



# System i Workspace v2015

## Extensions Guide

### **Important Notices**

The material contained in this publication (including any supplementary information) constitutes and contains confidential and proprietary information of Infor.

By gaining access to the attached, you acknowledge and agree that the material (including any modification, translation or adaptation of the material) and all copyright, trade secrets and all other right, title and interest therein, are the sole property of Infor and that you shall not gain right, title or interest in the material (including any modification, translation or adaptation of the material) by virtue of your review thereof other than the non-exclusive right to use the material solely in connection with and the furtherance of your license and use of software made available to your company from Infor pursuant to a separate agreement, the terms of which separate agreement shall govern your use of this material and all supplemental related materials ("Purpose").

In addition, by accessing the enclosed material, you acknowledge and agree that you are required to maintain such material in strict confidence and that your use of such material is limited to the Purpose described above. Although Infor has taken due care to ensure that the material included in this publication is accurate and complete, Infor cannot warrant that the information contained in this publication is complete, does not contain typographical or other errors, or will meet your specific requirements. As such, Infor does not assume and hereby disclaims all liability, consequential or otherwise, for any loss or damage to any person or entity which is caused by or relates to errors or omissions in this publication (including any supplementary information), whether such errors or omissions result from negligence, accident or any other cause.

Without limitation, U.S. export control laws and other applicable export and import laws govern your use of this material and you will neither export or re-export, directly or indirectly, this material nor any related materials or supplemental information in violation of such laws, or use such materials for any purpose prohibited by such laws.

### **Trademark Acknowledgements**

The word and design marks set forth herein are trademarks and/or registered trademarks of Infor and/or related affiliates and subsidiaries. All rights reserved. All other company, product, trade or service names referenced may be registered trademarks or trademarks of their respective owners.

### **Publication Information**

Release: System i Workspace v2015

Publication date: November 16, 2016

---

---

# Contents

<b>About this guide</b>	<b>7</b>
Intended audience	7
Organization	7
Related documents	8
Contacting Infor	8
<b>Chapter 1 Introduction</b>	<b>9</b>
Welcome	9
<b>Chapter 2 Events</b>	<b>11</b>
Overview	11
The Emulator Extensions file	11
Additional Extension files	12
The control screen visibility event	13
The screen updated event	14
The user action performed event	15
The user requested help event	17
<b>Chapter 3 Methods</b>	<b>19</b>
Overview	19
Emulator methods	19
GetFieldValue	19
GetFieldValueByIndex	20
SetFieldValue	21
SetFieldValueByIndex	21
GetIndexCount	22
SendKey	22
AddTooltip	23
SetFieldColour	23
SetFieldColourByIndex	24

SetFieldEnabled .....	25
SetFieldEnabledByIndex .....	26
SetFieldVisible .....	26
SetFieldVisibleByIndex .....	26
SetFieldAlignment .....	27
SetFieldAlignmentByIndex .....	28
GetFieldAlignment .....	28
GetFieldAlignmentByIndex .....	29
GetFieldAttribute .....	29
GetFieldAttributeByIndex .....	30
GetVariable .....	31
SetVariable .....	31
SetButtonHighlight .....	32
SetButtonHighlightByIndex .....	33
SetFieldFormat .....	33
SetFieldFormatByIndex .....	34
JavaScript Methods .....	35
setDiv2 .....	35
showDiv2 .....	35
clearDiv2 .....	36
moveDiv2 .....	36
showEmul .....	37
<b>Chapter 4    Variables .....</b>	<b>39</b>
Available values .....	39
<b>Chapter 5    Re-using a Jacada Extension .....</b>	<b>41</b>
Overview .....	41
Matching a screen .....	41
Updating div2 .....	42
Retrieving a field value .....	42
Updating a field value .....	43
Pressing a key .....	43
<b>Chapter 6    Standards for System i Workspace Extensions .....</b>	<b>45</b>
Overview .....	45
Impact for New Extensions .....	46

---

Impact for Existing Extensions .....	46
Common Problems .....	47
Missing Units .....	47
Percentage Units .....	48
Using Tables for Layout .....	50
<b>Chapter 7 Replacing an Entire Screen .....</b>	<b>51</b>
Overview of Screen Replacement .....	51
Supported JSON Messages .....	52
Outgoing Messages .....	52
Request Session Data .....	52
Request Screen Data .....	52
Set Frame Size .....	52
Perform an Action .....	53
Incoming Messages .....	54
Adding a Message Listener to your Replacement Screen .....	54
Session Data Response .....	55
Screen Data Response .....	56
Infor Business Context Messages .....	58
The Screen Replacement Helper Class .....	60
Initializing the Screen Replacement Helper Class .....	60
Screen Replacement Helper Methods .....	61
loadJSON .....	61
requestScreenData .....	61
requestSessionData .....	62
sendActionMessage .....	62
setFrameSize .....	62
<b>Chapter 8 Examples .....</b>	<b>63</b>
Adding client-side validation .....	63
Pre-filling screen values .....	68
Auto-navigating screens .....	70
Running server-side code .....	72
Using a Server-side Mapping File .....	72
Using a IBM i Database .....	74
Export an entire table to the clipboard .....	78
Avoiding Screen Flicker .....	83

---

Replacing a Prompt Screen with a Table .....	86
--	----

---

## About this guide

This guide covers the programmatic features of the System i Workspace v2015 Emulator product.

## Intended audience

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

Some of the terms and concepts used in this guide are covered within the System i Workspace v2015 Emulator Designer Guide, which should be read 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 from the Workspace Emulator.
Methods	The API methods available on the Workspace Emulator and within JavaScript.
Variables	System information variables that you can use within your Workspace Extensions.
Re-using a Jacada Extension	Information on how to convert a Jacada extension written for a previous release of Workspace to use the 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 System i Workspace provides to make this process as simple as possible.
Examples	Examples of using Workspace Extensions.

## Related documents

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

## Contacting Infor

If you have questions about Infor products, go to the Infor Xtreme Support portal at [www.infor.com/inforxtreme](http://www.infor.com/inforxtreme).

Updates to the product documentation and product fixes (PTFs) can be found on the Infor Xtreme website under Knowledge Base article 1648078:

<https://www.inforxtreme.com/espublic/EN/AnswerLinkDotNet/SoHo/Solutions/SoHoViewSolution.aspx?SolutionID=1648078>

We recommend that you subscribe to this solution, so that you can be informed automatically of any changes.

If you have comments about Infor documentation, contact [documentation@infor.com](mailto:documentation@infor.com).

### Welcome

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

For brevity, within this guide, we will use the term Emulator to reference the System i Workspace v2015 Emulator and we will use the term Designer to reference the System i Emulator Designer.

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

The 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 Emulator Extensions.

This guide assumes that the reader has a good understanding of JavaScript and HTML and the DOM within Internet Explorer, and XSL/XML. No guidance is provided for these technologies.



### 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\Workspace2015\tomcat\webapps\systemi\static\customScripts` folder on your Workspace 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\Workspace2015.ear\Workspace.war\static\customScripts` folder on your Workspace 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 Workspace server.

**Caution:** Changes made within this file will affect all Workspace profiles where the 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 Workspace 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 the System i Emulator. 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 System i Emulator.

## Additional Extension files

If you wish to import any additional JavaScript files into the Emulator web-page, such as 3<sup>rd</sup> party JavaScript libraries, you can place these files in a sub-folder of the `customScripts` folder within Workspace 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\Workspace2015\tomcat\webapps\systemi\static` folder on your Workspace 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\Workspace2015.ear\Workspace.war\static` folder on your Workspace server.

**Caution:** You must ensure that any 3<sup>rd</sup> party JavaScript files you import do not interfere with or replace code required for either Workspace or the 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 Workspace installation.

## The control screen visibility event

This event is fired by the 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 Workspace. Remove the comments from the start of each line to enable the function.

The function receives two parameters...

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

You can find the Magic Number of a screen by using the Emulator context menu and selecting the *About* menu item. The current Magic Number will be displayed in a popup dialog (referred to as *All Magics*).

Using the two Magic Number parameters, you can decide whether or not you want to show the screen. If you wish to hide the current screen, you should return a value of **1** 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 Emulator when a new screen arrives, but user-interaction is not 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 Workspace. Remove the comments from the start of each line to enable the function.

The function receives four parameters...

Parameter	Description
<code>strScreenId</code>	A string value containing the unique identifier for the current application screen (also known as a Magic Number).
<code>strLastScreenId</code>	A string value containing the unique identifier for the previous application screen.
<code>blnPopup</code>	A boolean value, set to true if the current application screen is within a Popup Window otherwise it will be false.
<code>obj5250</code>	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 Magic Number parameters, you can decide whether or not 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 "OE02X990E03699":
            obj5250.SetFieldValueByIndex(1, 1, "GC01");
            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 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 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.

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 Workspace. Remove the comments from the start of each line to enable the function.

The function receives three parameters...

Parameter	Description
<code>strScreenId</code>	A string value containing the unique identifier for the current application screen (also known as a Magic Number).
<code>intAction</code>	An integer value for the user's action (i.e. the aid or function key they have activated via the keyboard or mouse).
<code>obj5250</code>	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.

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 Magic Number parameter, you can decide whether or not 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 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 Emulator when the user requests help by pressing/activating the F1 command key. You can use this function to carry add 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) {  
}
```

There is a commented-out example of this method within the standard 5250-extensions.js shipped with Workspace. 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).

---

Parameter	Description
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 standard Emulator help processing, you should return a value of **0**. If you want to block the standard Emulator help processing and send the F1 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;  
    }  
    return 0;  
}
```

### Overview

The 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 methods that can be used to alter the display.

### Emulator methods

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

**Caution:** The 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
<code>intHostRow</code>	Row position of required field (0 – 23/27)
<code>intHostCol</code>	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 Emulator display is divided up into a grid of eighty columns and twenty four rows. Each field will have a unique row and column parameter position.

This example will return the value from the field on row ten, column position one...

```
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 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:

The screenshot shows the IBM i System i Workspace interface. At the top, there's a header with the Infor logo and 'System i Workspace'. Below that, a navigation bar shows 'System i Home' and 'Credit manager enquiry'. The main title is 'Credit Manager Enquiry'. A toolbar contains icons for 'Actions', a list view, a print icon, and a help icon. The screen displays several fields: 'SL243', 'Z1 - UK Demo Company', 'User WKS3\_5250', and '2/07/15 09:47:41'. There is an 'Account .....?:' field with a search icon. At the bottom, there are 'Exit' and 'Prompt' buttons.

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 three..

```
var value = obj5250.GetFieldValueByIndex(3, 2);
```

...which in the above screen is the text WKS3\_5250.

## 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
<code>intFieldType</code>	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
<code>intAction</code>	The action key code

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

<code>intAction</code> Value	User's Action
42	Enter
43 to 66	Command F1 to F24

intAction Value	User's Action
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
<code>intHostRow</code>	Row position of required field (0 – 23/27)
<code>intHostCol</code>	Column position of required field (0 – 79/132)
<code>intColour</code>	The color to apply to this field (see below)
<code>blnUnderline</code>	If true, apply the underline style to the field
<code>blnReverse</code>	If true, apply the reverse-image style to the field

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

intColour Value	Description
<code>Emulator_Colour_Green</code>	Apply the IBM 5250 Green color
<code>Emulator_Colour_White</code>	Apply the IBM 5250 White color
<code>Emulator_Colour_Red</code>	Apply the IBM 5250 Red color
<code>Emulator_Colour_Red_Highlight</code>	Apply the IBM 5250 Red Highlight color
<code>Emulator_Colour_Cyan</code>	Apply the IBM 5250 Turquoise color
<code>Emulator_Colour_Yellow</code>	Apply the IBM 5250 Yellow color
<code>Emulator_Colour_Pink</code>	Apply the IBM 5250 Pink color
<code>Emulator_Colour_Blue</code>	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
<code>intIndex</code>	The field index (starts at 0).
<code>intFieldType</code>	The field type (1 = edit, 2= static)
<code>intColour</code>	The color to apply to this field (see below)
<code>blnUnderline</code>	If true, apply the underline style to the field
<code>blnReverse</code>	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 above.

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
<code>intHostRow</code>	Row position of required field (0 – 23/27)
<code>intHostCol</code>	Column position of required field (0 – 79/132)
<code>blnEnable</code>	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
<code>intIndex</code>	The field index (starts at 0).
<code>blnEnable</code>	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
<code>intHostRow</code>	Row position of required field (0 – 23/27)
<code>intHostCol</code>	Column position of required field (0 – 79/132)
<code>blnVisible</code>	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
<code>intIndex</code>	The field index (starts at 0).
<code>intFieldType</code>	The field type (1 = edit, 2= static)
<code>blnVisible</code>	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
<code>intHostRow</code>	Row position of required field (0 – 23/27)
<code>intHostCol</code>	Column position of required field (0 – 79/132)
<code>intAlignment</code>	One of the pre-defined alignment variables

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

<code>intAlignment</code> Value	Description
<code>Emulator_Align_Left</code>	Left align the field text
<code>Emulator_Align_Center</code>	Center align the field text
<code>Emulator_Align_Right</code>	Right align the field text
<code>Emulator_Align_Default</code>	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 Label field alignment at the supplied index (see `GetFieldValueByIndex` for more information on indexes). This method takes three parameters...

Parameter	Description
<code>intIndex</code>	The field index (starts at 0).
<code>intFieldType</code>	The field type (2= static)
<code>intAlignment</code>	One of the pre-defined alignment variables

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

This method does not return a value.

This example will right-align the Label 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
<code>intHostRow</code>	Row position of required field (0 – 23/27)
<code>intHostCol</code>	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
<code>intIndex</code>	The field index (starts at 0).
<code>intFieldType</code>	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
<code>intHostRow</code>	Row position of required field (0 – 23/27)
<code>intHostCol</code>	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
<code>Emulator_Colour_Green</code>	The IBM 5250 Green color
<code>Emulator_Colour_White</code>	The IBM 5250 White color
<code>Emulator_Colour_Red</code>	The IBM 5250 Red color
<code>Emulator_Colour_Red_Highlight</code>	The IBM 5250 Red Highlight color

intColour Value	Description
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) alert("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) alert("Field is hidden");
```

## GetVariable

This method retrieves the content of a Designer Variable within the current IBM i application screen. See the System i Workspace v2015 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 System i Workspace v2015 Designer Guide for more details about Designer Variables.

**Caution:** The Designer Variable must have been created within the System i Workspace v2015 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).

The assignment will fail if the Designer Variable does not exist in the current Workspace Profile design data or if you attempt to put a blank/empty value into the Designer Variable and it has been configured not to accept blank/empty values.

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

```
If (obj5250.SetVariable("Last Order Number", "1000") == true) {  
    alert(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_Orange	Draw the button using the Infor User Experience highlight (Orange background)
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

This method does not return a value.

This example will apply the Orange background highlight to the button field at row twenty three and column ten...

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

## 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
<code>intIndex</code>	The field index (starts at 0).
<code>intFieldType</code>	The field type (2= static)
<code>intHighlight</code>	One of the pre-defined highlight variables

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

This method does not return a value.

This example will apply the Orange background highlight to the button field at index ten...

```
obj5250.SetButtonHighlight(10, 2, Emulator_Highlight_Orange);
```

## SetFieldFormat

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
<code>intHostRow</code>	Row position of required field (0 – 23/27)
<code>intHostCol</code>	Column position of required field (0 – 79/132)
<code>strFormat</code>	Field Format to apply to this field

See [http://msdn.microsoft.com/en-us/library/microsoft.visualbasic.strings.format\(v=vs.110\).aspx](http://msdn.microsoft.com/en-us/library/microsoft.visualbasic.strings.format(v=vs.110).aspx) for more information about Field Formats.

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

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
<code>intIndex</code>	The field index (starts at 0).
<code>intFieldType</code>	The field type (2= static)
<code>strFormat</code>	Field Format to apply to this field

See [http://msdn.microsoft.com/en-us/library/microsoft.visualbasic.strings.format\(v=vs.110\).aspx](http://msdn.microsoft.com/en-us/library/microsoft.visualbasic.strings.format(v=vs.110).aspx) for more information about Field Formats.

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 Methods

## setDiv2

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

This method takes a single parameter...

Parameter	Description
<code>strHTML</code>	A string value containing HTML that will be applied into the <code>div2</code> 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 Emulator task.

This method takes a single parameter...

Parameter	Description
<code>blnShow</code>	A boolean value. Set to true to show the <code>div2</code> 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 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 Emulator page.

This method takes a single parameter...

Parameter	Description
nPos	Position of the div2. Pass 0 to move below the Emulator (default) or 1 to move to the right of the 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 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 Emulator object within the current tab.

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 with the first screen of the Account Enquiry (1/ARE) 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" && strLastScreenId == "AM18099") {
        setDiv2("<p>Hidden panel</p><button onclick=\"showEmul
                (true);\"value=\"Show Emulator\">Show Emulator
                </button><button onclick=\"showEmul(false);\"
                value=\"Hide Emulator\">Hide Emulator</button>");
        showDiv2(true);
    }
}
```

### Available values

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

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

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



### Overview

An extension written for the Jacada UI (used in previous Workspace releases) cannot be used directly with the 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 Emulator uses. Some other important changes are...

- In Jacada, screens were recognised using the panel name defined within the Jacada ACE tool. In the Emulator we use the hidden Magic Number assigned to each screen.
- In Jacada, fields are referenced using the field name defined within the Jacada ACE tool. In the Emulator, you can use the 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 done 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 Changed event within the Emulator; 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 the Emulator, 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 the Emulator, 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 the Emulator, 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 the Emulator, you would use the `SendKey` method.



---

## Chapter 6 Standards for System i Workspace Extensions

# 6

### Overview

Infor System i Workspace v2015 has been re-written to support modern browser 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 System i Workspace v2015, 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.

Each release of Internet Explorer has been more compliant with W3C standards. At the time of writing, the latest supported Internet Explorer release for System i Workspace is version 11, which is highly standards compliant.

The `div2` element, which will contain any Extension HTML, is a static floating 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 System i Emulator, it will have a width style applied that is calculated using the page-width minus the width of the System i Emulator screen.

## 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 System i 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 (e.g. [http://msdn.microsoft.com/en-us/library/yek4tbz0\(v=vs.94\).aspx](http://msdn.microsoft.com/en-us/library/yek4tbz0(v=vs.94).aspx)).

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.

For Internet Explorer, this page explains the supported CSS standards for the different releases: [http://msdn.microsoft.com/en-us/library/hh781508\(v=vs.85\).aspx](http://msdn.microsoft.com/en-us/library/hh781508(v=vs.85).aspx).

This page lists the supported HTML elements (including which are standard and which are non-standard) [http://msdn.microsoft.com/en-us/library/hh772721\(v=vs.85\).aspx](http://msdn.microsoft.com/en-us/library/hh772721(v=vs.85).aspx).

W3C provides a CSS standards checking web-site that is very useful for checking any pages that you are planning to host within System i Workspace <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 System i Emulator page also includes the jQuery API JavaScript extensions. At time of writing, this is jQuery version 1.11.1 and jQuery UI version 1.10.4. 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 System i Workspace v2015.

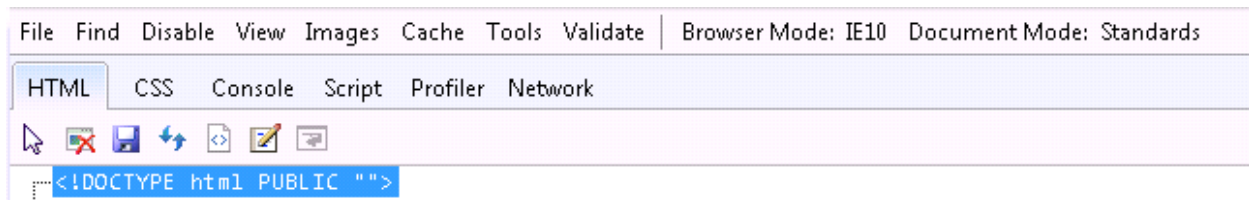
For existing Extensions that use no HTML/CSS then you should review the System i Emulator methods and events in this document to make sure you are using them correctly. As a general principal, we try not to change these methods and events between releases, so that Extensions from previous releases continue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

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

The level of standards depends on the versions of Internet Explorer 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 web-pages, then they could also be affected. We would recommend that any pages that are embedded within System i Workspace should use the HTML5 document type. You should then be able to test them outside of System i Workspace to check they function correctly.

To check the standards of an external page, you can use the Internet Explorer F12 Developer Tools option. Open the external page in your browser, start Developer Tools and it will show you the page source and also tell you which Document Mode is being used; E.g.



**Note:** If you need help with upgrading your existing 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 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="http://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="http://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 System i Workspace `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 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.

### 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).

Whilst it is perfectly possible to use the previously discussed System i 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 this release of System i Workspace. 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 W3C standards-based communication between your replacement screen and System i Workspace.

Screen Replacement is initiated using Designer (see the appropriate *System i Workspace – 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, Workspace will automatically hide the Emulator and display your web interface in its place. The Screen Replacement can then interrogate the Emulator and Workspace for details and drive the screen by populating fields and firing actions that. All this is done via the exchange of JSON messages, though we 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 System i Workspace. 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 System i Workspace to request a response containing a collection of information about the current System i Workspace installation and session (response message is described below);

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

## Request Screen Data

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

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

## Set Frame Size

Send the following message to System i Workspace 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 Emulator field and/or simulate a user action, send the following message to System i Workspace;

```
{
  "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 Emulator 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 Emulator will do any necessary type conversion).

The value of the `command` property should be one 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 System i 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.

## 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 System i Workspace. 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 = null;
        eval("ctrlMsg = " + event.data);
        :
        :
    } catch(e) {
        // Message was not JSON format, or had invalid syntax
        alert(e.description);
    }
}
```

```

    }
}

```

The `ctrlMsg` object can then be used to extract any information. If the message came from System i Workspace, 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 System i Workspace, 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 code
company	The current Company Code
companyDesc	The expanded description associated with the above code
locale	The current System i Workspace 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 System i Workspace server
staticURL	The URL to the System i Workspace 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
<code>edit_fields</code>	An array of zero or more elements describing any editable fields within the current screen
<code>static_fields</code>	An array of zero or more elements describing any static/label fields within the current screen
<code>commands</code>	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
<code>index</code>	The numeric index of the edit field (see <code>GetFieldValueByIndex</code> for more information on indexes)
<code>value</code>	The value of the edit field
<code>readOnly</code>	If the edit field can be altered interactively, this value will be <code>false</code> , otherwise <code>true</code>
<code>length</code>	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
<code>index</code>	The numeric index of the static field (see <code>GetFieldValueByIndex</code> for more information on indexes)
<code>value</code>	The value of the static field

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

Property	Description
<code>command</code>	The IBM i command (ENTER,F1-F24,PAGEUP,PAGEDOWN)
<code>value</code>	The command description

All of 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. They are broadcast to all windows inside System i Workspace.

Here is an example Infor Business Context Message;

```
{
  "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.
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 <code>inforPurchaseOrder</code> , <code>inforPurchaseOrderLine</code> , <code>inforCustomerPartyMaster</code> , etc. This is a mandatory field.
id1 to id15	Depending on the business entity type, the id fields (e.g.: <code>id1</code> , <code>id2</code> , etc...) can be from 1 to 15. <code>id1</code> 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.

Property	Description
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 <code>bodReference</code> object.
documentId	The documentId of the noun instance. This is a mandatory field for every existing instance of the <code>bodReference</code> object.

## The Screen Replacement Helper Class

The Screen Replacement Helper class is a JavaScript object that can be initiated and used to post messages in the correct syntax to the main System i 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 `server\screen\shared\js` sub-folder of your System i Workspace 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 System i Emulator page.

### 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 to the System i Emulator page for Screen Data. Your message listener will receive a Screen Data Response.

## requestSessionData

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

Sends a request to the System i Emulator page for Session Data. Your message listener will receive a Session Data Response.

## sendActionMessage

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

Send an action and/or a set of field updates to the System i Emulator page.

The value of the `command` property should be one 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.

### Adding client-side validation

In this example we will analyse the data entered into 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 Magic Number of **PL00R99**) containing supplier name, address etc.; for example:

infor System i Workspace

System i Home [Maintain suppliers](#)

Actions [Icons]

PL015 Z1 - UK Demo Company USER WKS3\_5250 2/07/15

Supplier Maintenance

Account ... TESTEXT1 **NEW**

Name .....  Alpha Seq

Address ...

Country ..?  Post code ...   
Language ...?

Phone .....  Fax .....

Remarks ...

Website ...

Primary Contact ..... ?

Groups 1-4....?

Exit Prompt Previous Contacts Fixed Format Address  
Extended Attributes

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 Workspace 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...

 System i Workspace

System i Home [Maintain suppliers](#)

Actions



PL015      Z1 - UK Demo Company      USER   WKS3\_5250      2/07/15

Supplier Maintenance

Account:      TESTEXT1      **NEW**

Name:            Alpha Seq

Address:

Country:            Post code:        
Language:

Phone:            Fax:

Remarks:

Website:

Primary Contact:

Groups 1-4:

F3:Exit      F4:Prompt      F12:Previous      F13:Contacts      F15:Fixed Format Address


F20:Extended Attributes

The phone and fax fields cannot be blank.












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...

System i Workspace v2015 Extensions Guide | 67

 System i Workspace

System i Home [Maintain suppliers](#)

Actions           

PL015      Z1 - UK Demo Company      USER    WKS3\_5250    2/07/15

Supplier Maintenance

Account:    TESTEXT1    NEW

Name:        Alpha Seq

Address:

Country:        Post code:      
Language:

Phone:        Fax:

Remarks:

Website:

Primary Contact:

Groups 1-4:

F3:Exit    F4:Prompt    F12:Previous    F13:Contacts    F15:Fixed Format Address

F20:Extended Attributes

The phone number is not a valid format (99999 999999).

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

## Pre-filling screen values

In this example, we will assume that each Workspace 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 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 Workspace 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:

The screenshot displays the Infor System i Workspace application. At the top, the header shows the Infor logo and 'System i Workspace'. Below this, a navigation bar includes 'System i Home' and a link 'Enquire on supplier'. A toolbar with various icons (Actions, Print, Save, etc.) is visible. The main content area shows the 'Supplier Enquiry' screen with the following details: 'PL500', 'Z1 - UK Demo Company', 'User', 'WKS3\_5250', and '2/07/15'. The 'Account code ...?' field is populated with 'SUPP2'. An 'Exit' button is located at the bottom left.

## 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...

```
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 Workspace 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 Workspace 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 Workspace server (or indeed any other web server), we will use the Microsoft XMLHttpRequest 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 Workspace 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 Workspace 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 name from the mapping XML using the `user` parameter via an XPath query. The supplier 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 Workspace 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 may 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 Workspace 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 Workspace 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 a IBM i Database

Let's us now look at dynamically retrieving the supplier from a 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...

**infor** System i Workspace

System i Home [Maintain descriptions](#)

Descriptions File Maintenance WSPC

Actions

Description type.....: TYPE  
Major type.....: WSPC  
Description.....:   
Description limit.....:  PV...: ☐

Desc'n I.D.	Description	Limit
ACDF	AC Terminology Definitions	30
ADES		
ADFM	Address format	30
ADJT	Adjustment Type	30
AEDI	Automatic Update from EDI	30
ALTT	Alternative Item Type	30
AN	Agent	30
ANTP	Article Number Type	
ARPM	Arrears Processing Method	30
ASAD	AS Address Type Codes	30

Exit Previous Description code details Language descriptions

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...

**infor** System i Workspace

System i Home [Maintain descriptions](#)

Descriptions File Maintenance WSPC

Actions

IN037 Z1 - UK Demo Company USER WKS3\_5250 2/07/15 10:34:15

Description type...: WSPC EXTEXAM Limit..: 8  
Description code...:

Desc'n I.D.	Description	Limit	Tax	PV	Rate
<input type="checkbox"/> USER1	SUPP1				.00

1=Select 2=Language descriptions

Exit Previous

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="$t2lib"/>/DESC
      WHERE CONO15=? AND PRMT15=? AND PSAR15=?
    </xsl:variable>
    <xsl:variable name="sqlPrams">
      <xsl:value-of select="$session-company"/>,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($sqlPrams), 1)"/>
    <xsl:value-of
select="$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 Workspace 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 a IBM i database, using the correct database pool (which you will have defined using Workspace Configuration when you installed Workspace).

The `sqlPreparedQuery` XSL extension function takes four parameters...

Parameter	Description
<code>pool</code>	A Workspace database pool defined within the Database Connections tab in Workspace Configuration.
<code>sql</code>	The SQL statement you wish to execute. Use question marks where you want to pass in parameters.
<code>sqlPrms</code>	A comma separated list of parameters which will be used within the SQL statement.
<code>records</code>	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 Workspace 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 Workspace server.

On starting the Supplier Maintenance task, the user profile is examined within the 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:

System i Workspace

System i Home
[Enquire on supplier](#)

Actions

PL500
Z1 - UK Demo Company
User WKS3\_5250
2/07/15

Supplier Enquiry -
All Items (Prime)

Account code ... ?
Premier Food and Drug Supplies..
Text

Currency ..... ?
Log value .... :
132405.95

Item details .....
Ledger value... :
111953.20

Item transactions ...
Collection docs :
.00

	Reference	Txt	Doc Date	Original	Outstanding	
1	IN 00001021		22/06/15	1200.00	1200.00	GBP
2	IN 00001012		19/06/15	8500.00	8500.00	GBP
3	IN 00000988		3/06/15	222.00	222.00	GBP
4	IN 00000987		3/06/15	111.00	111.00	GBP
5	IN 00000981		22/05/15	10000.00	10000.00	GBP
6	IN 00000976		13/04/15	2500.00	2500.00	GBP
7	IN 00001008		1/04/15	10000.00	10000.00	GBP
8	IN 00000977		1/04/15	12000.00	12000.00	GBP
9	PY 20001363		23/03/15	987.23-	.00	GBP
10	IN 00000946		9/03/15	510.00	510.00	GBP
11	PY 20001345		3/02/15	1200.00-	.00	GBP
12	IN 00000930		3/02/15	1200.00	.00	GBP

+

Exit

Text

Log items

Open items

All items

Pay history

Next payment

More keys

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><textarea id=\"copytext\"  
style=\"display:none\"></textarea>");

                showDiv2(true);

            }

            else if (m_mode == "paging") {

                showDiv2(false);

                captureTable();

            }

            break;

        default:

            showDiv2(false);

            break;

    }

}

function startCapture() {

    m_mode = "paging";

    m_buffer = "";
```

```
captureTable();
}

function captureTable() {
    // Add data onto the buffer
    for (i=9; i<21; i++) {
        for (j=0; j<8; j++) {
            m_buffer = m_buffer + m_objEmulInternal.GetFieldValue(i, m_cols[j]);
            if (j<7) m_buffer = m_buffer + "\t";
        }
        m_buffer = m_buffer + "\n";
    }
    // Check for more data
    var strPlus = m_objEmulInternal.GetFieldValue(21,1)
    if (strPlus == "") {
        // Copy data to the clipboard
        m_mode = "";
        copytext.innerText = m_buffer;

        var Copied = copytext.createTextRange();
        Copied.execCommand("Copy");

        alert("Copied data to the clipboard.");
    } else {
        // Move to next page
        m_objEmulInternal.SendKey(74);
    }
}
```

**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.

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 two elements to the `div2` HTML object. The first is a button that, when selected, will start the copy process by calling the `startCapture` method. The second is a hidden text area that is used in getting the text from the buffer to the Windows Clipboard.

When the `m_mode` value is set to **paging**, we hide the `div2` HTML object and call the `captureTable` 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 then display a message to the user.

If you wish to access the Emulator object outside of the events that pass it as a variable, then you can use the HTML object named `m_objEmulInternal`.

**Caution:** You should never create a variable called `m_objEmulInternal` or try to alter the properties or styles of this HTML object. You should only call the Emulator methods defined in this document.

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

 System i Workspace

System i Home    [Enquire on supplier](#)

Actions

PL500      Z1 - UK Demo Company      User WKS3\_5250      2/07/15

Supplier Enquiry -      All Items (Rev. base)

Account code:  Premier Food and Drug Supplies:      Text  
Currency:       Log value:      132405.95  
Item details:       Ledger value:      111953.20  
Item transactions:       Collection docs :      .00

Reference	Txt	Doc Date	Unrealised Gn/Loss	Outstanding	
1	IN 00001021	22/06/15	.00	1200.00	GBP
2	IN 00001012	19/06/15	.00	8500.00	GBP
3	IN 00000988	3/06/15	.00	222.00	GBP
4	IN 00000987	3/06/15	.00	111.00	GBP
5	IN 00000981	22/05/15	.00	10000.00	GBP
6	IN 00000976	13/04/15	.00	2500.00	GBP
7	IN 00001008	1/04/15	.00	10000.00	GBP
8	IN 00000977	1/04/15	.00	12000.00	GBP
9	PY 20001363	23/03/15	.00	.00	GBP
10	IN 00000946	9/03/15	.00	510.00	GBP
11	PY 20001345	3/02/15	.00	.00	GBP
12	IN 00000930	3/02/15	.00	.00	GBP
+					

F3:Exit

F10:Text

F11:Log items

F13:Open items

F14:All items

F15:Pay history

F16:Next payment

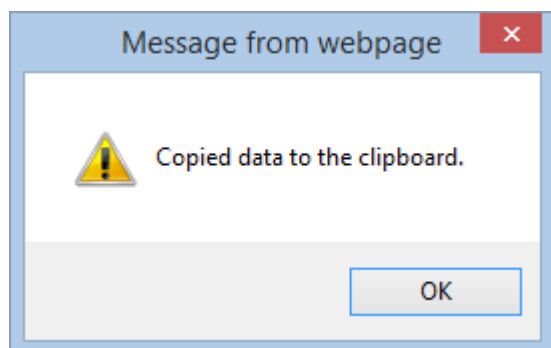
F24:More keys

Export Table To Clipboard

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).

Depending on your browser security settings, the first time you use this process you may see a warning relating to putting data onto the Windows Clipboard.

Once the process is complete, a message will appear; for example:



The data can now be pasted into an application, such as Microsoft Excel; for example:

	A	B	C	D	E	F	G	H	I	J
1	1 IN		480		1/11/11	14.22	14.22	GBP		
2	2 IN		479		1/11/11	119.85	119.85	GBP		
3	3 IN		478		1/11/11	74.03	74.03	GBP		
4	4 IN		477		1/11/11	44.42	44.42	GBP		
5	5 IN		476		1/11/11	116.33	116.33	GBP		
6	6 IN		475		1/11/11	11.63	11.63	GBP		
7	7 IN		474		1/11/11	67.21	67.21	GBP		
8	8 IN		473		1/11/11	37.6	37.6	GBP		
9	9 IN		472		1/11/11	95.18	95.18	GBP		
10	10 IN		471		1/11/11	105.75	105.75	GBP		
11	11 IN		470		1/11/11	94	94	GBP		
12	12 IN		469		1/11/11	105.75	105.75	GBP		
13	1 IN		468		31/10/11	5.88	5.88	GBP		
14	2 IN		464		31/10/11	129.25	129.25	GBP		
15	3 IN		463		31/10/11	117.5	117.5	GBP		
16	4 IN		462		31/10/11	14.1	14.1	GBP		
17	5 IN		460		27/10/11	15.51	15.51	GBP		
18	6 IN		459		27/10/11	11.75	11.75	GBP		
19	7 IN		461		26/10/11	11.75	11.75	GBP		
20	8 IN		458		26/10/11	105.75	105.75	GBP		
21	9 IN		457		26/10/11	82.25	82.25	GBP		
22	10 IN		456		26/10/11	129.25	129.25	GBP		

## 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><textarea id=\"copytext\"  
style=\"display:none\"></textarea>");

                showDiv2(true);

                captureTable();

            }

            break;

        default:

            showDiv2(false);

            break;

    }

}

function startCapture() {

    m_mode = "paging";

    m_buffer = "";

    captureTable();

}

function captureTable() {

    // Add data onto the buffer

    for (i=9; i<21; i++) {
```

---

```
        for (j=0; j<8; j++) {
            m_buffer = m_buffer + m_objEmulInternal.GetFieldValue(i,
m_cols[j]);
            if (j<7) m_buffer = m_buffer + "\t";
        }
        m_buffer = m_buffer + "\n";
    }
    // Check for more data
    var strPlus = m_objEmulInternal.GetFieldValue(21,1)
    if (strPlus == "") {
        // Copy data to the clipboard
        m_mode = "";
        copytext.innerText = m_buffer;
        var Copied = copytext.createTextRange();
        Copied.execCommand("Copy");
        alert("Copied data to the clipboard.");
        clearDiv2();
        m_objEmulInternal.SendKey(76);
    } else {
        // Move to next page
        m_objEmulInternal.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 may have been wrapped. Make sure these are restored to being a single line when you apply this to your server.

When the mode is **paging**, we hide the screen and show a “Processing” message. However, this would result in the screen still being hidden once the table copy process had completed. To avoid this we send the Reset key (action code **76**) which causes the screen to refresh and become visible once more to the user.

## Replacing a Prompt Screen with a Table

In this example, we will use the Designer Replacement Screen functionality to hide the initial System i Emulator screen of the Infor System21 Credit Manager Enquiry (1/ARE) application and replace it with a table control. Normally, the user has to 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 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 System21 data and return it as a JSON message.

To start, locate the `static\customScreens\examples\list-of-customer-accounts` sub-folder of your System i Workspace 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)...

```

1  <body onload="initialise()" class="full tight">
    <div class="inforModuleContent autoLabelWidth">
        <br/>
2      <div id="inforDataGrid"></div>
        <br/>
3      <div class="padLeft5">
          <button type="button"
                onclick="m_obj_helper.sendMessage('F3');"
                class="inforFormButton">Exit</button>
        </div>
    </div>
</body>

```

Point 1: Call the `initialise()` method in `list-customer.accounts.js` once the page has loaded. The CSS classes force the page to take up all available space (see `setFrameSize` call below).

Point 2: This `DIV` element will hold the table. Initially, it has no content.

Point 3: 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 `initialise` method (shown with additional discussion point indicators in blue)...

```

function initialise() {
1   m_obj_helper = new siwReplacementScreenHelper();
2   $("body").inforForm();
3   m_obj_helper.setFrameSize(1124, 400);
4   m_obj_helper.requestSessionData();
}

```

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

Point 2: Infor UI convenience routines to initialise all controls with default parameters.

Point 3: Set the size of the frame to 1124 x 400 pixels. The page will automatically “fill” to this size.

Point 4: Request session information from the System i Emulator. 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 {  
        var ctrlMsg = null;  
1      eval("ctrlMsg = " + event.data);  
2      switch(ctrlMsg.type) {  
3          case "siwControlMsg": {  
4              switch (ctrlMsg.subType) {  
5                  case "SessionData": {  
                      m_session_data = ctrlMsg.data;  
                      var func = function() {  
                          createUX3Table(arguments[0]);  
                      }  
6                      m_obj_helper.loadJSON("./list-of-customer-  
accounts-service.xsp?searchCompany=" + m_session_data.company, func);  
                      }  
                      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 in the `initialise` method.

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 System21 Company that is being used in System i Workspace). The JSON data, returned by the XSP page, will be passed into the `createUX3Table` method, which will update the display with an Infor SoHo Table Control.

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>
2  <xsl:variable name="cusSQLprams">
    <xsl:value-of select="string($searchCompany)"/>
  </xsl:variable>
3  <xsl:variable name="cusResults"
    select="sql:sqlPreparedQuery(string($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)"/>
4  [<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>]

```

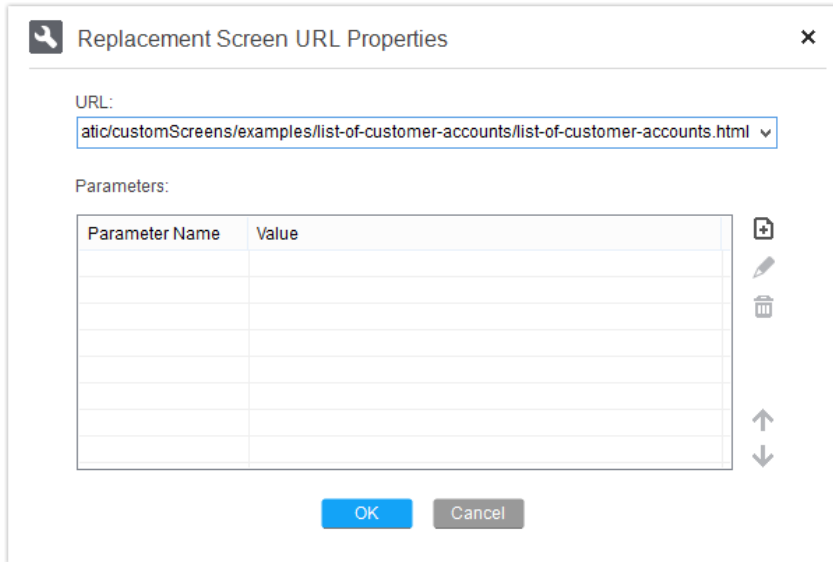
Point 1: Define the SQL statement to execute against the Infor System21 database. Use question marks where you want to pass in parameters.

Point 2: Define a comma separated list of parameters which will be used within the SQL statement. In the example above we have used the company code passed from the client-side code as a dynamic parameter to filter to a single Infor System21 Company.

Point 3: Use the `sqlPreparedQuery` extension function to execute the SQL statement and return the results (in XML format).

Point 4: Convert the XML results to a JSON format that the Infor SoHo Table Control can consume.

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



**Replacement Screen URL Properties**

URL:

Parameters:

Parameter Name	Value

OK Cancel

Click **OK** and save the Design change.


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

#### Credit Manager Enquiry

Account	Description	Post Code	Credit	Currency Code
(a)	(a)	(a)		(a)
UCH3	Petro Inc.		150000.00	USD
UC01	Dennett Chemicals Inc		150000.00	USD
UC02	D.G. Penby Inc		150000.00	USD
UC03	The Drug Store		200000.00	USD
UC04	Eyeliners		200000.00	USD
UC05	Dominiques		150000.00	USD
UC10	US customer		0.00	USD
UC11	Reeves Chemist's	NG7 1GG	999.00	USD
UKWERJ	KDRKJ		0.00	GBP
USD01	US Dollars customer		0.00	USD

Displaying: 531 - 540 of 643 | Selected: 0 | Page 54 of 65

Exit

The Page Navigation controls or Column Filters within the table header/footer can be used to traverse the list of accounts. Clicking the  icon will open the account details (behind the scenes it will send an Action message to the System i Emulator 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 to the System i Emulator page, causing the application screen to exit.

While in Screen Replacement mode, the Widget, 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.