at row four and column ten read-only…

```
obj5250.SetFieldEnabled(4, 10, false);
```

SetFieldEnabledByIndex

This method allows you to enable/disable an Edit field at the supplied index (see `GetFieldValueByIndex` for more information on indexes). This method takes two parameters…

Parameter	Description
`intIndex`	The field index (starts at 0).
`blnEnable`	Set to true to enable the Edit field or false to make the Edit field read-only

This method does not return a value.

This example will make the Edit field at index four read-only…

```
obj5250.SetFieldEnabledByIndex(4, false);
```

SetFieldVisible

This method allows you to show/hide a field at the supplied host row/column position. This method takes three parameters…

Parameter	Description
`intHostRow`	Row position of required field (0 – 23/27)
`intHostCol`	Column position of required field (0 – 79/132)
`blnVisible`	Set to true to show the field or false to make the field hidden

This method does not return a value.

This example will hide the field at row four and column ten…

```
obj5250.SetFieldVisible(4, 10, false);
```

SetFieldVisibleByIndex

This method allows you to show/hide a field at the supplied index (see `GetFieldValueByIndex` for more information on indexes). This method takes three parameters…

Parameter	Description
`intIndex`	The field index (starts at 0).
`intFieldType`	The field type (1 = edit, 2= static)
`blnVisible`	Set to true to show the field or false to make the field hidden

This method does not return a value.

This example will hide the Label field at index three …

```
obj5250.SetFieldVisibleByIndex(3, 2, false);
```

SetFieldAlignment

This method allows you to alter the field alignment at the supplied host row/column position. This method takes three parameters…

Parameter	Description
`intHostRow`	Row position of required field (0 – 23/27)
`intHostCol`	Column position of required field (0 – 79/132)
`intAlignment`	One of the pre-defined alignment variables

The value `intAlignment` must be one of the following JavaScript variables…

intAlignment Value	Description
`Emulator_Align_Left`	Left align the field text
`Emulator_Align_Center`	Center align the field text
`Emulator_Align_Right`	Right align the field text
`Emulator_Align_Default`	Restore default field text alignment

This method does not return a value.

This example will right-align the field at row four and column ten…

```
obj5250.SetFieldAlignment(4, 10, Emulator_Align_Right);
```

SetFieldAlignmentByIndex

This method allows you to alter the field alignment at the supplied index (see `GetFieldValueByIndex` for more information on indexes). This method takes three parameters…

Parameter	Description
`intIndex`	The field index (starts at 0).
`intFieldType`	The field type (1 = edit, 2= static)
`intAlignment`	One of the pre-defined alignment variables

The value `intAlignment` must be one of the JavaScript variables described for the `SetFieldAlignment` method.

This method does not return a value.

This example will right-align the static field at index ten…

```
obj5250.SetFieldAlignmentByIndex(10, 2, Emulator_Align_Right);
```

GetFieldAlignment

This method allows you to read the alignment of the field at the supplied host row/column position. This method takes two parameters…

Parameter	Description
`intHostRow`	Row position of required field (0 – 23/27)
`intHostCol`	Column position of required field (0 – 79/132)

This method returns an integer value containing the current field alignment. Use the JavaScript variables described for the `SetFieldAlignment` method above to check which alignment is applied.

This example will return the alignment of the field on row ten, column position one…

```
var value = obj5250.GetFieldAlignment(10, 1);
```

GetFieldAlignmentByIndex

This method allows you to read the alignment of the field at the supplied index (see `GetFieldValueByIndex` for more information on indexes). This method takes two parameters…

Parameter	Description
`intIndex`	The field index (starts at 0).
`intFieldType`	The field type (1 = edit, 2= static)

This method returns an integer value containing the current field alignment. Use the JavaScript variables described for the `SetFieldAlignment` method above to check which alignment is applied.

This example will return the alignment from the edit field at index two…

```
var value = obj5250.GetFieldAlignmentByIndex(2, 1);
```

GetFieldAttribute

This method allows you to read the attribute of the field at the supplied host row/column position. The attribute value can be used to determine the colour, enabled/disabled state, visibility, underline, or reverse image style that is applied to the field. This method takes two parameters…

Parameter	Description
`intHostRow`	Row position of required field (0 – 23/27)
`intHostCol`	Column position of required field (0 – 79/132)

This method returns an integer value containing the current field attribute. Use these JavaScript variables with the returned value to check which attributes are applied…

intColour Value	Description
`Emulator_Colour_Green`	The IBM 5250 Green color
`Emulator_Colour_White`	The IBM 5250 White color
`Emulator_Colour_Red`	The IBM 5250 Red color
`Emulator_Colour_Red_Highight`	The IBM 5250 Red Highlight color
`Emulator_Colour_Cyan`	The IBM 5250 Turquoise color
`Emulator_Colour_Yellow`	The IBM 5250 Yellow color
`Emulator_Colour_Pink`	The IBM 5250 Pink color
`Emulator_Colour_Blue`	The IBM 5250 Blue color
`Emulator_Colour_Reverse`	The field has the Reverse Image style applied
`Emulator_Colour_Underline`	The field has the Underline style applied
`Emulator_Read_Only_Field`	The field has the Read-Only style applied
`Emulator_Hidden_Field`	The field is hidden

This example will check if the field at row ten, column position one is hidden…

```
if ((obj5250.GetFieldAttribute(10, 1) & Emulator_Hidden_Field) ==
    Emulator_Hidden_Field) console.log("Field is hidden");
```

GetFieldAttributeByIndex

This method allows you to read the attribute of the field at the supplied index (see `GetFieldValueByIndex` for more information on indexes). The attribute value can be used to determine the colour, enabled/disabled state, visibility, underline, or reverse image style that is applied to the field. This method takes two parameters…

Parameter	Description
`intIndex`	The field index (starts at 0).
`intFieldType`	The field type (1 = edit, 2= static)

This method returns an integer value containing the current field attribute. Use the JavaScript variables described for the `GetFieldAttribute` method above to check which attributes are applied.

This example will check if the edit field at index two is hidden…

```
if ((obj5250.GetFieldAttributeByIndex(2, 1) & Emulator_Hidden_Field) ==
    Emulator_Hidden_Field) console.log("Field is hidden");
```

GetVariable

This method retrieves the content of a Designer Variable within the current IBM i application screen. See the Infor System i Workspace AnyWhere – 5250 AnyWhere Emulator Designer Guide for more details about Designer Variables. This method takes one parameter…

Parameter	Description
`strName`	String containing the name of the Designer Variable

This method returns the string content of the Designer Variable (or a blank string if no Designer Variable is found or the Designer Variable has not yet been assigned a value).

This example will return the value from the Designer Variable called “Last Order Number” …

```
var value = obj5250.GetVariable(“Last Order Number”);
```

SetVariable

This method sets the content of a Designer Variable within the current IBM i application screen. See the Infor System i Workspace AnyWhere – 5250 AnyWhere Emulator Designer Guide for more details about Designer Variables.

Caution: The Designer Variable must have been created within Infor System i Workspace AnyWhere Designer UI before it can be assigned a value.

This method takes two parameters…

Parameter	Description
`strName`	String containing the name of the Designer Variable
`strValue`	String containing the new value to assign to the Designer Variable

This method returns a Boolean value to inform whether the new value was set within the Designer Variable (TRUE for success, FALSE for failure).

Caution: Using this method within the 5250 AnyWhere Emulator will always result in a value of TRUE being returned, as the variable storage is held on the webserver, and updates are performed using asynchronous message calls. The FALSE variant is documented for backward compatibility with extensions written for older Infor System i Workspace AnyWhere versions.

This example will set the value of the Designer Variable called “Last Order Number” …

```
obj5250.SetVariable(“Last Order Number”, “1000”);
console.log(obj5250.GetVariable(“Last Order Number”));
```

SetButtonHighlight

This method allows you to alter the highlight of the button field at the supplied host row/column position. This method takes three parameters…

Parameter	Description
`intHostRow`	Row position of required field (0 – 23/27)
`intHostCol`	Column position of required field (0 – 79/132)
`intHighlight`	One of the pre-defined highlight variables

The value `intHighlight` must be one of the following JavaScript variables…

intHighlight Value	Description
`Emulator_Highlight_Off`	Remove any highlighting from the button
`Emulator_Highlight_More`	Show a “more detail” icon after the button text. This is typically used to indicate that if the user presses the button, further related information has been defined within the application
`Emulator_Highlight_Info_Icon`	Show an Informational Icon (blue circle with a white “i” in the middle) within the button
`Emulator_Highlight_Alert_Icon`	Show an Alert Icon (orange triangle with an exclamation mark within the middle) within the button
`Emulator_Highlight_Error_Icon`	Show an Error Icon (red circle with a white exclamation mark in the middle) within the button

Caution: The `Emulator_Highlight_Orange` option is also supported for backward compatibility. This will display a “more” icon next to the button text, as orange highlighting on buttons is not supported in the Infor Design System standard used by the 5250 AnyWhere Emulator UI.

This method does not return a value.

This example will apply the “more” icon to the button field at row twenty-three and column ten…

```
obj5250.SetButtonHighlight(23, 10, Emulator_Highlight_More);
```

SetButtonHighlightByIndex

This method allows you to alter the highlight of the button field at the supplied index (see `GetFieldValueByIndex` for more information on indexes). This method takes three parameters…

Parameter	Description
`intIndex`	The field index (starts at 0).
`intFieldType`	The field type (2= static)
`intHighlight`	One of the pre-defined highlight variables

The value `intHighlight` must be one of the JavaScript variables described for the `SetButtonHighlight` method.

This method does not return a value.

This example will apply the “more” icon to the button field at index ten…

```
obj5250.SetButtonHighlightByIndex(10, 2, Emulator_Highlight_More);
```

SetFieldFormat

Caution: This method is not supported by the 5250 AnyWhere Emulator. The method definition is provided so that existing extensions do not stop working, but it does not perform any action. It is documented here purely for backward compatibility.

This method applies the supplied Field Format to the field identified by the supplied row and column position. This method takes three parameters…

Parameter	Description
`intHostRow`	Row position of required field (0 – 23/27)
`intHostCol`	Column position of required field (0 – 79/132)
`strFormat`	Field Format to apply to this field

This method does not return a value. Even if a field is hidden, you can still apply a Field Format.

This example will apply the Field Format for the field on row twelve, column position ten to *New Value*…

```
obj5250.SetFieldFormat(12, 10, "#,###.00");
```

SetFieldFormatByIndex

Caution: This method is not supported by the 5250 AnyWhere Emulator. The method definition is provided so that existing extensions do not stop working, but it does not perform any action. It is documented here purely for backward compatibility.

This method applies the supplied Field Format to the field identified by the supplied index (see `GetFieldValueByIndex` for more information on indexes). This method takes three parameters…

Parameter	Description
`intIndex`	The field index (starts at 0).
`intFieldType`	The field type (2= static)
`strFormat`	Field Format to apply to this field

This method does not return a value. Even if a field is hidden, you can still apply a Field Format.

This example will apply the Field Format to the Output field at index ten…

```
obj5250.SetFieldFormatByIndex(10, 2, "#,###.00");
```

JavaScript Objects

div2

The `div2` object is a HTML DIV DOM element that can be used to put additional HTML output into the 5250 AnyWhere Emulator display. The name of the DIV element is “div2”.

The `div2` object is created when the 5250 AnyWhere Emulator task is first launched and is hidden by default. The content of `div2` will initially be empty.

Any changes to the content and position of `div2` should be performed using the JavaScript methods below. You may choose to apply additional CSS changes to `div2`, if these do not interfere or effect the function of the 5250 AnyWhere Emulator display.

Caution: The 5250 AnyWhere Emulator will never change the state or content of the `div2` object so it will be up to your Emulator Extension code to modify, show and hide `div2` as appropriate to your requirements.

JavaScript Methods

setDiv2

This method updates the content of the `div2` HTML object within the tab of each 5250 AnyWhere Emulator task.

This method takes a single parameter…

Parameter	Description
`strHTML`	A string value containing HTML that will be applied into the `div2` HTML object.

This method has no return value.

Here is an example of using this method to output the current screen passed to the Screen Updated event…

```
function CustomScreenChanged(strScreenId, strLastScreenId,
                             blnPopup, obj5250) {
      setDiv2("<h1>" + strScreenId + "</h1>");
      showDiv2(true);
}
```

showDiv2

This method controls the visibility of the `div2` HTML object within the tab of each 5250 AnyWhere Emulator task.

This method takes a single parameter…

Parameter	Description
`blnShow`	A Boolean value. Set to true to show the `div2` HTML object or false to hide it.

This method has no return value.

Here is an example of using this method to only output the current screen passed to the Screen Updated event when the first task screen is displayed…

```
function CustomScreenChanged(strScreenId, strLastScreenId,
                             blnPopup, obj5250) {
if (strLastScreenId == "AM18099") {
            setDiv2("<h1>" + strScreenId + "</h1>");
            showDiv2(true);
} else
            showDiv2(false);
}
```

clearDiv2

This method will remove all the content of the `div2` HTML object within the tab of each 5250 AnyWhere Emulator task.

This method has no parameters.

This method has no return value.

Here is an example of using this method to only output the current screen passed to the Screen Updated event when the first task screen is displayed…

```
function CustomScreenChanged(strScreenId, strLastScreenId,
                             blnPopup, obj5250) {
if (strLastScreenId == "AM18099")
            setDiv2("<h1>" + strScreenId + "</h1>");
else
            clearDiv2();
}
```

moveDiv2

This method will move the position of the `div2` HTML object within the 5250 AnyWhere Emulator page.

This method takes a single parameter…

Parameter	Description
`nPos`	Position of the `div2`. Pass 0 to move below the 5250 AnyWhere Emulator (default) or 1 to move to the right of the 5250 AnyWhere Emulator.

This method has no return value.

Here is an example of using this method to move the `div2` HTML object to the right of the 5250 AnyWhere Emulator for a specific task…

```
function CustomScreenChanged(strScreenId, strLastScreenId,
                             blnPopup, obj5250) {
if (strScreenId == "SL07K99") {
            setDiv2("<h1>" + strScreenId + "</h1>");
            moveDiv2(1);
            showDiv2(true);
} else if (strLastScreenId == "AM18099") {
            showDiv2(false);
            moveDiv2(0);
   }
}
```

showEmul

This method controls the visibility of the 5250 AnyWhere Emulator within the current tab.

This method takes a single parameter…

Parameter	Description
`blnShow`	A Boolean value. Set to true to show the 5250 AnyWhere Emulator or false to hide it.

This method has no return value.

Here is an example of using this method with the first screen of the Account Enquiry (1/ARE) from the Infor System21 ERP application. On first entry to the application, the first screen will be automatically hidden and replaced by two buttons (defined within the `div2` HTML object) that can either show or hide the current screen dynamically when pressed. These buttons call the `showEmul` method.

```
function CustomHideEmul(strScreenId, strLastScreenId) {
var result = 0;

switch(strScreenId) {
case "SL07K99":
result = 1;
break;

}
return result;
}


function CustomScreenChanged(strScreenId, strLastScreenId,
                             blnPopup, obj5250) {
   if (strScreenId === "SL07K99") {
      setDiv2("<p>Hidden panel</p><br/><button class=\"btn-primary\"
   onclick=\"showEmul(true);\"value=\"Show Emulator\">Show
   Emulator</button><button class=\"btn-secondary\"
   onclick=\"showEmul(false);\"value=\"Hide Emulator\">Hide
   Emulator</button>");
      showDiv2(true);
   }
}
```

Caution: This method should only be used to control the visibility of the 5250 AnyWhere Emulator **after** the screen has been processed and is active (e.g. in response to a user action in either the Emulator itself or in your own extension code). If you wish to hide the screen **before** it has been processed and activated, the `CustomHideEmul` event should be used instead.

is5250AnyWhereEmulator

Caution: This method is documented here purely for backward compatibility.

This method can be called from any Extension Event to determine if the user is running the current task using the 5250 AnyWhere Emulator.

This method takes no parameters.

This method will always return a Boolean value of **true**.

Chapter 4 Variables

Available values

There are several JavaScript variables available that may be useful to developers of Emulator Extensions.

Caution: You should **never** update or alter the content of these variables.

Variable	Description
g_profile_id	The Infor System i Workspace AnyWhere profile identifier that the task was ran under.
g_app	The System Manager application code for the current task.
g_version	The System Manager version code for the current task.
g_release	The System Manager release code for the current task.
g_task	The System Manager task code for the current task.
g_command	The IBM i command (if the current task is not a System Manager task).
g_company	The System Manager company code that this task was launched with.
SESSION_ID	The unique session id for this Infor System i Workspace AnyWhere web session (same across all 5250 AnyWhere Emulator tasks).
AURORA_URL	The URL to the Infor System i Workspace AnyWhere server.
STATIC_URL	The URL to the Infor System i Workspace AnyWhere server's static content folder
g_user	The Infor System i Workspace AnyWhere user profile that this task was launched with.
g_locale	The Infor System i Workspace AnyWhere locale that this task was launched with.
g_env	The Infor System i Workspace AnyWhere environment code that this task was launched with.

Chapter 5 Re-using a Jacada Extension

Overview

An extension written for the Jacada UI (the UI used in pre-Infor System i Workspace AnyWhere releases) cannot be used directly with the 5250 AnyWhere Emulator without some re-work. The main change is to switch from using the XSL driven extensions that the Jacada UI uses to the JavaScript events that the 5250 AnyWhere Emulator uses. Some other important changes are…

- In Jacada, screens were recognised using the panel name defined within the Jacada ACE tool. The 5250 AnyWhere Emulator uses the Screen ID/Magic Number assigned to each screen.
- In Jacada, fields are referenced using the field name defined within the Jacada ACE tool. The 5250 AnyWhere Emulator uses either the field index or the host row/column position.

Matching a screen

For Jacada, your client-side XSL may have had the following template style to detect when a new screen was displayed...

```
<xsl:template match="panel[(@name='PANEL') and @state='before']">
<!-- your code here -->
</xsl:template>
```

Within this template you will have written any extension code for the Jacada screen that you needed to apply before the user got control. This XSL template is equivalent to the Screen Updated event within 5250 AnyWhere Emulator pages, for example:

```
function CustomScreenChanged(strScreenId, strLastScreenId,
                             blnPopup, obj5250) {
  switch(strScreenId) {
    case "MAGIC":
      // Your code here
      break;
    // All other screens
```

```
        default:
          break;
        }
    }
```

Updating div2

For Jacada, your client-side XSL may have had the following code...

```
<xsl:value-of select="java:eval($transformer,'setDiv2',string($myHtml))"/>
```

This was used to update the `div2` HTML object.

For 5250 AnyWhere Emulator extensions, you would use the `setDiv2` JavaScript method to set the content and the `showDiv2` JavaScript method to show/hide it.

Retrieving a field value

For Jacada, your client-side XSL may have had the following code...

```
<xsl:variable name="panel"
              select="java:getApplicationPanel($transformer)"/>
<xsl:variable name="panel-class" select="java:getClass($panel)"/>
<xsl:variable name="field" select="java:getField($panel-
class,'FIELD1')"/>
```

This was used to retrieve a value from a named Jacada field.

For 5250 AnyWhere Emulator extensions, you would use either the `GetFieldValue` or `GetFieldValueByIndex` methods.

Updating a field value

For Jacada, your client-side XSL may have had the following code...

```
<xsl:value-of select="java:overrideField($transformer,'FIELD1','VALUE')"/>
```

This was used to set a value of a named Jacada field.

For 5250 AnyWhere Emulator extensions, you would use either the `SetFieldValue` or `SetFieldValueByIndex` methods.

Pressing a key

For Jacada, your client-side XSL may have had the following code...

```
<xsl:variable name="press-cancel"
      select="java:activate($transformer,'CMDCAN')"/>
```

This was used to press a key on the current screen.

For 5250 AnyWhere Emulator extensions, you would use the `SendKey` method.

Chapter 6 Standards for Infor System i Workspace AnyWhere Extensions

Overview

Infor Infor System i Workspace AnyWhere has been re-written to support the latest Hyper Text Markup Language (HTML) and Cascading Style Sheet (CSS) standards defined by the World Wide Web Consortium (W3C http://www.w3.org/).

In previous releases of System i Workspace, the System i Workspace main page and any System i Emulator pages containing Extensions may have run in what is known as “Quirks Mode”. When this page mode was used, W3C standards were not strictly applied. Old, non-standard styles and elements could be used in your System i Workspace extension code. However, at the same time, not all the new standards could be used and layout of elements within pages was more difficult.

In Infor System i Workspace AnyWhere, the main page and all the HTML pages that run inside now have a HTML5 standard document type (DOCTYPE) as the first element in the page, e.g.

```
<!DOCTYPE html>
```

This tells the browser to run in “Standards Mode” and implies that all the code within the page meets the W3C standards that the current browser allows/supports.

The `div2` element, which will contain any extension HTML, is a HTML `DIV`. Initially it has no width/height style applied so it will inherit the width and height from any child elements you add to it. If you use the `moveDiv2` method to move the `div2` element to the right of the Infor System i Workspace AnyWhere Emulators, it will have a width style applied that is calculated using the page-width minus the width of the Infor System i Workspace AnyWhere Emulator display.

Caution: Infor System i Workspace AnyWhere now supports multiple web-browser and deployment platforms. You should ensure that any extension code you write does not use any browser-specific methods or styles that only work in one type of web browser.

Impact for New Extensions

The impact will depend on the type of extension you are planning to write. If you are planning to use the standard Infor System i Workspace AnyWhere Emulator events and methods (i.e., pure JavaScript), to provide custom field validation for example, then these should just be used as described in the guide and follow the browser's JavaScript standards.

However, if you plan to insert HTML elements into the page (using the `setDiv2` method) or include other external pages using an `IFRAME` element, then you need to make sure that these elements and pages are compliant with the HTML and CSS W3C standards of the browser.

W3C provides a CSS standard checking website that is very useful for checking any pages that you are planning to host within Infor System i Workspace AnyWhere http://jigsaw.w3.org/css-validator/ .

For general web development reference across multiple browsers releases and technologies, the W3SCHOOLS site is an invaluable resource http://www.w3schools.com/.

The 5250 AnyWhere Emulator pages also include the jQuery API JavaScript extensions. At time of writing, this is jQuery version 3.6.0. You can find out more about jQuery here http://jquery.com/.

Note: If you need help with writing an extension, please contact Infor Consulting Services via Infor Xtreme Support or your Infor Account Manager.

Impact for Existing Extensions

All the information and references in the previous section are relevant to uplifting an Extension from a previous release of System i Workspace to Infor System i Workspace AnyWhere.

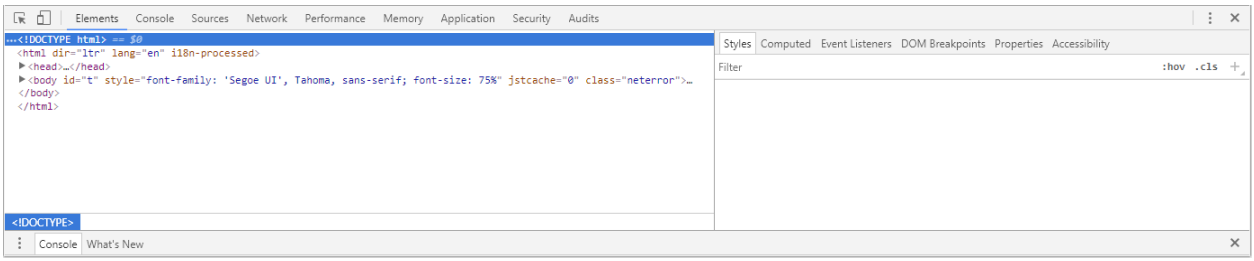
For existing Emulator Extensions that use no HTML/CSS then you should review the 5250 AnyWhere Emulator methods and events in this document to make sure you are using them correctly. As a general principal, we try not to change these methods and events between releases, so that Emulator Extensions from previous releases continue to work, though we may extend them from time to time to add new features.

If you have created Emulator Extensions that dynamically add HTML elements and CSS styles to a page using the `setDiv2` method, then you need to check they are standards compliant, using the resources in this and the previous chapters.

The level of standards depends on the type/version of the web browser you are using within your enterprise, and the type of HTML/CSS Extension you are writing, but as a guideline, we would recommend trying to reach the highest W3C standard you can.

If you embed IFRAME elements to other webpages, then they could also be affected. We would recommend that any pages that are embedded within Infor System i Workspace AnyWhere should use the HTML5 document type. You should then be able to test them outside of Infor System i Workspace AnyWhere to check they function correctly.

To check the standards of an external page, you can use your browsers “Developer Tools” option. Open the external page in your browser, start Developer Tools and it will show you the page source, E.g.



Note: If you need help with upgrading your existing Emulator Extension, please contact Infor Consulting Services via Infor Xtreme Support or your Infor Account Manager.

Common Problems

Here are some issues that Infor have seen within existing customer Emulator Extensions, with steps to resolve them.

Missing Units

Since CSS1, you must specify units when using numeric values for such things as width, height, and position. For example…

```
setDiv2('<div style="width:1000; height 35; background-color:
        green;">Some text</div>');
```

In the above, we intended for a wide green bar at the bottom of the page, but the bar will be sized to the width of the text because the units are missing. Adding units, which in most cases is for pixels (`px`) is the only change required, E.g.

```
setDiv2('<div style="width:1000px; height 35px; background-color:
        green;">Some text</div>');
```

Percentage Units

Percentage units should only be used when you know that the container (parent element) has a size that a percentage can be calculated against. For example…

```
setDiv2('<div style="width:50%; background-color: green;">Some
         text</div>');
```

As the `div2` element does not have explicit width/height values, unless you have used the `moveDiv2` method where a width value will be set, the 50% is essentially being applied to a block layout of “auto”, and therefore ignored. To resolve this, you need to insert an additional element with a fixed size, E.g.

```
setDiv2('<div style="width: 1000px;"><div style="width:50%;
         background-color: green;">Some text</div></div>');
```

The same applies for pages that you embed within the Extension. For example, if your `IFRAME` element has no size, and elements within the page rely on a size being set, you will see issues, E.g.

Extension Code:

```
setDiv2('<iframe src="https://myextension/page1.html"/> ');
```

page1.html :

```
<!DOCTYPE html>
<html>
  <body>
    <div style=”width: 75%”>My Extension Page</div>
  </body>
</html>
```

The `BODY` and `HTML` elements have no size applied, and neither does the `IFRAME`. In this scenario, you would typically set a fixed size on the `IFRAME` element and then use percentage values on the `HTML` and `BODY` elements so the page fits inside the `IFRAME` fully, E.g.

Extension Code:

```
setDiv2('<iframe style="width: 1000px; height 200px;"
         src="https://myextension/page1.html"/> ');
```

page1.html:

```
<!DOCTYPE html>
<html style="width:100%; height:100%">
  <body style="width:100%; height:100%;">
    <div style="width: 75%">My Extension Page</div>
  </body>
</html>
```

In this scenario, to avoid style repetition, you would typically create a new CSS class (in a style element or external CSS file) that has the width and height values assigned and then use the class attribute to assign it to an element. There is a class within the Infor System i Workspace AnyWhere `aurora.css` class file that does this (`full`); E.g.

CSS file:

```
.full {
width: 100%;
height: 100%;
}
```

page1.html:

```
<!DOCTYPE html>
<html class="full">
  <head>
    <link href="test.css" rel="stylesheet" type="text/css"/>
  </head>
  <body class="full">
    <div style="width: 75%">My Extension Page</div>
  </body>
</html>
```

It is therefore very important to make sure that any pages that you host within the Emulator Extension are written to be displayed within an `IFRAME` and do not assume they will be the top-level page element.

Using Tables for Layout

Tables should be used for displaying tabular data, such as a list of stock order lines, and not for layout. In general, the `DIV` element should be used for layout, as it is more flexible and has associated CSS styles and features for doing this.

Using HTTPS for URL access

By default, and by recommendation, Infor System i Workspace AnyWhere is installed using an SSL Certificate, ideally from a Trusted Root Authority, so any references to Emulator Extension pages or files that reside on the Infor System i Workspace AnyWhere server should use the HTTPS URL protocol (or use the `AURORA_URL` or `STATIC_URL` variables).

Any webpages or files that reside on other servers will need to be accessed over the HTTPS protocol too, otherwise they will be blocked as modern web browsers which prohibit the facility to run unsecured content inside a secured parent web site.

Cross-site/domain Access

If your Emulator Extension includes pages from third-party sites that come from a different server or web domain that are housed in a `FRAME` or `IFRAME` element, you may find that the web-browser will block the page. To resolve this, the page would need to be modified to specify the Infor System i Workspace AnyWhere URL as part of its Content Security Policy HTTP header using the `frame-src` policy (see https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Content-Security-Policy for further details). This informs the web browser that your page is allowed to be hosted within the Infor System i Workspace AnyWhere page.

Caution: Some 3rd party web sites no longer allow themselves to be hosted inside other web sites as framed elements. Unless you have access to the configuration of the 3rd party site source/web server, it is unlikely that this type of access blocking can be bypassed.

Internet Explorer XML Processing

Emulator Extensions written for Internet Explorer may need to be re-written if its non-standard features were used (as noted above), but also if ActiveX components/objects were created using the `ActiveXObject` method, particularly for XML processing using the Microsoft XML DOM, e.g.

```
var xmlDoc = new ActiveXObject("Microsoft.XMLDOM");
xmlDoc.async = "false";
xmlDoc.loadXML(xmlString);
```

This `ActiveXObject` method is not supported in other web browsers, so you may want to use standard HTML DOM methods, or use the `DOMParser` object, e.g.

```
var parser = new DOMParser();
var xmlDoc = parser.parseFromString(xmlString,"text/xml");
```

User Experience Changes

Infor System i Workspace AnyWhere uses the Infor Design System (IDS) user experience. This means that if you have written your existing Extensions to utilise HTML controls or CSS styles from previous Infor User Experience versions, then they will need to be changed to work with the IDS user experience ones. If you have referenced older JavaScript or CSS files from the `static/infor10/ux30` sub-folder in Infor System i Workspace AnyWhere, then you will need to change these references, as this folder no longer exists.

The new Infor Design System user experience files are now located in the `static/sohoxi` sub-folder. The version of jQuery used within Infor System i Workspace AnyWhere has also been upgraded and can also be found in the `static/sohoxi` sub-folder.

If you have previously upgraded your Extensions to use IDS with Infor System i Workspace AnyWhere FP13 or earlier, and referenced the original IDS CSS files in the `static/sohoxi/css` sub-folder, such as `light-theme.css` or `light-theme.min.css` (or their “dark” and “high-contrast” variants), then be advised that the CSS file names have changed at the IDS 4.56.0 release to `theme-classic-light.css` or `theme-classic-light.min.css` respectively (with “dark” and “contrast”, not “high-contrast”, variants). The names of the SVG icon files in the `static/sohoxi/svg` sub-folder have similarly changed (e.g., `svg.html` is now called `theme-classic-svg.html`).

Caution: The Infor Design System is now an open-source UI product that can be found here: https://design.infor.com/.

Infor System i Workspace AnyWhere uses a version of the Components, which are documented here (https://design.infor.com/code/ids-enterprise/latest).

The version of IDS used by Infor System i Workspace AnyWhere can be found in the `sohoxi.js` file in the `static/sohoxi` sub-folder.

Chapter 7 Re-using a System i Emulator Extension

Overview

Extensions written for the now deprecated System i Emulator should be highly compatible with the 5250 AnyWhere Emulator and require little to no modification unless they utilize older HTML/CSS standards or older Infor User Experience standards (see the previous chapter for further information and guidance on resolving these issues).

This chapter covers some common problems that may need to be resolved when porting a System i Emulator extension to the 5250 AnyWhere Emulator.

Common Problems

Use of SetFieldFormat/SetFieldFormatByIndex

Whilst these methods are available for backward compatibility with the System i Emulator, they have no effect on the 5250 AnyWhere Emulator display.

Field Formatting should be carried out using either Designer, or by using suitable JavaScript methods to format screen data as required, and then apply the data using the `SetFieldValue` or `SetFieldValueByIndex` methods (as documented in chapter 3).

Use of the m_objEmulInternal variable

Within System i Emulator extensions, the `m_objEmulInternal` variable may have been used to access methods outside of the Screen Updated and User Action Performed events that would normally be invoked via the `obj5250` object parameter.

The `m_objEmulInternal` variable is not available within the 5250 AnyWhere Emulator, but you can dynamically construct your own instance of the `obj5250` using the following code…

```
var obj5250 = new SiW5250AnywhereEmul(lastMessage, true);
```

Caution: The `lastMessage` variable should **NEVER** be altered by your Emulator Extension code.

Caution: A new instance of the `SiW5250AnywhereEmul` object should be initiated each time you wish to call 5250 AnyWhere Emulator Extension methods, and especially after any screen update has occurred.

Use of the is5250AnyWhereEmulator method

The `is5250AnyWhereEmulator` method is provided for backward compatibility but will always return a value of `true`, so any code that has been written that relies on this method returning a value of `false` will never be executed and should be removed.

Chapter 8 Replacing an Entire Screen

Overview of Screen Replacement

Screen Replacement is used when the IBM 5250 application screen needs to be customised or altered in a way that cannot be achieved by bespoke RPG development or Designer changes alone. Types of customisation may be the need to integrate data from a non-IBM i system into the display or maybe the use of user interface (UI) controls or concepts that are not available under the IBM i 5250 technology (e.g., you want a display that is not restricted by 80x24 or 132x27 columns or a fixed layout).

Whilst it is perfectly possible to use the previously discussed 5250 AnyWhere Emulator events and methods to hide the current application screen and show an alternative web-based interface in its place, we would **highly recommend** that if you wish to do this, you use the newer *Screen Replacement* features that are available in Infor System i Workspace AnyWhere. These features make it even easier to write your own extensions using standard W3C technologies, and by utilising JavaScript Object Notation (JSON) and the HTML5 postMessage API technologies, they allow safe W3C standards-based communication between your replacement screen and Infor System i Workspace AnyWhere.

Screen Replacement is initiated using Designer (see the *Infor System i Workspace AnyWhere–5250 AnyWhere Emulator Designer Guide* for more details). Using Designer, you specify a URL, along with optional parameters, to your new browser-based UI. This browser-based UI could be a HTML page, a JavaScript Service Page (JSP), servlet, or any other technology that can return a stream of data via HTTP/HTTPS that can be displayed and execute within a HTML IFRAME. When the user navigates to the application screen, and a Screen Replacement is active, Infor System i Workspace AnyWhere will automatically hide the current Emulator and display your web interface in its place. The Screen Replacement can then interrogate the IBM i program and Infor System i Workspace AnyWhere for details and drive the screen by populating fields and firing actions that. All this is done via the exchange of JSON messages, and Infor provide a JavaScript helper class to simplify this process.

Supported JSON Messages

Outgoing Messages

These are messages that you can send from your Replacement Screen to Infor System i Workspace AnyWhere. The JSON message should be encoded as a String variable and passed to the page's parent window using the HTML5 postMessage API, e.g.

```
parent.postMessage("{JSON string}", "*");
```

Request Session Data

Send the following message to Infor System i Workspace AnyWhere to request a response containing a collection of information about the current Infor System i Workspace AnyWhere installation and session (response message is described below)…

```
{
     "type": "siwControlMsg",
     "subType": "SessionData"
};
```

Request Screen Data

Send the following message to Infor System i Workspace AnyWhere to request a response containing a collection of information about the current Emulator screen (response message is described below)…

```
{
     "type": "siwControlMsg",
     "subType": "ScreenData"
};
```

Set Frame Size

Send the following message to Infor System i Workspace AnyWhere to set the size of the parent `IFRAME`...

```
{
     "type": "siwControlMsg",
     "subType": "Size",
     "data":
     {
          "width":1234,
          "height":1234
     }
};
```

The `width` and `height` values must be numeric (representing the pixel width/height of the `IFRAME`). Both values must be specified. This message will not return a reply to the caller.

Perform an Action

To set the value of a field and/or simulate a user action, send the following message to Infor System i Workspace AnyWhere;

```
{
     "type": "siwControlMsg",
     "subType": "Action",
     "data":
     {
          "command":"ENTER|F1-F24|PAGEUP|PAGEDOWN",
          "edit_fields": [{
          "index":0,
          "value":"new value"
          }]
     }
};
```

The value of the `edit_fields` property is an array of objects, each of which must have `index` and `value` properties, as shown above

The `index` property is a numeric value representing the index of the field on the original IBM i application screen. Each field within the IBM i application screen is given a unique index. The indexes are assigned to edit fields starting at the top left corner of the screen and going left to right, then down the screen.

The `value` property is the content that will be written to the specified field index. The value should be encoded as a string, even if the IBM i application field itself is numeric.

The value of the `command` property should be one of `ENTER`, `F1`, `F2`, `F3`, `F4`, `F5`, `F6`, `F7`, `F8`, `F9`, `F10`, `F11`, `F12`, `F13`, `F14`, `F15`, `F16`, `F17`, `F18`, `F19`, `F20`, `F21`, `F22`, `F23`, `F24`, `PAGEUP` or `PAGEDOWN`.

You can specify either the `command` or `edit_fields` properties, or both. The `edit_fields` property will always be processed first so that any field values are updated before the `command` is processed.

On receipt of this message, the 5250 AnyWhere Emulator will iterate through any `edit_fields`, populating the values. If any errors occur due to incorrect message content, for example an out-of-range `index` or a `value` that is too long for the displayed field, they will be ignored. It will then process the `command`, sending the action to the IBM i server.

This message will not return a reply to the caller.

Request Business Context Data

Send the following message to Infor System i Workspace AnyWhere to request a response containing the last Infor Business Context message (response message is described below)…

```
{
     "type": "siwControlMsg",
     "subType": "BusinessContext"
};
```

Incoming Messages

Adding a Message Listener to your Replacement Screen

Some of the requests above will produce a response message, again in JSON format, from Infor System i Workspace AnyWhere. If you want to receive these messages, you will need to register a message listener function, E.g.

```
window.addEventListener("message", messageHandler);
```

You will then need to define the `messageHandler` method and add code to convert the incoming message from a string to a JSON object, E.g.

```
function messageHandler(event) {
  try{
    // Try to convert the message to an object
    // (if it's not JSON, this will fail)
```

```
        var ctrlMsg = JSON.parse(event.data);
        ::
      } catch(e) {
        // Message was not JSON format, or had invalid syntax
        console.log(e.description);
      }
    }
```

The `ctrlMsg` object can then be used to extract any information. If the message came from Infor System i Workspace AnyWhere, it will have a `type` property set to `siwControlMsg` and a `subType` property that can be used to decide what to do with the incoming information (see below).

Caution: Within Infor System i Workspace AnyWhere, there are various messages that are used to pass information around between windows and sessions, some of which may not be in JSON format. You should make sure that your message listener code is robust enough to cope with, and ignore, any unexpected messages it may receive.

Session Data Response

In response to a Session Data request message (see Outgoing Messages above), the following JSON message will be sent as a reply to the caller…

```
{
     "type": "siwControlMsg",
     "subType": "SessionData",
     "data": {
          "environment":  "AUL",
          "environmentDesc": "Default Environment",
          "company":   "Z1",
          "companyDesc":  "Default Company",
          "locale":   "en_GB",
          "server":   "MYIBMI",
          "role":   "ROLECODE",
          "user":   "USERNAME",
          "siwURL":   "http://myserver/systemi",
          "staticURL":   "http://myserver/systemi/static"
     }
}
```

The property values of the `data` element are as follows…

Property	Description
`environment`	The current Environment Code
`environmentDesc`	The expanded description associated with the above Environment code

Property	Description
company	The current Company Code
companyDesc	The expanded description associated with the above Company code
locale	The current Infor System i Workspace AnyWhere locale
server	The name of the IBM i server (as defined in Workspace Configuration)
role	For a role enabled environment, the default/current Role Code
user	The IBM i user profile code
siwURL	The base URL to the Infor System i Workspace AnyWhere server
staticURL	The URL to the Infor System i Workspace AnyWhere static content

Screen Data Response

In response to a Screen Data request message (see Outgoing Messages above), the following JSON message will be sent as a reply to the caller…

```
{
     "type": "siwControlMsg",
     "subType": "ScreenData",
     "data": {
          "edit_fields": [{
          "index":0,
          "value":"field value",
          "readOnly":"true | false",
          "length":10
          }],
          "static_fields": [{
          "index":0,
          "value":"field value"
          }],
          "commands": [{
          "command":"F1-F24|ENTER",
          "value":"command text"
          }]
      }
  };
```

The property values of the `data` element are as follows…

Property	Description
edit_fields	An array of zero or more elements describing any editable fields within the current screen
static_fields	An array of zero or more elements describing any static/label fields within the current screen

Property	Description
`commands`	An array of zero or more elements describing any command options within the current screen

The property values of each `edit_fields` element array are as follows…

Property	Description
`index`	The numeric index of the edit field (see `GetFieldValueByIndex` for more information on index values)
`value`	The value of the edit field
`readOnly`	If the edit field can be altered interactively, this value will be `false`, otherwise `true`
`length`	The size of the edit field (i.e., the maximum characters that can be entered into the field)

The property values of each `static_fields` element array are as follows…

Property	Description
`index`	The numeric index of the static field (see `GetFieldValueByIndex` for more information on index values)
`value`	The value of the static field

The property values of each `commands` element array are as follows…

Property	Description
`command`	The IBM i command (`ENTER, F1-F24, PAGEUP, PAGEDOWN`)
`value`	The command description

All the `value` elements are returned as JavaScript strings, even when the field they represent is of numeric or date format.

Infor Business Context Messages

Infor Business Context Messages is the name given to the JSON structure that is produced when you use the Designer *Publish Data* option. These messages contain context-specific information about the current screen that can be consumed by a wide variety of Infor products.

In response to a Business Context Data request message (see Outgoing Messages above), the following example Infor Business Context JSON message will be sent as a reply to the caller…

```
{
     "entities": [{
          "accountingEntity": "",
          "location": "",
          "entityType": "InforSalesOrder",
          "id1": "0000100",
          "name": "",
          "description": "",
          "messageText": "",
          "drillbackURL": "?LogicalId=…",
          "readonly": true|false,
          "visible": true|false,
          "bodReference": {
               "accountingEntity": "",
               "location": "",
               "noun": "SalesOrder",
               "documentId": "0000100"
          }
     }],
     "screenId": "s21_OE02X99OE02Y01",
     "logicalId": "",
     "contextId": "14244383816539",
     "originatingTime": 1424438381653,
     "messageText": ""
}
```

The property values of the message are as follows…

Property	Description
`entities`	An array of one or more entity definitions, described below
`screenId`	The ID of the screen from where the data was published
`logicalId`	The ID of this application when running inside Infor Ming.le™
`contextId`	Unique message ID
`originatingTime`	Time, in epoch milliseconds, when the message was issued
`messageText`	Message data that is used when using the Share functionality within Infor Ming.le™

The property values of the `entities` element array are as follows…

Property	Description
`accountingEntity`	The accounting entity of the entity instance. This is an optional field and depends on a specific entity instance.

Property	Description
location	The location of the entity instance. This is an optional field and depends on a specific entity instance.
entityType	The type of the business entity, like inforPurchaseOrder, inforPurchaseOrderLine, inforCustomerPartyMaster, etc. This is a mandatory field.
id1 to id15	Depending on the business entity type, the id fields (e.g.: id1, id2, etc...) can be from 1 to 15. id1 is mandatory.
name	Name of the instance of the entity. This field is optional.
description	Description for the instance of the entity. This field is optional.
messageText	A message to summarize the current activity the user is performing. This field is optional.
drillbackURL	The drillback URL for the entity. This is a relative URL with just query parameters starting with question mark character. This field is optional.
readonly	A boolean flag to indicate the status of the entity. This field is optional.
visible	A boolean flag to indicate if the entity needs to be displayed in the contextual applications. This field is optional.
bodReference	A reference to the Infor Business Object Document noun instance

The property values of the bodReference element array are as follows…

Property	Description
accountingEntity	The accounting entity of the noun instance. This is an optional field and depends on a specific noun instance.
location	The location of the noun instance. This is an optional field and depends on a specific noun instance.
noun	The type of the noun, like ItemMaster, PurchaseOrder, etc. This is a mandatory field for every existing instance of the bodReference object.
documentId	The documentId of the noun instance. This is a mandatory field for every existing instance of the bodReference object.

The Screen Replacement Helper Class

The Screen Replacement Helper class is a JavaScript object that can be initiated and used to post a message in the correct syntax to the Emulator page.

Initializing the Screen Replacement Helper Class

To use the Screen replacement Helper, you must include the `siw-replacement-screen-functions.js` in your page. This is shipped in the `static\customScreens\shared\js` sub-folder of your Infor System i Workspace AnyWhere installation.

To create an instance of the Screen Replacement Helper class, use the following JavaScript code…

```
var m_obj_helper = new siwReplacementScreenHelper();
```

Screen Replacement Helper Methods

The Screen replacement Helper provides the following methods for interacting with the IBM i program.

loadJSON

```
m_obj_helper.loadJSON(url, func);
```

Call a web-based service that returns JSON data as a result. The JSON object is passed to the supplied function.

The `url` parameter should be the address of the web service that will return a JSON string. The `loadJSON` function will convert the string to a JSON object. If it cannot convert it to an object, it will pass the string result through to the supplied JavaScript function.

The `func` parameter should be the JavaScript function that receives the JSON result from the web-service; e.g.

```
var func = function() {
  var oJSON = arguments[0];
}
m_obj_helper.loadJSON("./service-url", func);
```

The `func` parameter may be null if the web-service does not return a result or you do not require the result data in your Screen Replacement code.

requestScreenData

```
m_obj_helper.requestScreenData();
```

Sends a request for Screen Data. Your message listener will receive a Screen Data Response.

requestSessionData

```
m_obj_helper.requestSessionData();
```

Sends a request for Session Data. Your message listener will receive a Session Data Response.

requestIBC

```
m_obj_helper.requestIBC();
```

Sends a request for the latest Business Context Data message. Your message listener will receive an Infor Business Context Data Response.

sendActionMessage

```
m_obj_helper.sendActionMessage(command, editFields);
```

Send an action and/or a set of field updates to the IBM i program.

The value of the `command` property should be one of `ENTER`, `F1`, `F2`, `F3`, `F4`, `F5`, `F6`, `F7`, `F8`, `F9`, `F10`, `F11`, `F12`, `F13`, `F14`, `F15`, `F16`, `F17`, `F18`, `F19`, `F20`, `F21`, `F22`, `F23`, `F24`, `PAGEUP` or `PAGEDOWN`.

The value of the `editFields` property should contain a JavaScript array of objects, each of which must have `index` and `value` elements (see *Perform an Action* in the *Outgoing Messages* section above).

Either `command` or `editFields` may be set to `null`;

setFrameSize

```
m_obj_helper.setFrameSize(width, height);
```

Set the size of the HTML IFRAME containing the replacement screen.

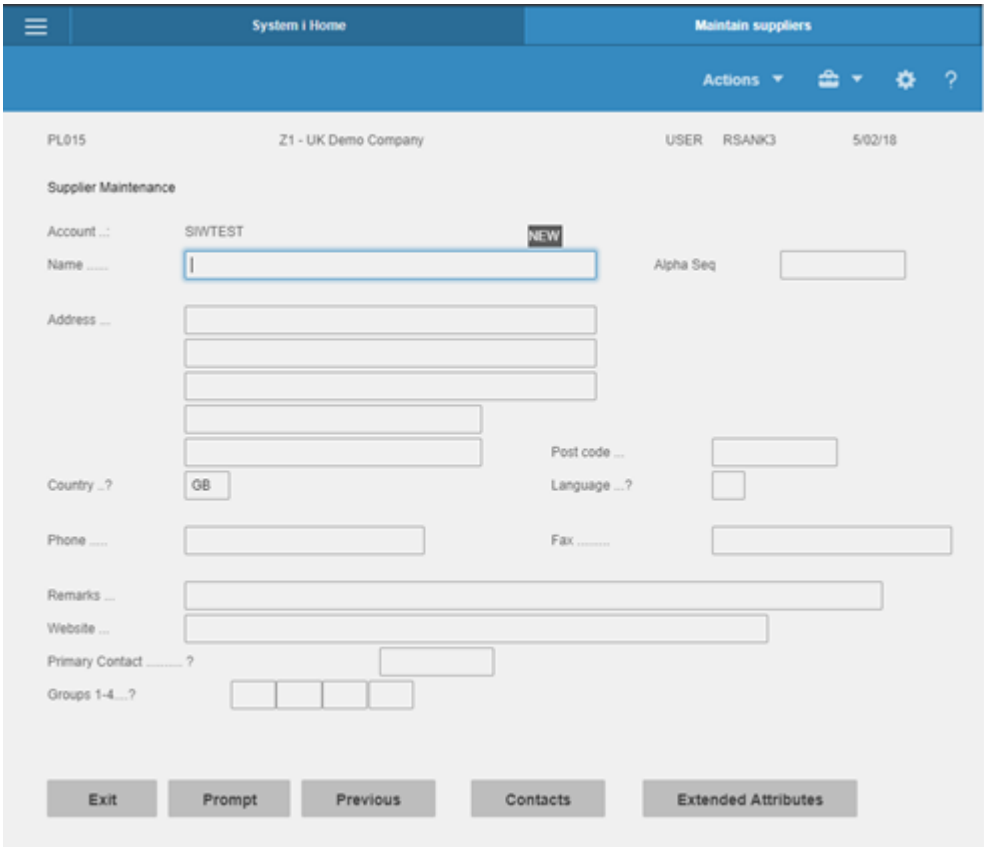
The `width` and `height` values must be numeric (representing the pixel width/height of the IFRAME). Both values must be specified.

Chapter 9 Examples

Adding client-side validation

In this example we will analyse the data entered within a task by the user when they perform an action and, if there is a problem, display an error message in the `div2` HTML object within the screen.

We will use the Supplier Maintenance task (1/APM). After entering a new supplier, you are presented with a screen (with a Screen ID/Magic Number of **PL00R99**) containing supplier name, address etc.; for example…



Normally, only the supplier *Name* and *Alpha Seq* fields are mandatory but let's extend this so that the user must enter numbers into the *Phone* and *Fax* fields too, and do not allow this screen to progress until they do.

We will perform the validation in the User Action Performed event when the Enter key (action code 42) is pressed. From the screen above, we can calculate that the index of the *Phone* edit field is 10 and the index of the *Fax* edit field is 11. We will assume that if these fields do not follow the format nnnnn nnnnnn (where n is any digit) or are blank that the number is invalid. If either number is invalid, we use the `setDiv2` method to display a message in the `div2` HTML object.

Here is the code required…

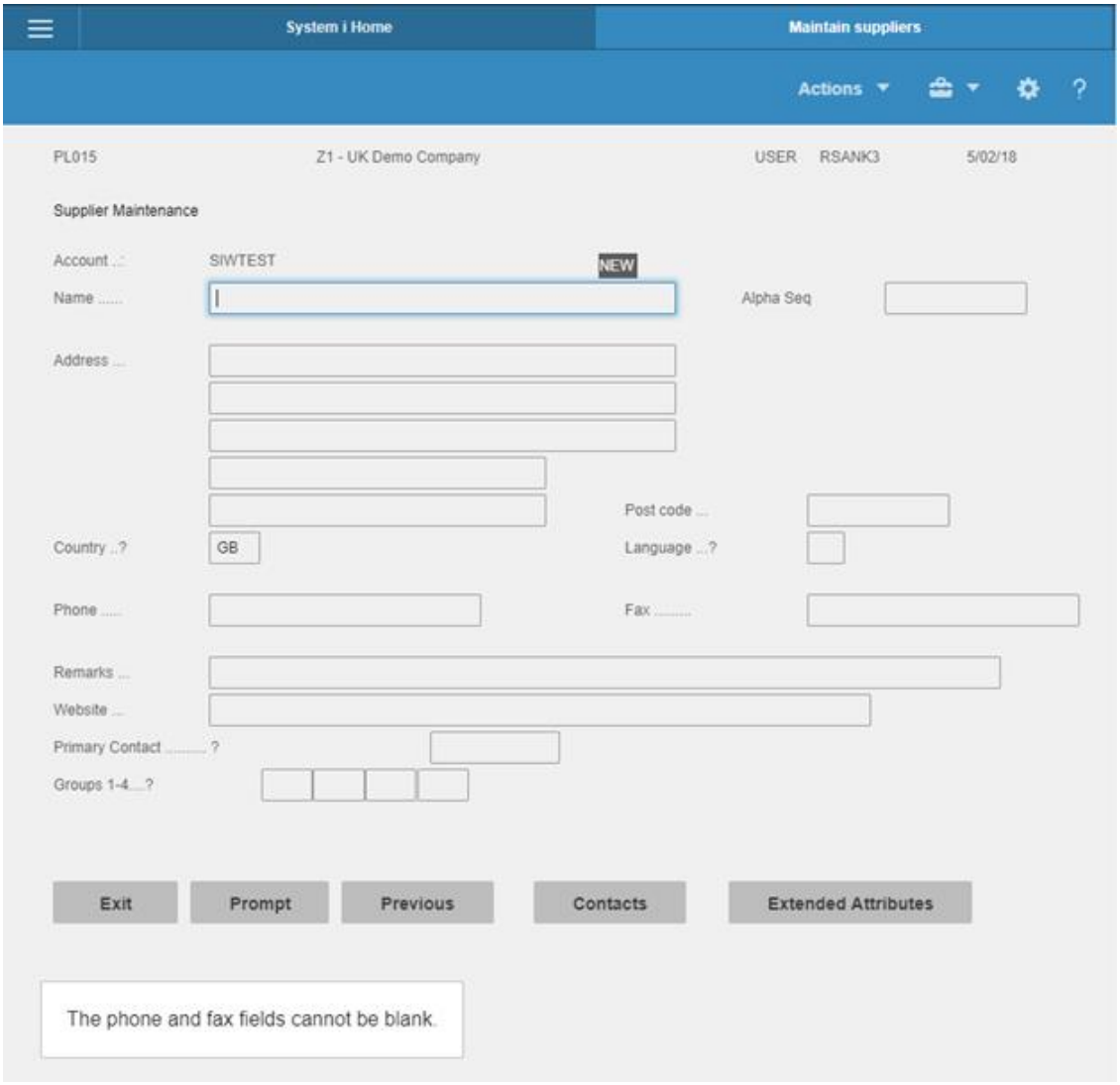
```
function CustomActionPerformed(strScreenId, intAction, obj5250) {
   var result = 0;
   switch(strScreenId) {
   // Process Supplier Maintenance screen 1/APM
   case "PL00R99":
      // Clear the div2 area
      clearDiv2();
      showDiv2(true);
      // Did the user press Enter?
      if (intAction == 42) {
        var msg = "";
        var phone = obj5250.GetFieldValueByIndex(10, 1);
        var fax = obj5250.GetFieldValueByIndex(11, 1);
      // Check the phone/fax values are non-blank and
      // contain only numbers
        if (phone == "" || fax == "") {
            msg = "<h3>The phone and fax fields cannot be blank.</h3>";
            result = 1;
        } else if (!validNumber(phone)) {
            msg = "<h3>The phone number is not a valid format (99999
999999).</h3>";
            result = 1;
        } else if (!validNumber(fax)) {
            msg = "<h3>The fax number is not a valid format (99999
999999).</h3>";
            result = 1;
        }
        // Update the display
        setDiv2(msg);
        break;
      }
   default:
      clearDiv2();
      showDiv2(false);
      break;
  }
   return result;
}
// A valid phone number is an area-code of 5 numbers followed by a space
```

```
// then 6 numbers
function validNumber(number) {
  var objRegExp  = /\d{5}\s\d{6}/;
  return objRegExp.test(number);
}
```

Caution: Some of the message lines above may have been wrapped. Make sure these are restored to being a single line when you apply this to your server.

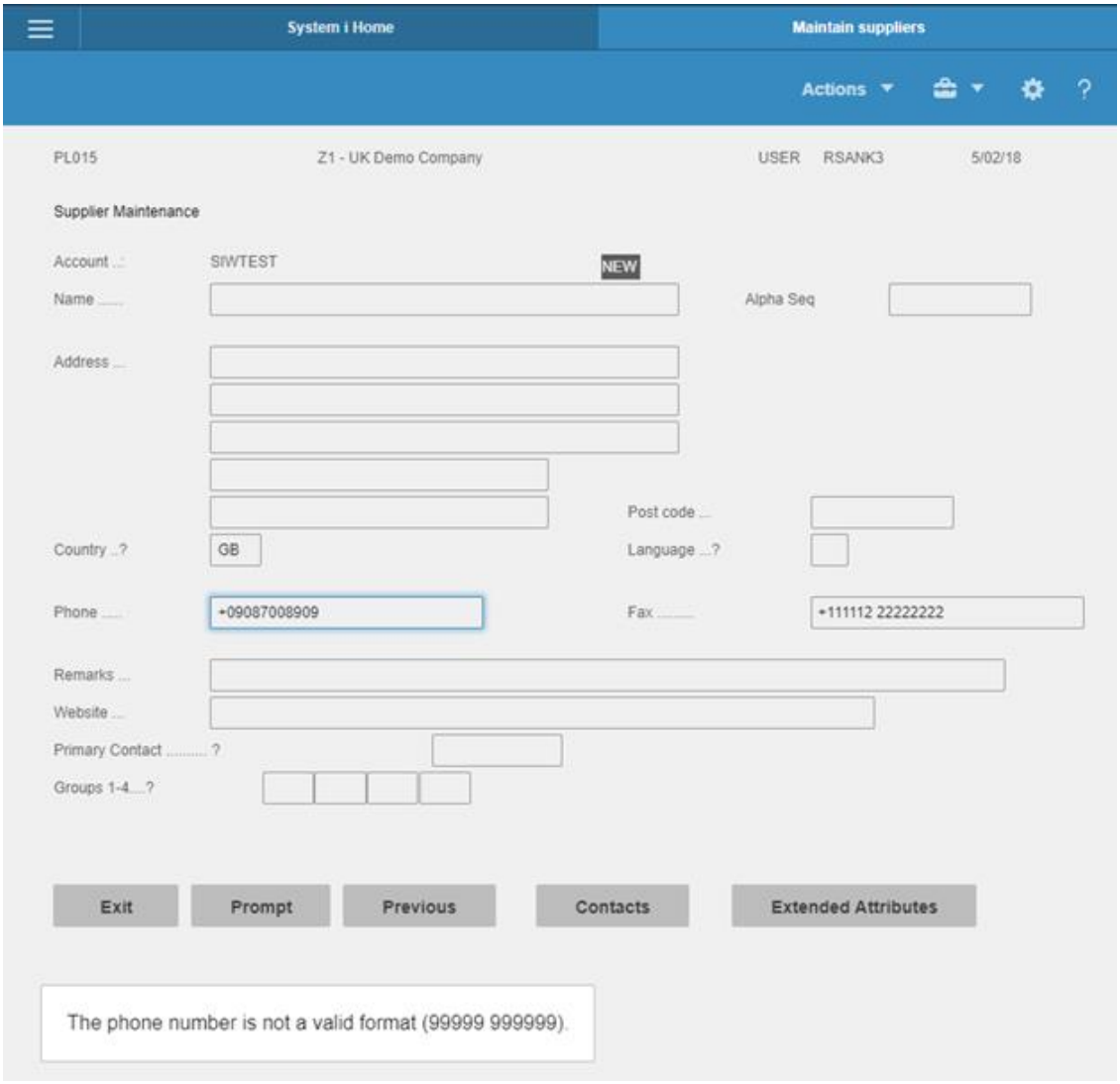
Once you have applied this code to the `5250-extensions.js` file, restart your Infor System i Workspace AnyWhere server.

At run-time, within the Supplier Maintenance screen, if the user does not enter a *Phone* or *Fax* number, they will see a message like this when using the 5250 AnyWhere Emulator…



The next screen cannot be reached until the user enters valid numbers.

If the user enters an invalid *Phone* or *Fax* number, they will see a message like this when using the 5250 AnyWhere Emulator …



The next screen cannot be reached until the user enters valid numbers.

Pre-filling screen values

In this example, we will assume that each Infor System i Workspace AnyWhere user is assigned to look after a different supplier. To save them having to remember the supplier account code when they use the Supplier Maintenance or Supplier Enquiry Infor ERP System21 applications, we will use the current user profile variable to look-up their assigned supplier and populate the form with this value.

The first screen of Supplier Maintenance task (1/APM) has a Magic Number of **PL00Q99,** and the first screen of the Supplier Enquiry task (1/APE) has a Magic Number of **PL0EC99**. On each screen, the *Supplier* edit field is at index 0.

We will use the Screen Updated event to perform our update. We will only perform this update on the first entry to this screen from when the task is launched (i.e., the previous screen will be the Workspace Entry Point Panel application which has a Magic Number of **AM18099**).

Here is the code required…

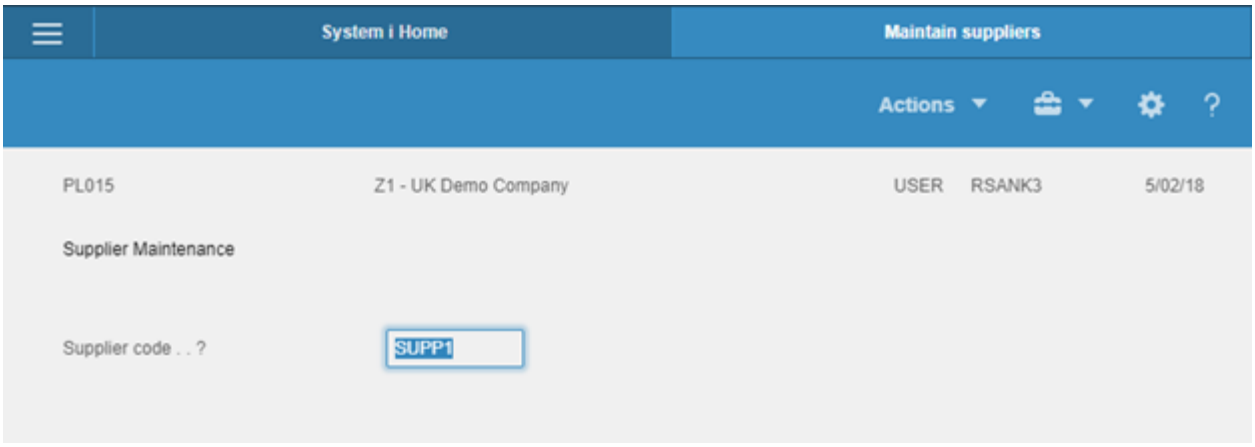
```
function CustomScreenChanged(strScreenId, strLastScreenId,
        blnPopup, obj5250) {
  switch(strScreenId) {
   // Populate the 1st edit field of 1/APE or 1/APM with
   // the default supplier for this user
   case "PL00Q99":
   case "PL0EC99":
      if (strLastScreenId == "AM18099") {
      obj5250.SetFieldValueByIndex(0, 1, getSupplierForUser());
      }
      break;
   default:
      break;
  }
}
function getSupplierForUser() {
   var supplier = "";
   switch (g_user) {
   case "USER1":
      supplier = "SUPP1";
      break;
   case "USER2":
      supplier = "SUPP2";
      break;
   case "USER3":
      supplier = "SUPP3";
      break;
   default:
      break;
  }
  return supplier;
}
```

Caution: Some of the message lines above may have been wrapped. Make sure these are restored to being a single line when you apply this to your server.

You will need to edit the `getSupplierForUser` method to provide valid supplier and user profile codes for your system.

Once you have applied this code to the `5250-extensions.js` file, restart your Infor System i Workspace AnyWhere server.

At run-time, with this code in place, when the user enters the Supplier Enquiry/Maintenance task, they will find that the field is populated with their preferred supplier. They can still over-type the *Supplier* edit field if they wish to select another; for example, when using the 5250 AnyWhere Emulator …



Auto-navigating screens

Let us extend the previous example to show how to auto-navigate through a screen by sending action keys to the IBM i application.

In the Supplier Maintenance task, if we wanted to prevent the user from changing any other supplier data, except the one they are assigned to, then we could alter the code to auto-navigate through the first screen (pre-fill the supplier based on the user and then press **Enter**) and, if the user navigates back to that screen, automatically end the application by firing the **F3** command.

Here is the code required (keep `getSupplierForUser` method from previous example)…

```
function CustomScreenChanged(strScreenId, strLastScreenId, blnPopup,
        obj5250) {
  switch(strScreenId) {
   // On entry to the application, populate the 1st
   // edit field of 1/APM with the default supplier
   // for this user and navigate to the next screen.
   // On exit, press F3 to end the application
   case "PL00Q99":
      if (strLastScreenId == "AM18099") {
      obj5250.SetFieldValueByIndex(0, 1, getSupplierForUser());
```

```
            obj5250.SendKey(42);
            } else
            obj5250.SendKey(45);
            break;
            // Populate the 1st edit field of 1/APE with
        // the default supplier for this user
        case "PL0EC99":
            if (strLastScreenId == "AM18099") {
            obj5250.SetFieldValueByIndex(0, 1, getSupplierForUser());
            }
            break;
        default:
            break;
        }
    }
```

Caution: Some of the message lines above may have been wrapped. Make sure these are restored to being a single line when you apply this to your server.

The `getSupplierForUser` method is the same as in the previous example.

Once you have applied this code to the `5250-extensions.js` file, restart your Infor System i Workspace AnyWhere server.

On starting the Supplier Maintenance task, the user profile is examined, and the appropriate supplier code is returned. Enter will be automatically pressed to take the user to the Supplier Maintenance details. If they select F12:Previous on this screen then the task will exit.

Running server-side code

In the previous two examples, we have used a client-side method to determine the appropriate supplier for a given user. However, in reality, this type of information would be either held in a lookup table on the Infor System i Workspace AnyWhere server or within a file on the IBM i server. In either scenario, the client code needs to make a call back to the server to retrieve the value.

To call back to the Infor System i Workspace AnyWhere server (or indeed any other web server), we will use the XMLHTTP object.

Using a Server-side Mapping File

First, let's create a mapping XML file on the server. Name the file `mapping.xml` and place this in the `static\customScripts` folder of your Workspace installation. Set the content of this file to the following XML…

```
<?xml version="1.0" encoding="UTF-8"?>
<users>
      <user name="USER1" supplier="SUPP1"/>
      <user name="USER2" supplier="SUPP2"/>
      <user name="USER3" supplier="SUPP3"/>
</users>
```

You will need to edit the values to provide valid supplier and user profile codes for your system.

Create a processing file on the server. Name the file `get-supplier-for-user.xsp` and place this in the `static\customScripts` folder of your Workspace installation. Set the content of this file to the following XSL…

```
<xsl:stylesheet version="1.0"
     xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:param name="user"/>
<xsl:output method="text"/>
<xsl:variable name="mapping" select="document('mapping.xml')"/>
<xsl:template match="/">
<xsl:value-of select="$mapping/users/user[@name=$user]/ @supplier"/>
</xsl:template>
</xsl:stylesheet>
```

We have used XSL in this example as this technology is used throughout the Infor System i Workspace AnyWhere product and there are various extensions built for use in XSL, but you could also use Java Server Pages (JSP) if that technology is more familiar to you.

The XSP extension is used to signify that this is an XSL file that is transformed by the Infor System i Workspace AnyWhere server.

The XSL style sheet takes a single parameter, `user`. It loads the `mapping.xml` file into a variable, `mapping`, and then on the default template, it extracts the supplier's name from the mapping XML using the `user` parameter via an XPath query. The supplier's name is then returned as the result of the transformation.

We now need to write the client-side code to call this page. Edit the `5250-extensions.js` file on your Infor System i Workspace AnyWhere server and change the `getSupplierForUser` method to the following…

```
function getSupplierForUser() {
  var url = STATIC_URL + "/customScripts/
          get-supplier-for-user.xsp?user=" + g_user;
  var objRequest = new XMLHttpRequest();
```

```
    objRequest.open("GET", url, false);
    objRequest.send();
    return objRequest.responseText;
}
```

Caution: Some of the message lines above have been wrapped. Make sure these are restored to being a single line when you apply this to your server.

The first line of the method constructs a URL to the `get-supplier-for-user.xsp` page. We use the `STATIC_URL` global variable so we don't have to hard-code the Infor System i Workspace AnyWhere server name. The `g_user` global variable that contains the user profile is then appended as a parameter.

On line two, we create an instance of the standard XML HTTP Request object.

On line three, we setup the connection to the `get-supplier-for-user.xsp` page using the HTTP GET method, passing the URL constructed on line one, and pass `false` so the connection is synchronous.

Line four will connect to the server, perform the XSL transformation, and return the result to the XML HTTP Request object.

Finally, we return the content of the `responseText` property as the result.

Once you have applied this code, restart your Infor System i Workspace AnyWhere server.

On starting the Supplier Maintenance task, the user profile is examined on the server and the appropriate supplier code is returned. Enter will be automatically pressed to take the user to the Supplier Maintenance details. If they select F12:Previous on this screen then the task will exit.

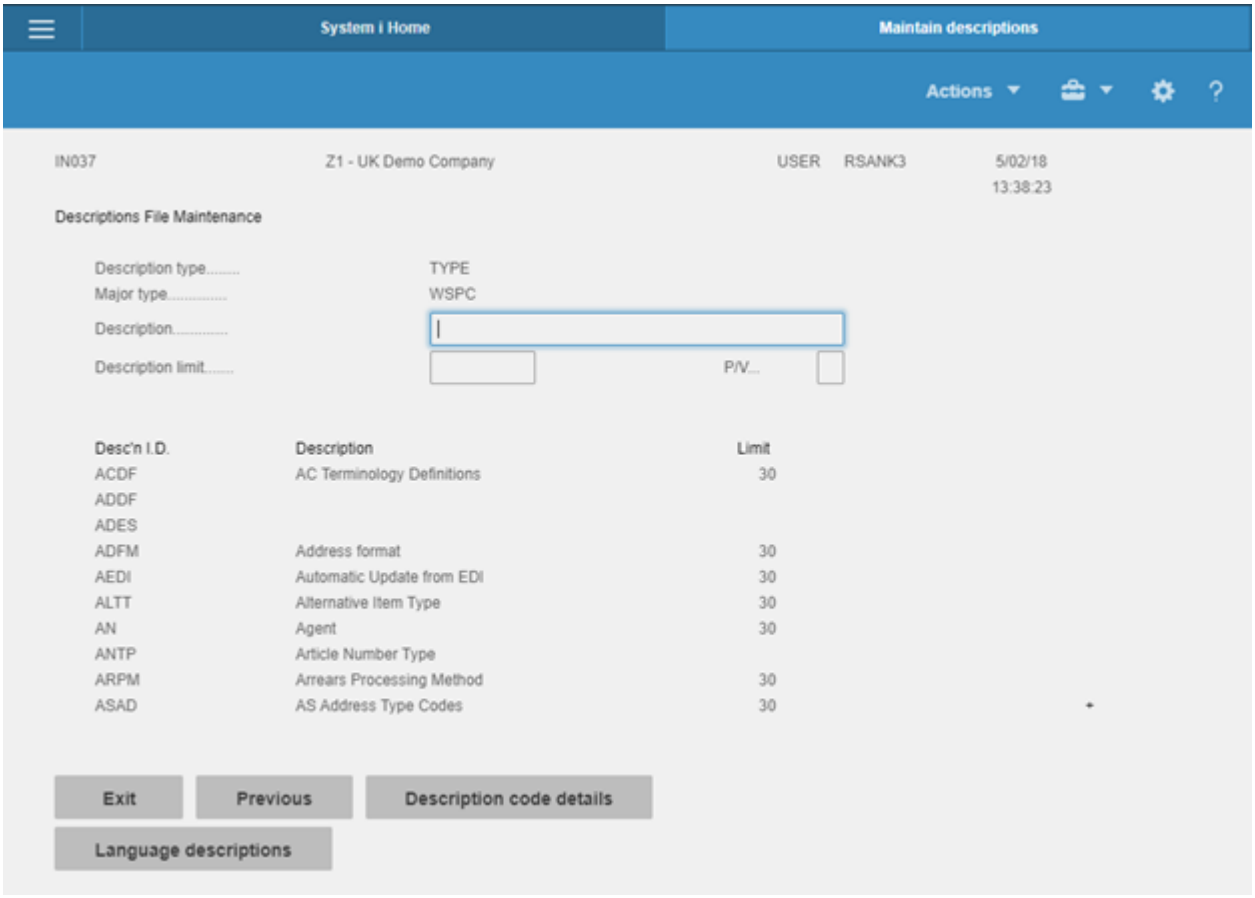
Using an IBM i Database

Let's us now look at dynamically retrieving the supplier from an IBM i application database.

We will use the Inventory Descriptions Maintenance task (1/INM) to store the user to supplier mappings.

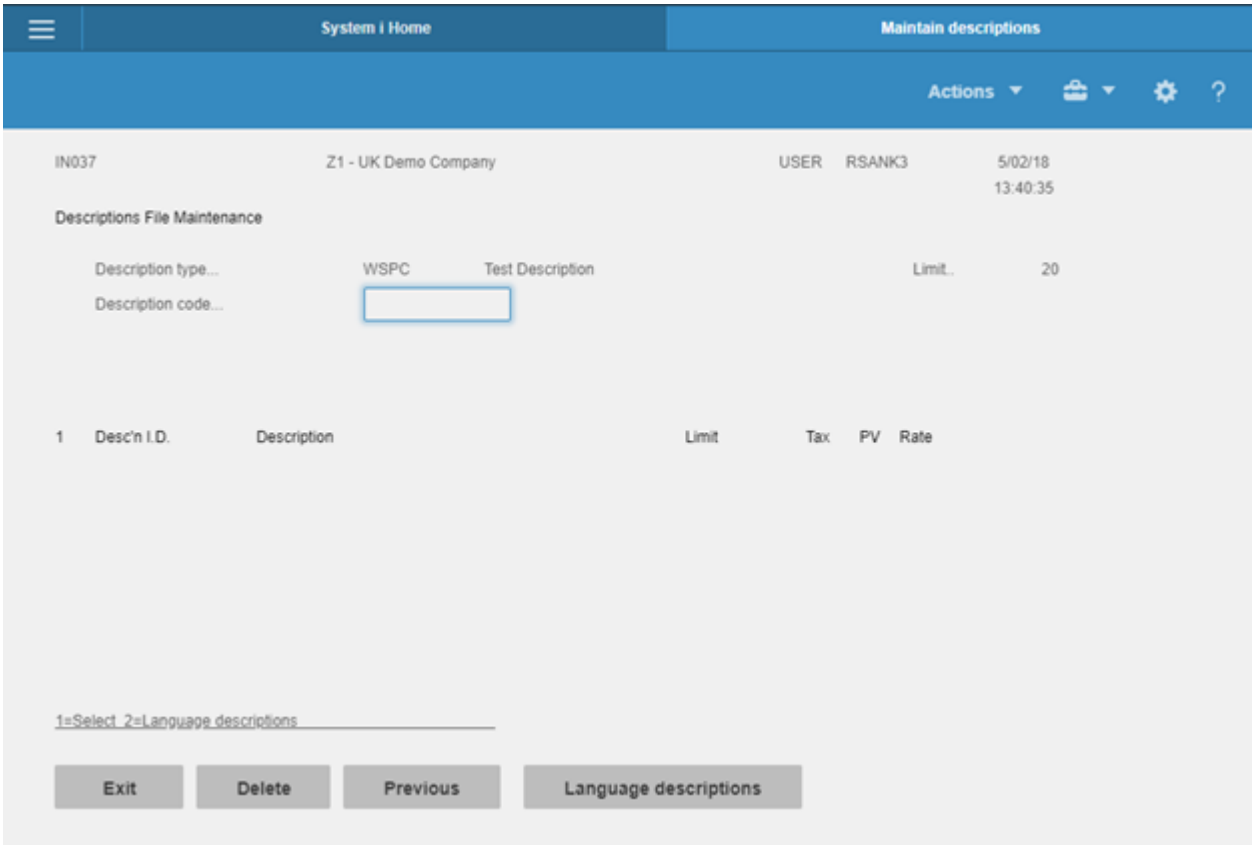
This information for this application is written to the `DESC` table in the library `AULT2F3` in your IBM i database.

Start Inventory Descriptions Maintenance and enter a *Major type* of **WSPC** and press **Enter**…



Enter a *Description* and a *Description limit* then press **F15** to enter the *Description code details* screen.

For each user profile you wish to map, enter a *Description code* and press **Enter**, then enter the supplier code in the *Description* field…



Press **Enter** to assign the mapping. You can add as many mappings as you require. Press **F3** to end the task.

Create a new processing file on the server. Name the file `get-supplier-for-user-sql.xsp` and place this in the `static\customScripts` folder of your Workspace installation. Set the content of this file to the following XSL…

```
<xsl:stylesheet version="1.0"
      exclude-result-prefixes="sql"
      xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
      xmlns:sql="com.geac.xtrane.extensions.SQLQuery">
<xsl:import href="../../server/xsl/logon-validate.xsl"/>
<xsl:param name="user"/>
<xsl:output method="text"/>
<xsl:variable name="t2lib" select="'AULT2F3'"/>
<xsl:template name="logon-validate-complete">
   <xsl:param name="current-profile"/>
   <xsl:param name="session-company"/>
   <xsl:variable name="sql">
      SELECT PRMD15 FROM <xsl:value-of select="$t2lib"/>/DESC
      WHERE CONO15='<xsl:value-of select="$session-company"/>' AND
      PRMT15= ? AND PSAR15=?
```

```
    </xsl:variable>
    < xsl:variable name="sql-params">
       WSPC, <xsl:value-of select="$user"/>
    </xsl:variable>
    <xsl:variable name="sql-pool" select="string($current-
profile/sql_pool)"/>
    <xsl:variable name="sqlRes" select="sql:sqlPreparedQuery(string($sql-
pool), normalize-space($sql), normalize-space($sql-params), 1)"/>
    <xsl:value-of
select="$sqlRes/resultset/row/column[@name='PRMD15']/@value"/>
</xsl:template>
</xsl:stylesheet>
```

Caution: Some of the message lines above may have been wrapped. Make sure these are restored to being a single line when you apply this to your server.

This XSL style sheet uses the standard Infor System i Workspace AnyWhere session login processing to pick up the various settings and configuration properties for the server and user. This processing is contained within the imported `logon-validate.xsl` file.

Do not attempt to modify any of the standard Workspace files.

In the `logon-validate-complete` XSL template, we dynamically create the SQL statement to query the `DESC` table in `AULT2F3`, including the `user` parameter as part of the query.

Depending on your ERP application setup, you may have different files libraries dependant on your environment (customer) code. Use the `t2lib` variable to set the correct library containing the `DESC` table.

The style sheet uses the `sqlPreparedQuery` XSL extension function to perform the query against an IBM i database, using the correct database pool (which you will have defined using Workspace Configuration when you installed Infor System i Workspace AnyWhere).

The `sqlPreparedQuery` XSL extension function takes four parameters…

Parameter	Description
`pool`	An Infor System i Workspace AnyWhere database pool defined within the Database Connections tab in Workspace Configuration.
`sql`	The SQL statement you wish to execute.
`sql-params`	Comma separated list of SQL parameter values for the statement
`records`	The number of records you wish to return. Use -1 to return all matching records

The function returns a node-set containing the results of the query, or error information if the query failed.

Finally, in our server-side extension code, we extract and return the supplier from the XML node-set returned from the query back to the client.

We now need to update the client-side code to call this page. Edit the `5250-extensions.js` file on your Infor System i Workspace AnyWhere server and change the `url` variable in the `getSupplierForUser` method to call the `get-supplier-for-user-sql.xsp` page.

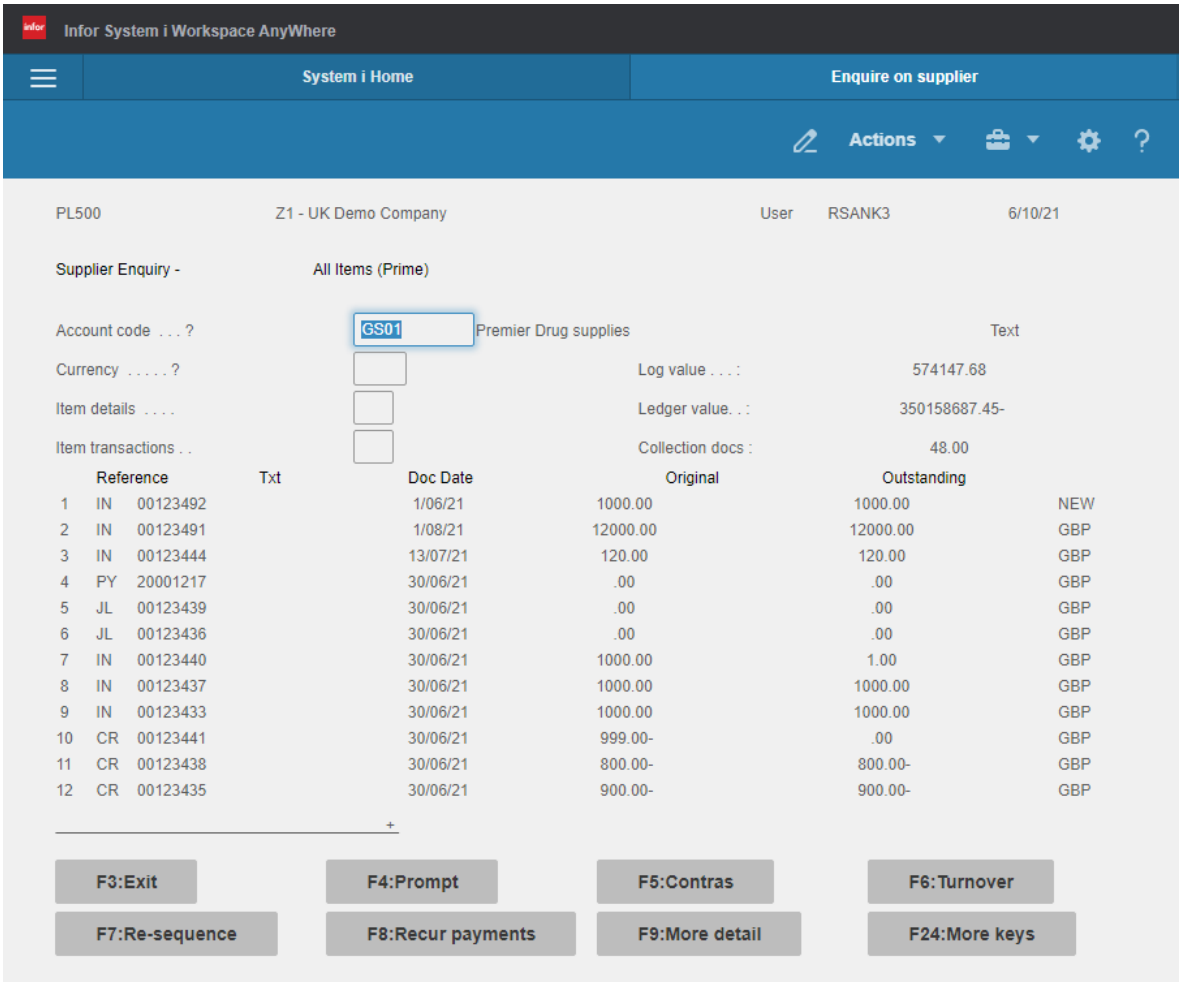
Once you have applied this code, restart your Infor System i Workspace AnyWhere server.

On starting the Supplier Maintenance task, the user profile is examined within the Infor ERP System21 database and the appropriate supplier code is returned. Enter will be automatically pressed to take the user to the Supplier Maintenance details. If they select F12:Previous on this screen then the task will exit.

Export an entire table to the clipboard

For this example, we will display a button within the `div2` HTML object that can be used to perform an action that drives through the current IBM i screen to retrieve all data from a table and store it within the Windows Clipboard for use in other applications.

We will use the *Supplier Enquiry - All Items* screen (Magic Number **PL0E702**) within the Enquire on Supplier application (1/APE). You can navigate to this screen by entering a valid supplier, pressing *Enter* and then pressing *F14:All Items* then pressing *F9:Less Detail*, for example…



We have selected *F9:Less Detail* to show more records on the screen.

Notice the plus sign at the bottom of the table. When there are no more pages of data, this field will not appear. We can use this to auto-page down through all the data, capturing the content into our buffer as we go.

We will use the Screen Changed event to capture the data off the screen. As the data is tabular with a fixed starting row and a fixed number of columns/rows, we can use a loop to read in the values.

Here is the code required to capture the screen data…

```
var m_mode = "";
var m_buffer = "";
var m_cols = new Array(1, 4, 7, 17, 27, 38, 57, 75);

function CustomScreenChanged(strScreenId, strLastScreenId, blnPopup, obj5250) {
  switch(strScreenId) {
    case "PL0E702":
      if (m_mode == "") {
        setDiv2("<button onclick=\"startCapture()\">Export Table To
Clipboard</button>");
```

```
        showDiv2(true);
      } else if (m_mode == "paging") {
        captureTable(obj5250);
      }
      break;

    default:
      showDiv2(false);
      break;
  }
}

function startCapture() {
  m_mode = "paging";
  m_buffer = "";
  captureTable();
}

function captureTable(obj5250) {
  if (obj5250 == undefined) obj5250 = new SiW5250AnywhereEmul(lastMessage, true);
  // Add data onto the buffer
  for (i=9; i<21; i++) {
    for (j=0; j<8; j++) {
      m_buffer = m_buffer + obj5250.GetFieldValue(i,  m_cols[j]);
      if (j<7) m_buffer = m_buffer + "\t";
    }
    m_buffer = m_buffer + "\n";
  }
  // Check for more data
  var strPlus = obj5250.GetFieldValue(21,1)
  if (strPlus.trim() == ""){
    m_mode = "";
    showDiv2(false);
    window.console.log(m_buffer);
    // Copy data to the clipboard using 5250 AW Emulator method
    updateClipboard(m_buffer);
  } else {
    // Move to next page
    obj5250.SendKey(74);
  }
}
```

Caution: Some of the message lines above have been wrapped. Make sure these are restored to being a single line when you apply this script to your Infor System i Workspace AnyWhere server.

We use the global variable `m_mode` to determine whether we are in the process of paging or not.

We use the variable `m_buffer` to hold the screen data.

We use the array variable `m_cols` to hardcode a list of the columns we wish to read from, this simplifies the loop that retrieves the data.

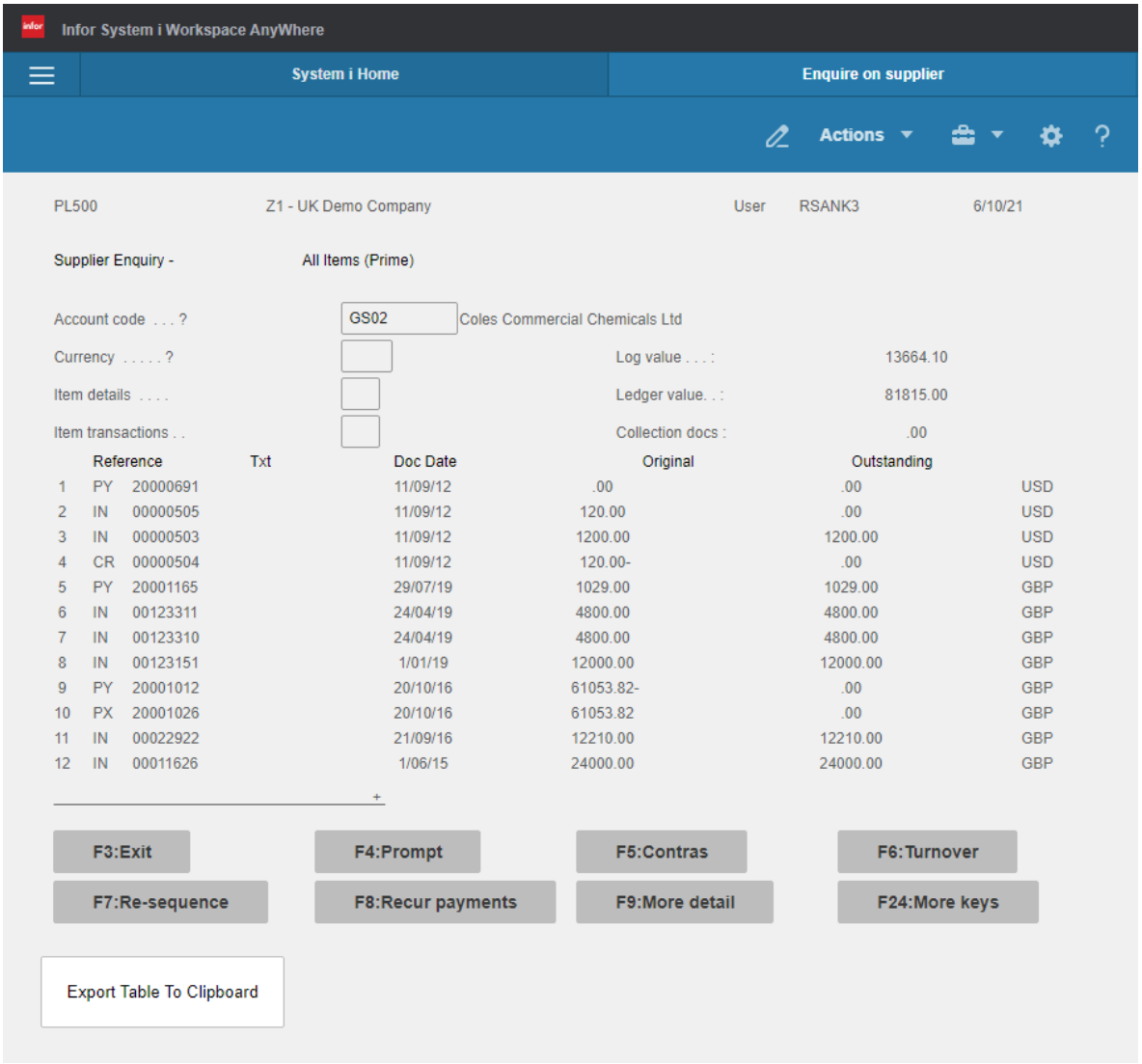
In the Screen Changed event, we check for the **PL0E702** screen and, when the screen is shown we check the `m_mode` variable.

When there is no value set, we add one element to the `div2` HTML object, a button that, when selected, will start the copy process by calling the `startCapture` method.

The `startCapture` method simply clears the `m_buffer` variable, sets the `m_mode` to paging and calls the `captureTable` method.

The `captureTable` method uses a loop to append the data from the screen to the buffer, adding a tab character between each field and a new line character at the end of row. It then checks to see if the plus-sign is available on row 21. If it is, we auto-press the page down key (action code **74**). If there is no plus-sign, we clear the `m_mode`, copy the data to the Windows Clipboard and remove the export button from the screen.

At runtime, the user will see the button underneath the main screen, for example…



When the *Export Table to Clipboard* button is pressed, the paging process will begin. The screen may flicker during this process as it rapidly moves through each page of data (the amount of flicker will depend on the speed and load of your IBM i server).

Once the process is complete, the *Export Table to Clipboard* button will be removed and the Windows Clipboard will contain the data (this is also written to the web browser's developer tools console for you to review).

The data can now be pasted into an application, such as Microsoft Excel.

Caution: The web browser may prevent the copy to clipboard if the data buffer is overly large (for example 10 pages or more). In the subsequent example, we will show how this can be avoided by exporting to a CSV file.

Avoiding Screen Flicker

To avoid the screen "flickering" as the script processes through the data, we can use the Control Screen Visibility event to hide the screen during paging mode, for example:

```
var m_mode = "";
var m_buffer = "";
var m_cols = new Array(1, 4, 7, 17, 27, 38, 57, 75);

function CustomScreenChanged(strScreenId, strLastScreenId, blnPopup, obj5250) {
  switch(strScreenId) {
    case "PL0E702":
      if (m_mode == "") {
        setDiv2("<button onclick=\"startCapture()\">Export Table To
Clipboard</button>");
        showDiv2(true);
      } else if (m_mode == "paging") {
        setDiv2("<p>Processing...</p>");
        showDiv2(true);
        captureTable(obj5250);
      }
      break;

    default:
      showDiv2(false);
      break;
  }
}

function startCapture() {
  m_mode = "paging";
  m_buffer = "";
  captureTable();
}
```

```
function captureTable(obj5250) {
  if (obj5250 == undefined) obj5250 = new SiW5250AnywhereEmul(lastMessage, true);
  // Add data onto the buffer
  for (i=9; i<21; i++) {
    for (j=0; j<8; j++) {
      m_buffer = m_buffer + obj5250.GetFieldValue(i,  m_cols[j]);
      if (j<7) m_buffer = m_buffer + "\t";
    }
    m_buffer = m_buffer + "\n";
  }
  // Check for more data
  var strPlus = obj5250.GetFieldValue(21,1)
  if (strPlus.trim() == ""){
    m_mode = "";
    showDiv2(false);
    window.console.log(m_buffer);
    // Save data to a CSV file using 5250 AW Emulator method
    saveExportFile('application/csv', m_buffer, "supplier-items" + ".csv");
  } else {
    // Move to next page
    obj5250.SendKey(74);
  }
}

function CustomHideEmul(strScreenId, strLastScreenId) {
  var result = 0;
  switch(strScreenId) {
    case "PL0E702":
      if (m_mode == "paging") result = 1;
      break;
    // Show all other screens
    default:
      break;
  }
  return result;
}
```

Caution: Some of the message lines above have been wrapped. Make sure these are restored to being a single line when you apply this to your Infor System i Workspace AnyWhere server.

When the mode is **paging**, we hide the 5250 AnyWhere Emulator screen and show a “Processing” message in `div2`. When complete, the data captured will be written to a CSV file, which will be saved, by default, in your “downloads” folder. The CSV file can be opened using a suitable application, such as Microsoft Excel.

Replacing a Prompt Screen with a Table

In this example, we will use the Designer Replacement Screen functionality to hide the initial screen of the Infor ERP System21 Credit Manager Enquiry (1/ARE) application and replace it with a table control. Normally, the user must use the Prompt control to be able to view a list of customer accounts, in this example, the list will be displayed by default. The Replacement Screen will need to provide the following functionality…

1. Retrieve all account codes, descriptions, and other information from the Infor ERP System21 database
2. Display the data in a table that can be searched, sorted
3. Allow the user to select a row within the table and navigate to the Account Details screen
4. Allow the user to Exit the screen

The example is made up of three parts; a main HTML file containing the user-interface, a JavaScript file, and a server-side XSL processing file (XSP) that can retrieve the Infor ERP System21 data and return it as a JSON message.

To start, locate the `static\customScreens\examples\list-of-customer-accounts` sub-folder of your Infor System i Workspace AnyWhere installation. This contains the fully commented sample code for this example.

Within the `list-of-customer-accounts.html` file, you will find the following `body` section (shown with additional discussion point indicators in blue)…

```
<body class="full tight">
<div class="svg-icons">
          <!-- Icon defintions -->
</div>
     <div class="page-container">
<div class="row">
<div class="twelve columns">
<div class="field">
1<div id="inforDataGrid"></div>
</div>
</div>
</div>
<div class="row">
<div class="twelve columns">
<div class="field">
2<button type="button" onclick="m_obj_helper.sendActionMessage('F3');"
class="btn-primary">Exit</button>
</div>
</div>
</div>
     </div>
</body>
```

Point 1: This `DIV` element will hold the table. Initially, it has no content.

Point 2: Add an Exit button to the bottom of the Replacement Screen. When the user presses this button, it will use the Screen Replacement Helper class to send the `F3` command to the IBM i server.

Within the `list-of-customer-accounts.js` file, you will find the following JQuery `ready` method (shown with additional discussion point indicators in blue)…

```
$().ready(function () {
1m_obj_helper = new siwReplacementScreenHelper();
2m_obj_helper.setFrameSize(1124, 400);
3m_obj_helper.requestSessionData();
});
```

Point 1: Create an instance of the SiW Replacement Screen Helper class.

Point 2: Set the size of the frame to 1124 x 400 pixels. The page will automatically “fill” to this size.

Point 3: Request session information from the current session. A Session Data response message will be received within the `messageHandler` method.

Within the `list-of-customer-accounts.js` file, you will find the following `messageHandler` method (shown with additional discussion point indicators in blue)…

```
     function messageHandler(event) {
     try{
     1var ctrlMsg = JSON.parse(event.data);
     2switch(ctrlMsg.type) {
     3case "siwControlMsg": {
     4switch (ctrlMsg.subType) {
     5case "SessionData": {
     m_session_data = ctrlMsg.data;
$('body').on('initialized', function() {
     var func = function() {
     createUX3Table(arguments[0]);
     }
     6m_obj_helper.loadJSON("./list-of-customer-accounts-
     service.xsp?searchCompany=" + m_session_data.company, func);
7}).initialize({locale: m_session_data.locale.
                                                  replace("_","-")});
     }
     break;
     }
     }
     break;
     }
     }
```

```
    catch(e) {
    alert(e.description);
    }
    }
```

Point 1: Convert the incoming string to a JSON object.

Point 2: Filter the operation based on the `type` element's value.

Point 3: We are only interested in `siwControlMsg` message types.

Point 4: Filter the operation based on the `subType` element's value.

Point 5: We are only interested in `SessionData` message types. This message is triggered by the `requestSessionData` call.

Point 6: Use the Screen Replacement Helper `loadJSON` method to call the XSP page, passing the current company as a dynamic parameter (i.e., only show data relevant to the current Infor ERP System21 Company that is being used in Infor System i Workspace AnyWhere). The JSON data, returned by the XSP page, will be passed into the `createUX3Table` method, which will update the display with an Infor SoHo Xi Table Control.

Point 7: Initialise the Infor SoHo Xi controls using the locale value passed in the session data. Once all the SoHo Xi controls are initialised, the function assigned to the "`initialized`" event will be processed.

Within the `list-of-customer-accounts-service.xsp` file, you will find the following code fragments (shown with additional discussion point indicators in blue)…

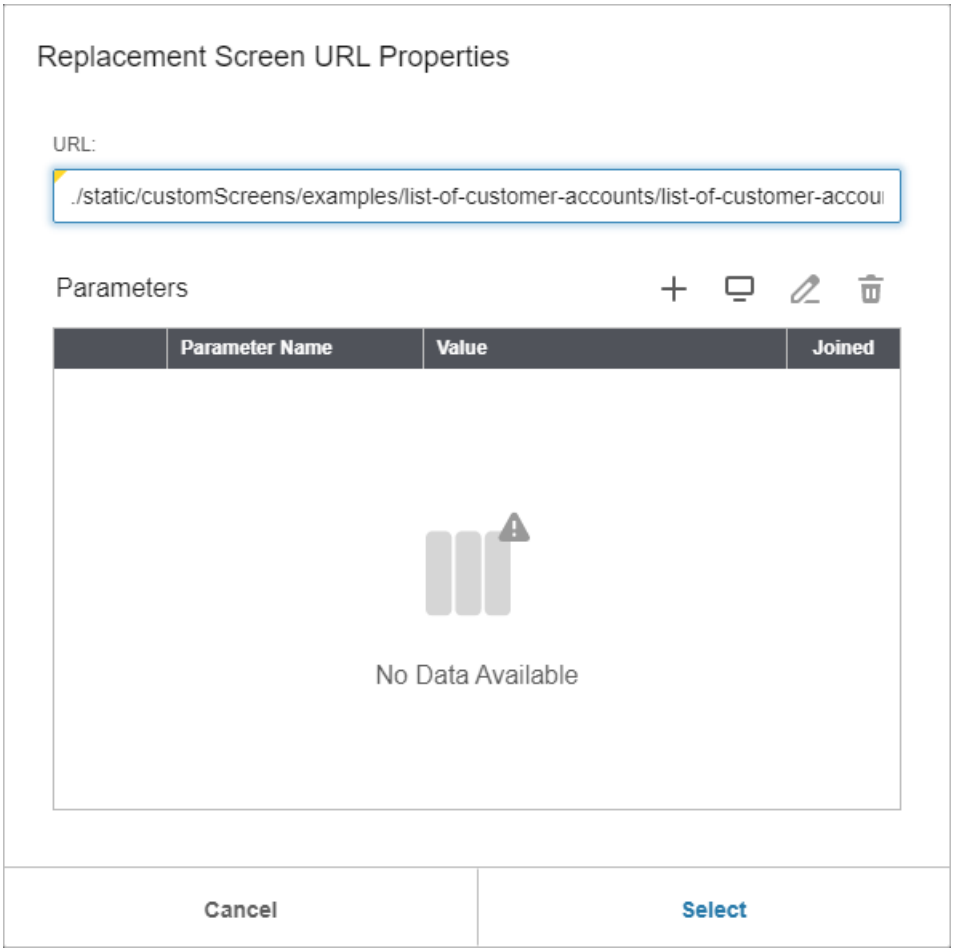
```
1<xsl:variable name="cusSQL">
SELECT DISTINCT cusn05, cnam05, pcd105, pcd205, crlm05, curn05 FROM slp05
WHERE cono05=? ORDER BY cusn05 ASC
</xsl:variable>
<xsl:variable name="cusSQLParams">
<xsl:value-of select="string($searchCompany)"/>
</xsl:variable>
2<xsl:variable name="cusResults"
                    select="sql:sqlPreparedQuery(string($global-sql-
               pool),string($global-environment),'SL', $searchVersion,
               normalize-space(string($cusSQL)), normalize-
               space(string($cusSQLprams)), -1)"/>
<xsl:variable name="rowCount" select="count($cusResults/resultset/row)"/>
3[<xsl:for-each select="$cusResults/resultset/row">{"id":"<xsl:value-of
select="position()"/>",account:"<xsl:value-of
select="./column[@name='CUSN05']/@value"/>","desc":"<xsl:value-of
select="./column[@name='CNAM05']/@value"/>","postCode":"<xsl:value-of
select="./column[@name='PCD105']/@value"/> <xsl:value-of
select="./column[@name='PCD205']/@value"/>","credit":"<xsl:value-of
select="./column[@name='CRLM05']/@value"/>","currency":"<xsl:value-of
select="./column[@name='CURN05']/@value"/>"}<xsl:if test="position() !=
$rowCount">,</xsl:if></xsl:for-each>]
```

Point 1: Define the SQL statement to execute against the Infor ERP System21 database, using the company code passed from the client-side code as a dynamic parameter to filter to a single Infor System21 Company.

Point 2: Use the `sqlPreparedQuery` extension function to execute the SQL statement and return the results (in XML format).

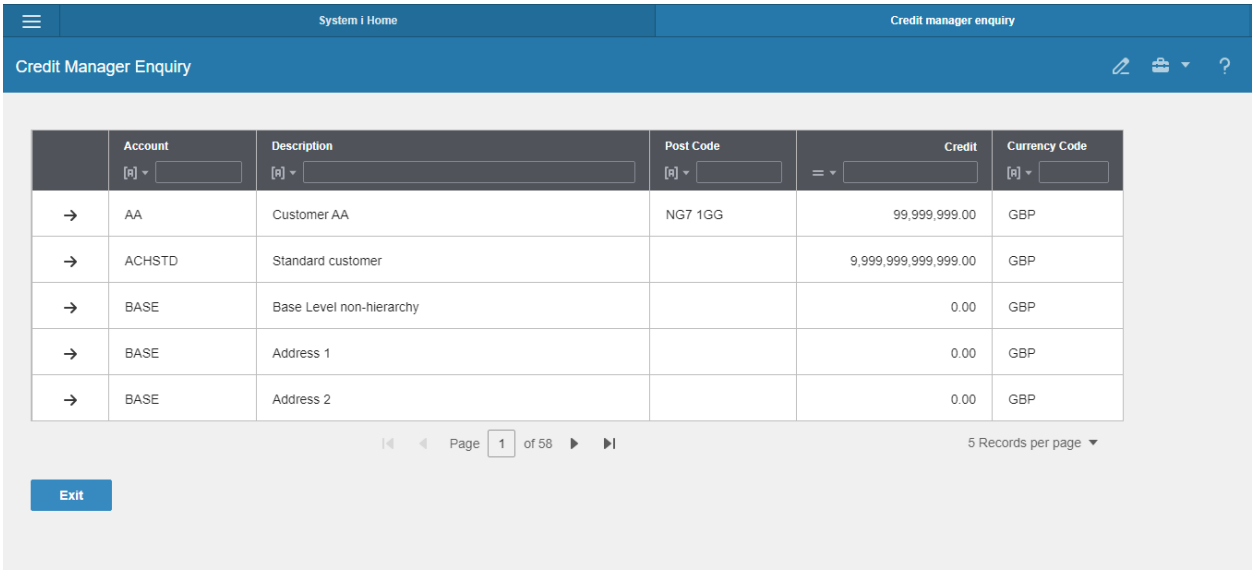
Point 3: Convert the XML results to a JSON format that the Infor SoHo Xi Table Control can consume.

To activate this example, start Infor System i Workspace AnyWhere and run the Credit Manager Enquiry task (1/ARE). Open Designer and sect the *Screen -> Define Replacement Screen* option and enter the full URL to the `list-of-customer-accounts.html` file, E.g.



Click *OK* and save the Design change.

On Exit from Designer, the regular IBM i application screen should disappear and be replaced by the new UI, e.g.



The Page Navigation controls or Column Filters within the table header/footer can be used to traverse the list of accounts. Clicking the arrow icon will open the account details (behind the scenes it will send an Action message to the application page containing an instruction to update the edit field with the selected account code and press Enter. Selecting the *Exit* button will send the F3 command, causing the application screen to exit.

While in Screen Replacement mode, the Utilities, Designer and Help icons are still accessible from the Toolbar. By default, the Help may relate to the original IBM i application screen, but you can use the “User Requested Help Event” within extension function to display a different help file specific to the new screen.