aXes®

Documentation Library

eXtensions Tutorials

aXes Documentation Library

Extensions Tutorial 0 - Getting Started

Check your Development PC is ready

To do application development using eXtensions you need to have an up to date development PC. Complete this checklist:

Check	Okay
Core 2 Duo class processor, equivalent or better (see note 1)	
1Gb RAM or more (see note 1)	
Screen resolution of 1440×1080 or better (see note 2)	
Windows 7 or later.	
Internet Explorer set as the default browser	

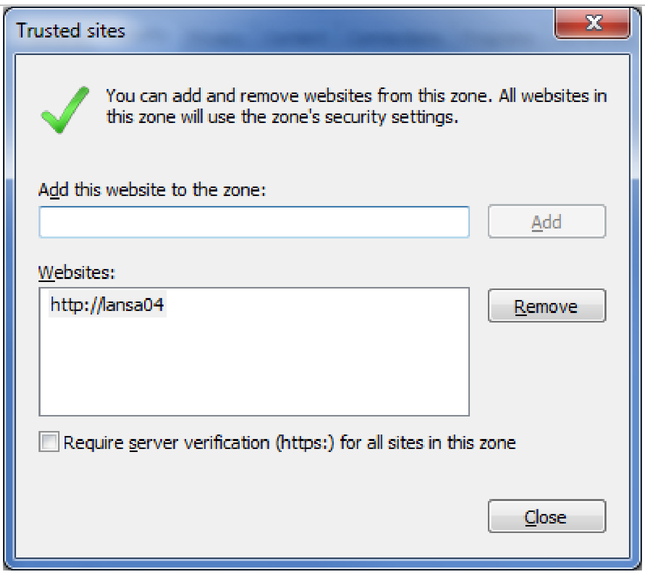
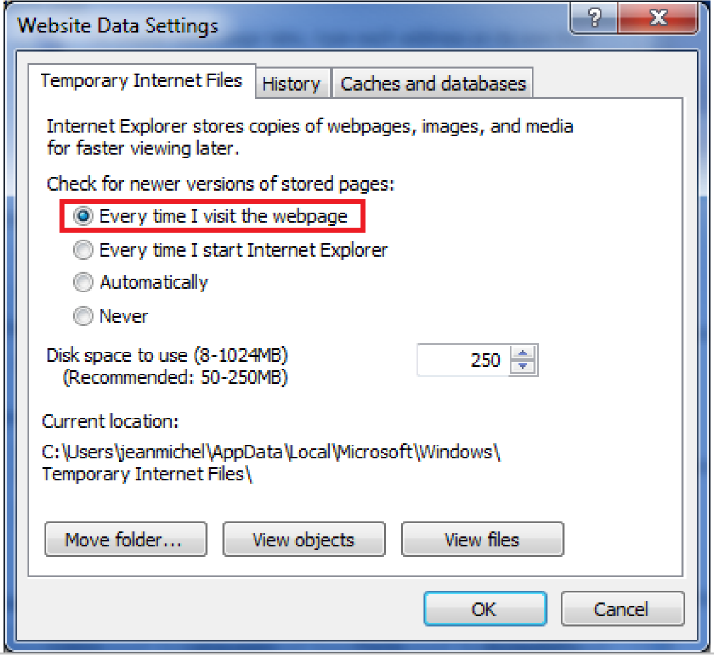
Note 1: You can use extensions on systems below this recommended specification, but your development activities will be impacted accordingly. You should consider upgrading your development system.

Note 2: There are productivity and ease of use benefits to be gained by using a wide screen monitor or multiple side-by-side monitors when doing *any* type of IT application development. Any form of IT application development on a low resolution monitor (eg: 1024x768) will take longer and be more cumbersome than it needs to be.

Check your Internet Explorer Settings

To do aXes application development you will use Internet Explorer.
Complete this Internet Explorer checklist:

Check	Okay
Check that Internet Explorer's SmartScreen Filter is turned off. Use the *Safety* button to view the status of your Internet Explorer SmartScreen Filter. If it is turned on then turn it off. This off setting is strongly recommended for application developers to avoid Internet Explorer failures.	
Check that the aXes server is in your trusted sites list. Open your browser. Use *Tools -> Internet Options -> Security* tab. Select *Trusted sites*, and click the *Sites* button. In this example http://lansa04 is the aXes server root address, so it was added:	

 You can add and remove websites from this zone. All websites in this zone will use the zone's security settings. Add this website to the zone: Websites: http://lansa04 □ Require server verification (https:) for all sites in this zone	
In order to see the changes you make immediately, ensure that you are not using cached pages. To do this, click on the Settings button in the Browsing history section of the General tab, and then select the option to check for newer versions of stored pages *Every time I visit the webpage:*  Website Data Settings Temporary Internet Files History Caches and databases Move folder... View objects View files OK Cancel Internet Explorer stores copies of webpages, images, and media for faster viewing later. Check for newer versions of stored pages: ● Every time I visit the webpage ○ Every time I start Internet Explorer ○ Automatically ○ Never Disk space to use (8-1024MB) (Recommended: 50-250MB) 250 Current location: C:\Users\jeanmichel\AppData\Local\Microsoft\Windows\Temporary Internet Files\	
Check that any pop up blocker(s) you have installed do not block pop ups initiated by scripts loaded from your aXes server.	

Check that library AXESDEMO is installed and useable

To complete the eXtension tutorials you must have library AXESDEMO installed on your system.

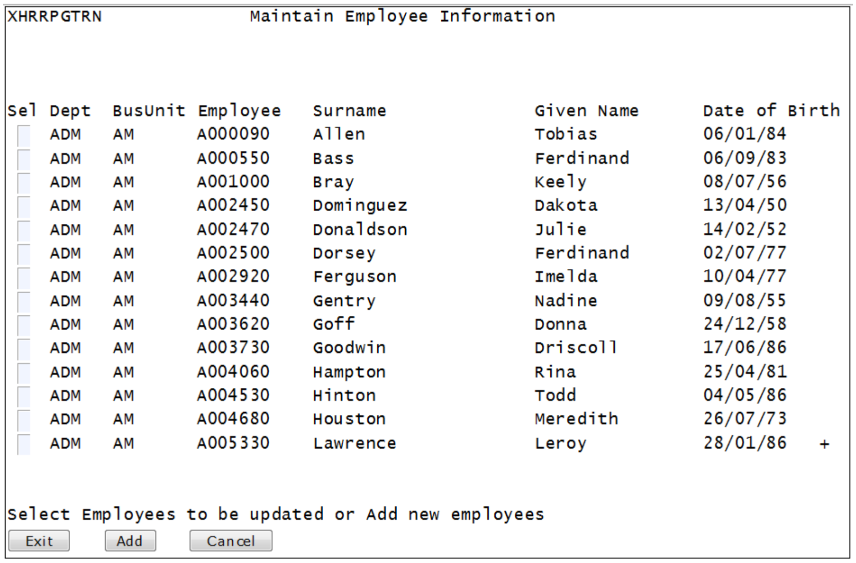
The Axes demonstration material is installed by option 99 during an Axes install.

Perform this check before starting any eXtension tutorials:

Check	Okay
Sign on to a 5250 screen	

ADDLIBLE AXESDEMO	
CALL XHRRPGTRN	

The resulting 5250 display should look something like this:



```
XHRRPGTRN                Maintain Employee Information



Sel Dept  BusUnit Employee   Surname                Given Name       Date of Birth
    ADM   AM      A000090    Allen                  Tobias           06/01/84
    ADM   AM      A000550    Bass                   Ferdinand        06/09/83
    ADM   AM      A001000    Bray                   Keely            08/07/56
    ADM   AM      A002450    Dominguez              Dakota           13/04/50
    ADM   AM      A002470    Donaldson              Julie            14/02/52
    ADM   AM      A002500    Dorsey                 Ferdinand        02/07/77
    ADM   AM      A002920    Ferguson               Imelda           10/04/77
    ADM   AM      A003440    Gentry                 Nadine           09/08/55
    ADM   AM      A003620    Goff                   Donna            24/12/58
    ADM   AM      A003730    Goodwin                Driscoll         17/06/86
    ADM   AM      A004060    Hampton                Rina             25/04/81
    ADM   AM      A004530    Hinton                 Todd             04/05/86
    ADM   AM      A004680    Houston                Meredith         26/07/73
    ADM   AM      A005330    Lawrence               Leroy            28/01/86    +


Select Employees to be updated or Add new employees
```

The eXtension tutorials will use these objects from the AXESDEMO library:

Object	Type	Comment
XHRRPGTRN	*PGM	5250 Employee Maintenance Program
XHRRPGTRND	*DSPF	Display file used by program XHRRPGTRN
XHRBU	*PF	Physical file of Business Units
XHRBU01	*LF	View of Business Units
XHRDEPT	*PF	Physical file of Departments
XHREMPTN	*PF	Physical file of Employees
XHREMPTN01	*LF	View of Employees
XHREMPTN02	*LF	View of Employees

Check your aXes Server Settings (aXes upgrades only)

Do not check this if you have installed a brand new aXes system.

Do check this if your aXes system has been upgraded from 1.32 (or earlier).

Locate your aXes W3 configuration file.

Typically it is located in folder **axes/configs** and named **aXesW3.conf**.

Edit the file and locate the PostLimit directive. It may look like this:

```
# The maximum allowed size of data POSTed to the transaction server
# PostLimit=5000
```

Change the line to be like this:

```
# The maximum allowed size of data POSTed to the transaction server
PostLimit=12000
```

Note that the PostLimit directive is no longer a comment and it has been changed to 12000.

If you need to change this file, save the changes and stop and restart your aXes server instances so that the change takes effect.

Set up an eXtensions Project

Create a new short cut onto your workstation's desktop:

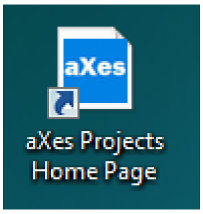
Location	http://<your aXes host>/ts/dev/index.html
Name	aXes Projects Home Page

replacing <your aXes host> with the appropriate IP address and port number.

For example:

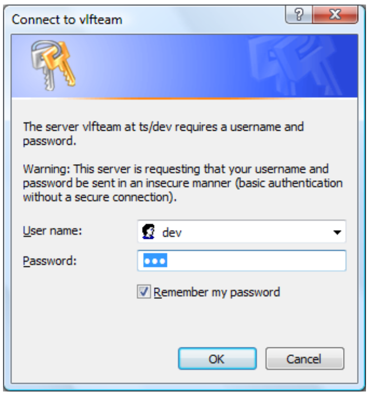
```
http://lansa08:8310/ts/dev/index.html
```

It should appear on your desktop like this:



Use the new shortcut to open the aXes **Projects Home Page**.

When requested, enter the aXes developer user profile and password.



The server vlfteam at ts/dev requires a username and password.

Warning: This server is requesting that your username and password be sent in an insecure manner (basic authentication without a secure connection).

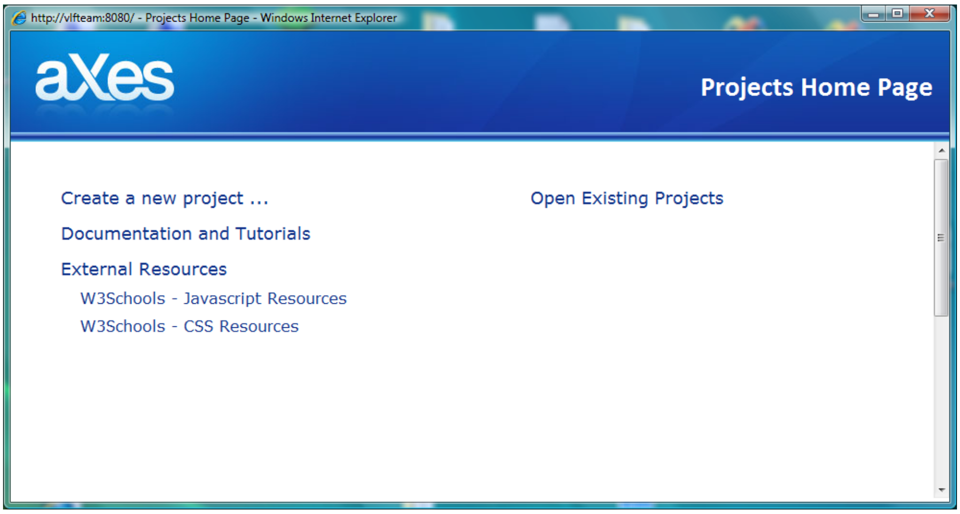
User name: dev

Password: •••

☑ Remember my password

Note: The shipped defaults are User Name: dev, Password: dev
You should change these default values.

The aXes **Projects Home Page** should now be displayed, something like this:



http://vlfteam:8080/ - Projects Home Page - Windows Internet Explorer

Projects Home Page

Create a new project ...

Documentation and Tutorials

External Resources

W3Schools - Javascript Resources

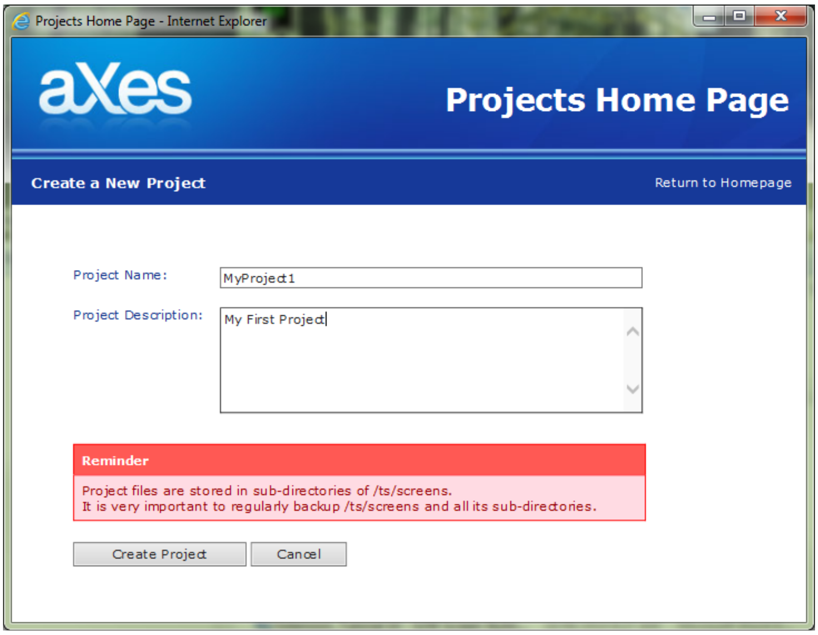
W3Schools - CSS Resources

Open Existing Projects

Decide on a unique name for your aXes project.

> Note: Your chosen project name should be short and consist of only letters from the English alphabet and numbers. For example, MyProject1 or PaulProject2 would be good names. If existing projects are shown in the list on the right, use a name for your new project that is not already shown in the list.

Click on the ***Create a new Project ...*** option.
The resulting display should look like this:



Projects Home Page - Internet Explorer

Projects Home Page

Create a New Project

Return to Homepage

Project Name: MyProject1

Project Description: My First Project

Reminder

Project files are stored in sub-directories of /ts/screens.
It is very important to regularly backup /ts/screens and all its sub-directories.

Create Project

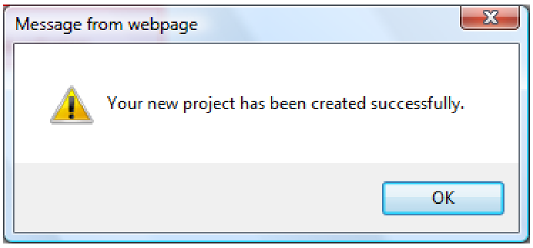
Cancel

Note the warning about backing up your IFS project folders.

Enter a name and description for your project.

Then click the ***Create Project*** button.

A message like this should appear:



Your new project has been created successfully.

Close the Projects Home Page now.

> Note: While learning to use eXtension is it common to have a private or personal project. However, when working on a real project you should create just a single project that all the developers on the project work within. Projects are discrete entities and cannot be merged.

Please start Tutorial 1 now.

aXes® Documentation Library

eXtensions Tutorial 1 - Screen and Field Identification

Objectives

In this tutorial you will learn about screen signatures, how to identify and name screens, and define names for fields.

Prerequisites

Getting Started tutorial

Before attempting this tutorial please complete all of the steps in the Getting Started tutorial.

Outline

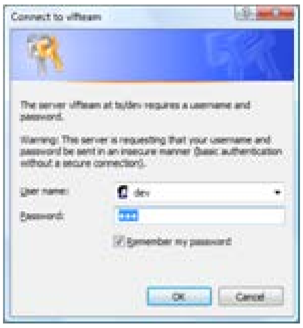
Screen identification – screens created without DDS

Screen identification – screens created using DDS

Screen identification - Screens created without DDS

Use your desktop shortcut to open the aXes Projects Home Page.

When requested, enter the aXes developer user profile and password.



Connect to vfteam

The server vfteam at ts/dev requires a username and password.

Warning: This server is requesting that your username and password be sent in an insecure manner (basic authentication without a secure connection).

User name: dev

Password: •••

☑ Remember my password

The shipped defaults are,

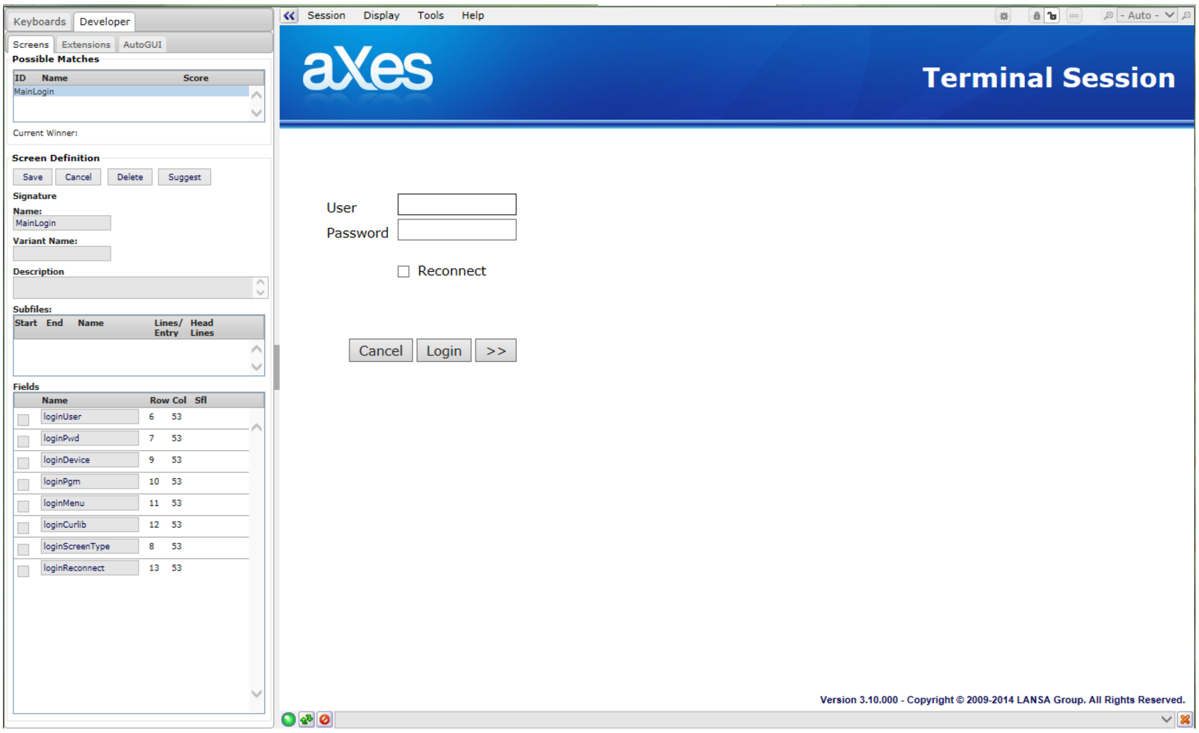
User Name: dev, Password: dev

You should change these default values.

On the **Projects Home Page**, click on your project in the projects list on the right.

On the resulting screen, click on the **Work as TS2 Developer** option.

A 5250 Terminal Session sign on prompt should appear next:



Possible Matches

ID	Name	Score
	MainLogin	

Current Winner:

Screen Definition

Signature

Name: MainLogin

Variant Name:

Description

Subfiles:

Start	End	Name	Lines/ Entry	Head Lines

Fields

	Name	Row	Col	Sfl
□	loginUser	6	53	
□	loginPwd	7	53	
□	loginDevice	9	53	
□	loginPgm	10	53	
□	loginMenu	11	53	
□	loginCurlib	12	53	
□	loginScreenType	8	53	
□	loginReconnect	13	53	

User

Password

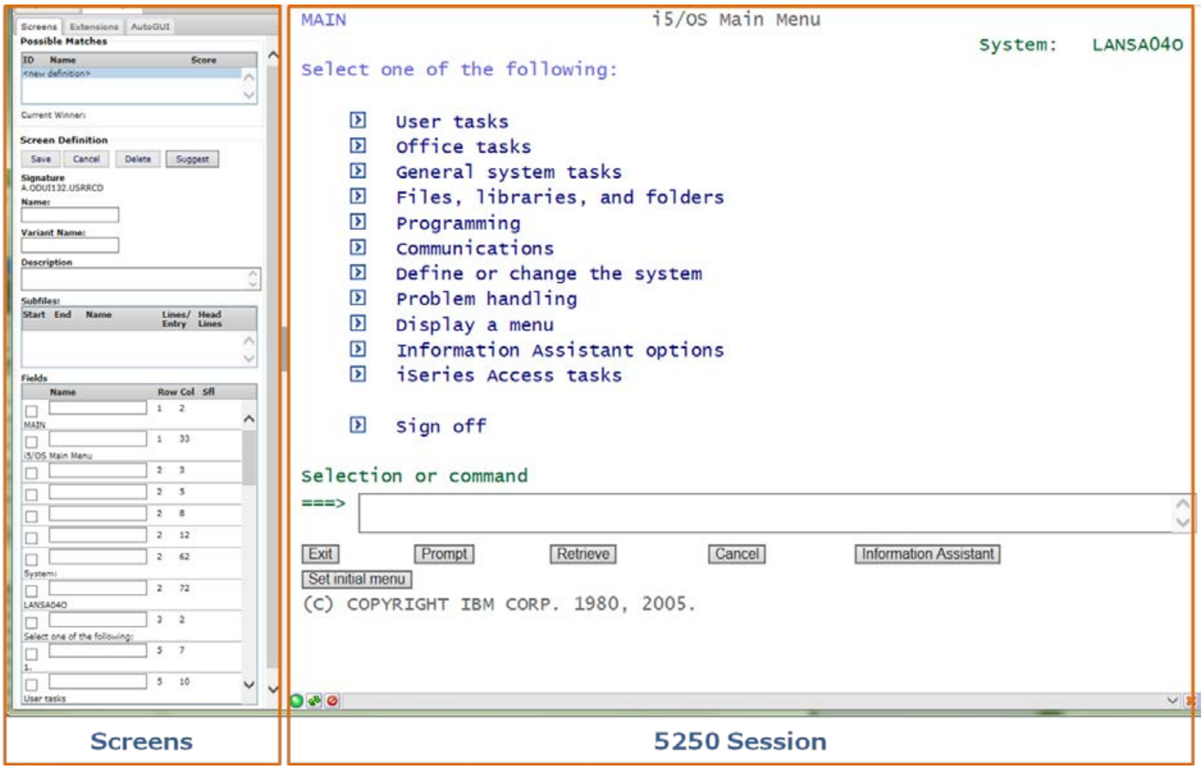
□ Reconnect

Version 3.10.000 - Copyright © 2009-2014 LANSA Group. All Rights Reserved.

Sign on with an IBM i user profile and password that displays the Main Menu.

> When identifying 5250 screens for subsequent use in an aXes-Cloud environment ensure you are executing them in the same way that your end users will.
>
> Refer to the "If You Use aXes-eXtensions with aXes-Cloud" section in the Best Practices tutorial for an explanation of the reasons this is important.

The next screen should look something like this:



ID	Name	Score
	<new definition>	

Signature
A.ODUI132.USRRCD

Name:

Variant Name:

Description

Subfiles:

Start	End	Name	Lines/ Entry	Head Lines

Fields

	Name	Row	Col	Sfl
□ MAIN		1	2	
□ i5/OS Main Menu		1	33	
□		2	3	
□		2	5	
□		2	8	
□		2	12	
□ System:		2	62	
□ LANSA040		2	72	
□ Select one of the following:		3	2	
□ 1.		5	7	
□ User tasks		5	10	

```
MAIN                               i5/OS Main Menu
                                                             System:    LANSA040
Select one of the following:

     User tasks
     Office tasks
     General system tasks
     Files, libraries, and folders
     Programming
     Communications
     Define or change the system
     Problem handling
     Display a menu
     Information Assistant options
     iSeries Access tasks

     Sign off

Selection or command
===>

(C) COPYRIGHT IBM CORP. 1980, 2005.
```

On the left side is a tab titled **Screens** - this is used to identify 5250 screens and the fields on them.

On the right side is a **5250 session** you can use just like any normal 5250 session.

> Hint
>
> 5250 sessions used when logged on as an aXes developer are much slower than normal aXes user 5250 sessions.

> Warning
>
> You should only ever use aXes developer 5250 sessions for aXes development work.
>
> For other development work, like writing RPG programs, use normal user aXes 5250 sessions.
>
> You should never assess the performance of your application using an aXes developer session.

You are now going to identify the Main Menu screen and assign a name to the field on the screen where commands are typed in.

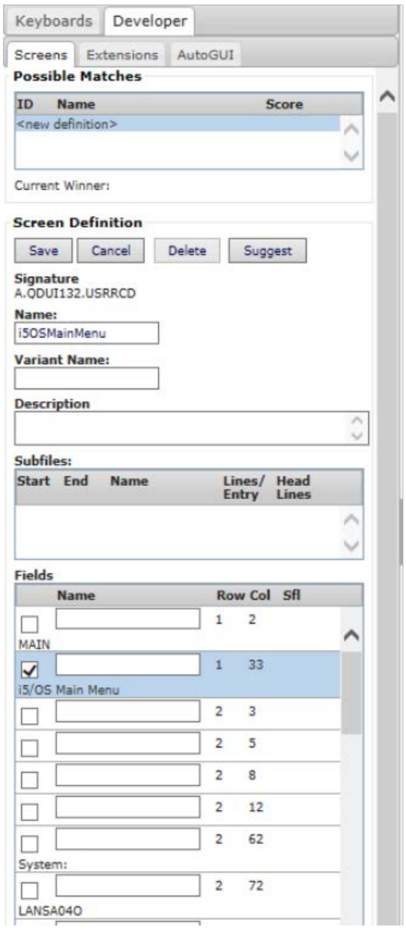
First, click the **Suggest** button on the Screens tab - aXes will examine the screen and suggest a name and a description.

Next, lock the screen for design by using the 🔒🔓 switch in the top right corner of the screen. Always lock the screen for design before customising it.

Then Click on the screen title on the 5250 screen to select it (e.g. IBM i Main Menu, System i Main Menu or i5/OS Main Menu). This should highlight the field in the Screens tab on the left.

Check the box beside the field.

Your Screens tab should now look like this:



Possible Matches

ID	Name	Score
	<new definition>	

Current Winner:

Screen Definition

Signature
A.QDUI132.USRRCD

Name:
i5OSMainMenu

Variant Name:

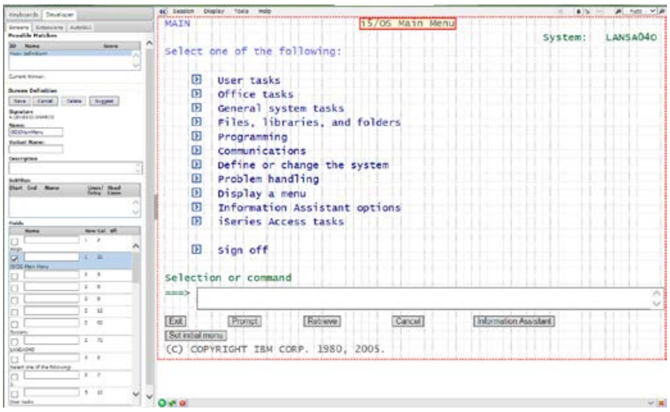
Description

Subfiles:

Start	End	Name	Lines/ Entry	Head Lines

Fields

	Name	Row	Col	Sfl
□ MAIN		1	2	
☑ i5/OS Main Menu		1	33	
□		2	3	
□		2	5	
□		2	8	
□		2	12	
□ System:		2	62	
□ LANSA040		2	72	



```
MAIN                         i5/OS Main Menu
                                                    System:   LANSA040
Select one of the following:

   User tasks
   Office tasks
   General system tasks
   Files, libraries, and folders
   Programming
   Communications
   Define or change the system
   Problem handling
   Display a menu
   Information Assistant options
   iSeries Access tasks

   Sign off

Selection or command
===>
```

(C) COPYRIGHT IBM CORP. 1980, 2005.

Change the screen **Name** to **MAIN**.

Naming Conventions

Choose names for screens and screen elements like you would name a variable in a programming language.

Use only letters of the English alphabet, the numbers 0 to 9 and _ (underscore).

Do not use imbedded blanks in names.

Names are case sensitive.

Change the screen **Description** to IBM i Main Menu.

Click the **Save** button to save the definition of the screen you have called MAIN.

To check that aXes now identifies the MAIN screen correctly, display a different 5250 screen by entering the IBM i command **WRKJOB**.

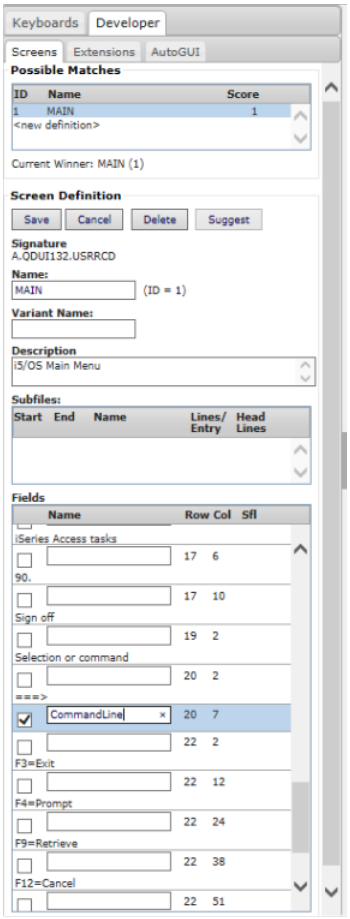
You should see the **Possible Matches** list on the Screens tab change to say **<new definition>** indicating that the Work Job screen currently has no name.

Return to the main menu and you should see the **Possible Matches** list change to indicate that this screen name is **MAIN**.

With the MAIN screen displayed, select the field where commands are entered by clicking on it. This should automatically select the field on the Screens tab and scroll down to it.

Type in the name **CommandLine** for the field.

Your Screens tab should now look like this:



Possible Matches

ID	Name	Score
1	MAIN	1
	<new definition>	

Current Winner: MAIN (1)

Screen Definition

Signature
A.QDUI132.USRRCD

Name:
MAIN (ID = 1)

Variant Name:

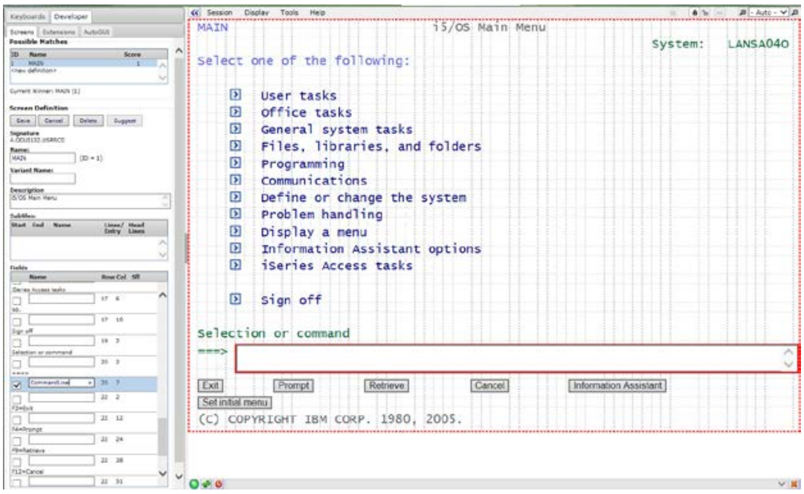
Description
i5/OS Main Menu

Subfiles:

Start	End	Name	Lines/ Entry	Head Lines

Fields

	Name	Row	Col	Sfl
	iSeries Access tasks			
□		17	6	
	90.			
□		17	10	
	Sign off			
□		19	2	
	Selection or command			
□		20	2	
	===>			
☑	CommandLine	20	7	
□		22	2	
	F3=Exit			
□		22	12	
	F4=Prompt			
□		22	24	
	F9=Retrieve			
□		22	38	
	F12=Cancel			
□		22	51	



```
MAIN                            i5/OS Main Menu
                                                          System:   LANSA040
Select one of the following:

     User tasks
     Office tasks
     General system tasks
     Files, libraries, and folders
     Programming
     Communications
     Define or change the system
     Problem handling
     Display a menu
     Information Assistant options
     iSeries Access tasks

     Sign off

Selection or command
===>

(C) COPYRIGHT IBM CORP. 1980, 2005.
```

Click the Save button again to (re)save the definition of the screen called MAIN.

> Hint
>
> Before starting a real project you should create a naming standard for screens and fields.

You have now completed these tasks:

1. Identified the Main Menu screen and assigned it the name MAIN.

2. Named the command input field as CommandLine.

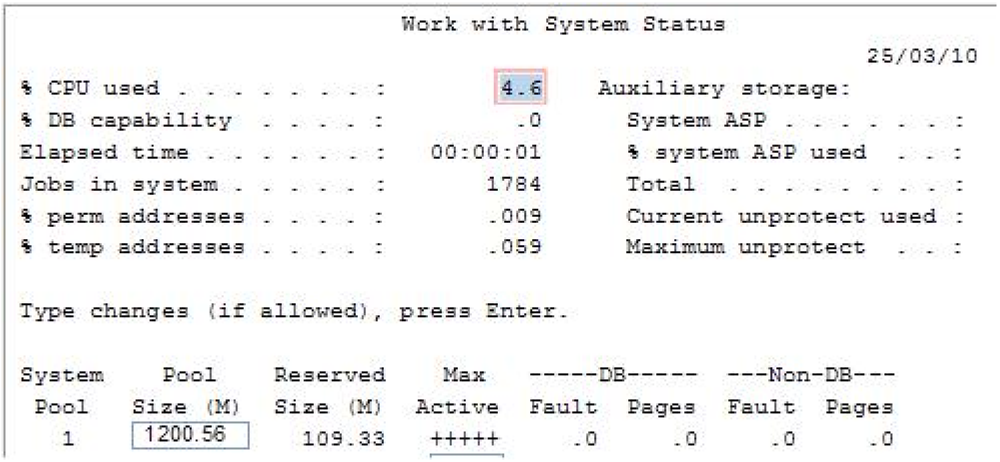
The Output Field Dilemma

When a field is not really a field - it's just a bit of output text

Usually, output fields are correctly recognised as starting at the beginning of the field. This means that when named, they continue to be recognised, no matter what value they contain.

But in one special case, when the record format does not define the fields, (as in for example the "fields" on the WRKSYSSTS screen), aXes works out the field start position based on the field contents. For right-adjusted numeric fields, this means that when the screen is redisplayed with a value with more digits, the start position of the "field" changes, and since aXes identifies fields by their start position, it does not recognize the text as belonging to the same field.

You should be aware of this special case when naming fields:



```
                         Work with System Status
                                                              25/03/10
% CPU used . . . . . . . . :         4.6    Auxiliary storage:
% DB capability  . . . . . :          .0      System ASP . . . . . . . :
Elapsed time . . . . . . . :   00:00:01      % system ASP used  . . . :
Jobs in system . . . . . . :        1784      Total  . . . . . . . . . :
% perm addresses . . . . . :        .009      Current unprotect used :
% temp addresses . . . . . :        .059      Maximum unprotect  . . :

Type changes (if allowed), press Enter.

System    Pool    Reserved    Max   -----DB-----  ---Non-DB---
 Pool   Size (M)  Size (M)  Active  Fault  Pages  Fault  Pages
   1    1200.56    109.33   +++++      .0     .0     .0     .0
```

Screen identification - Screens created using DDS

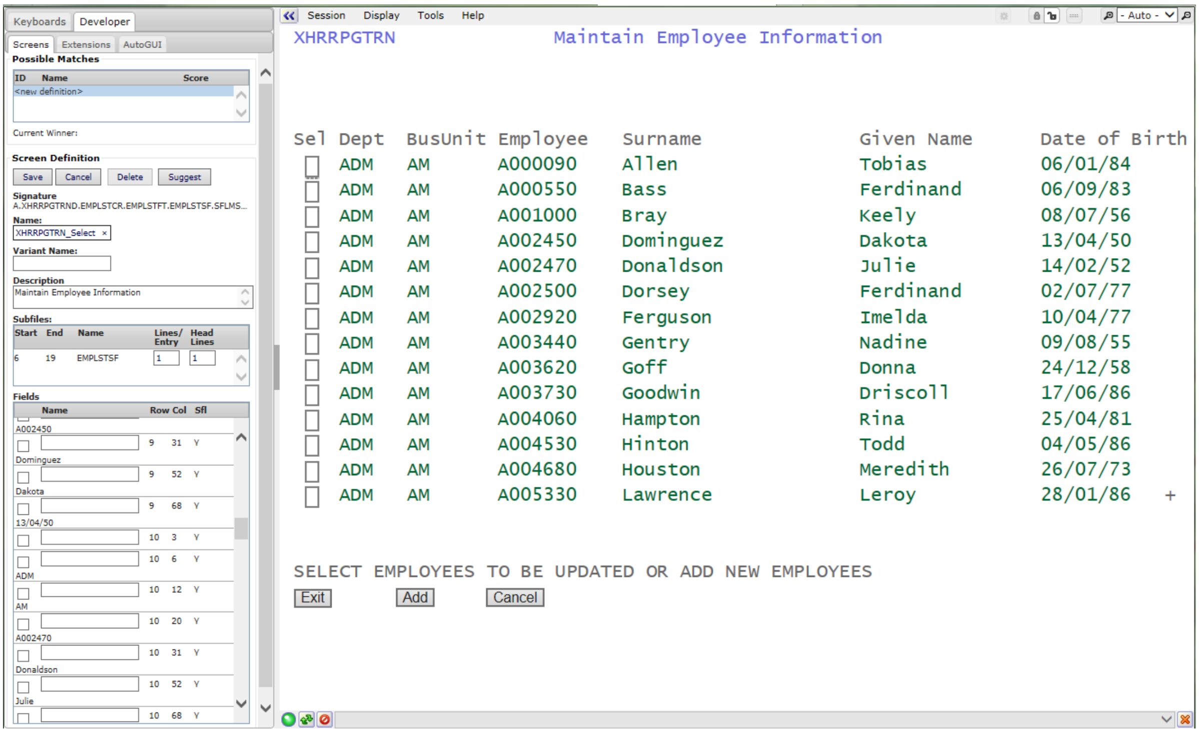
Next you are going to identify two application screens created by normal DDS.

To do this start the shipped aXes demo system from a command line by entering the commands:

ADDLIBLE AXESDEMO

CALL XHRRPGTRN

The resulting 5250 screen should look something like this:



Possible Matches

ID	Name	Score
<new definition>		

Current Winner:

Screen Definition

Signature
A.XHRRPGTRND.EMPLSTCR.EMPLSTFT.EMPLSTSF.SFLMS...

Name:
XHRRPGTRN_Select

Variant Name:

Description
Maintain Employee Information

Subfiles:

Start	End	Name	Lines/ Entry	Head Lines
6	19	EMPLSTSF	1	1

Fields

	Name	Row	Col	Sfl
A002450				
☐		9	31	Y
Dominguez				
☐		9	52	Y
Dakota				
☐		9	68	Y
13/04/50				
☐		10	3	Y
☐		10	6	Y
ADM				
☐		10	12	Y
AM				
☐		10	20	Y
A002470				
☐		10	31	Y
Donaldson				
☐		10	52	Y
Julie				
☐		10	68	Y

```
XHRRPGTRN                Maintain Employee Information


Sel Dept  BusUnit Employee  Surname              Given Name     Date of Birth
 ☐  ADM   AM      A000090   Allen                Tobias         06/01/84
 ☐  ADM   AM      A000550   Bass                 Ferdinand      06/09/83
 ☐  ADM   AM      A001000   Bray                 Keely          08/07/56
 ☐  ADM   AM      A002450   Dominguez            Dakota         13/04/50
 ☐  ADM   AM      A002470   Donaldson            Julie          14/02/52
 ☐  ADM   AM      A002500   Dorsey               Ferdinand      02/07/77
 ☐  ADM   AM      A002920   Ferguson             Imelda         10/04/77
 ☐  ADM   AM      A003440   Gentry               Nadine         09/08/55
 ☐  ADM   AM      A003620   Goff                 Donna          24/12/58
 ☐  ADM   AM      A003730   Goodwin              Driscoll       17/06/86
 ☐  ADM   AM      A004060   Hampton              Rina           25/04/81
 ☐  ADM   AM      A004530   Hinton               Todd           04/05/86
 ☐  ADM   AM      A004680   Houston              Meredith       26/07/73
 ☐  ADM   AM      A005330   Lawrence             Leroy          28/01/86  +


SELECT EMPLOYEES TO BE UPDATED OR ADD NEW EMPLOYEES
Exit        Add         Cancel
```

On the Screens tab, click the **Suggest** Button.

Change the suggested screen name to **XHRRPGTRN_Select**

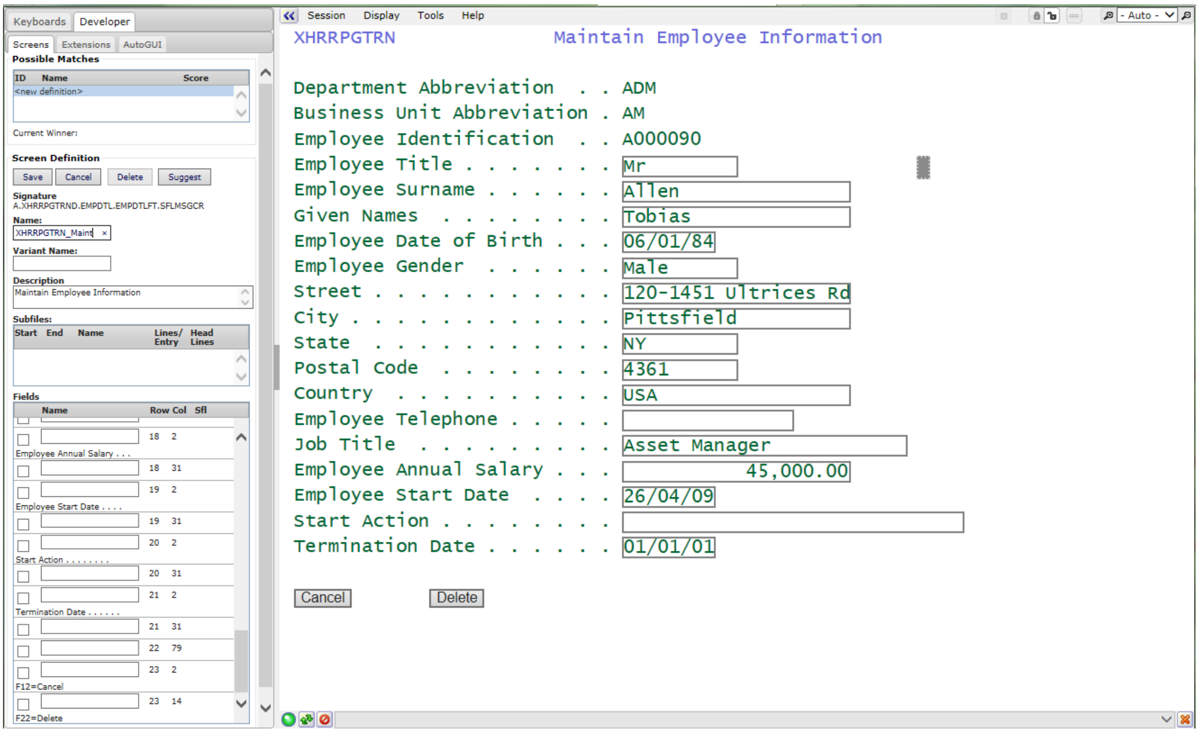
> Remember to choose names for screens and screen elements like you would name a variable in a programming language.
>
> Use only letters of the English alphabet, numbers 0 to 9 and _ (underscore).
>
> Do not use imbedded blanks in names.
>
> Names are case sensitive.

Click the Save button.

You have now assigned the name **XHRRPGTRN_Select** to this particular 5250 screen.

Next, enter an X beside an employee and press the Enter key to display the employee information.

The resulting 5250 screen should look something like this:



Possible Matches

ID	Name	Score
	<new definition>	

Current Winner:

Screen Definition

Signature
A.XHRRPGTRND.EMPDTL.EMPDTLFT.SFLMSGCR

Name: XHRRPGTRN_Maint

Variant Name:

Description: Maintain Employee Information

Subfiles:

Start	End	Name	Lines/ Entry	Head Lines

Fields

	Name	Row	Col	Sfl
□		18	2	
□	Employee Annual Salary . . .	18	31	
□		19	2	
□	Employee Start Date	19	31	
□		20	2	
□	Start Action	20	31	
□		21	2	
□	Termination Date	21	31	
□		22	79	
□		23	2	
□	F12=Cancel	23	14	
	F22=Delete			

```
XHRRPGTRN                 Maintain Employee Information

Department Abbreviation  . . ADM
Business Unit Abbreviation . AM
Employee Identification  . . A000090
Employee Title . . . . . . . Mr
Employee Surname . . . . . . Allen
Given Names  . . . . . . . . Tobias
Employee Date of Birth . . . 06/01/84
Employee Gender  . . . . . . Male
Street . . . . . . . . . . . 120-1451 Ultrices Rd
City . . . . . . . . . . . . Pittsfield
State  . . . . . . . . . . . NY
Postal Code  . . . . . . . . 4361
Country  . . . . . . . . . . USA
Employee Telephone . . . . .
Job Title  . . . . . . . . . Asset Manager
Employee Annual Salary . . .        45,000.00
Employee Start Date  . . . . 26/04/09
Start Action . . . . . . . .
Termination Date . . . . . . 01/01/01

Cancel         Delete
```

Check that your Screens tab says this is a new (unnamed) 5250 screen.

On the Screens tab, click the Suggest Button.

Change the suggested screen name to **XHRRPGTRN_Maint**

Click the Save button.

You have now assigned the name **XHRRPGTRN_Maint** to this particular 5250 screen.

Now, double check that aXes correctly identifies the named screens.

To do this, return to the IBM i main menu, then call program XHRRPGTRN again. Make sure the selection screen and the details screen are correctly identified on the aXes Screens tab as you navigate between the screens.

You have completed this tutorial are now ready to proceed to the next tutorial.

Screen Signatures and the Concept of a "5250 Screen"

Did you notice how you did not have to select screen elements to identify the preceding two screens? Since they were created from DDS they each have a unique "signature". Their unique signatures were enough to identify each screen as being different.

That is an easy rule to remember – a different screen signature means use a different screen name. Unfortunately it's not quite that simple - sometimes screens with different signatures are given the same screen name.

This is usually done when different screen signatures actually represent subtle variations of what is considered to be the same 5250 screen. In these cases the variant name may also be used to identify different variations of the same screen name.

The key question: What is a "5250 screen" exactly?

There is no answer to that question.

Imagine an RPG program, using display file DSP, to display records named HEAD, BODY and FOOT (say) onto a 5250 screen. The visual result of this is said to comprise a 5250 screen named "Screen1" (say).

However, what if under some circumstances, it displays records HEAD, BODY, BODY_EXT and FOOT.

Is the resulting 5250 screen a new screen, or just a variation of "Screen1"?

There is no correct answer to the preceding question, because the thing that is called "Screen1" is really just a concept.

Both possible answers are equally correct - so aXes eXtensions allow you to answer the question either way.

If you decide these are two variations of the same screen you simply say this ...

Signature	Name
DSP+HEAD+BODY+FOOT	"Screen1"

DSP+HEAD+BODY+BODY_EXT+FOOT	"Screen1" with variant name "BODY_EXT"

Or, if you decide they are different screens, you say this ...

Signature	Name
DSP+HEAD+BODY+FOOT	"Screen1"
DSP+HEAD+BODY+BODY_EXT+FOOT	"Screen2"

So what exactly is a "5250 screen"?

It's whatever you want it to be.

aXes® Documentation Library

eXtensions Tutorial 2 - Basic Screen Enhancement

Using aXes Developer

Use your desktop shortcut to open the aXes Projects Home Page.

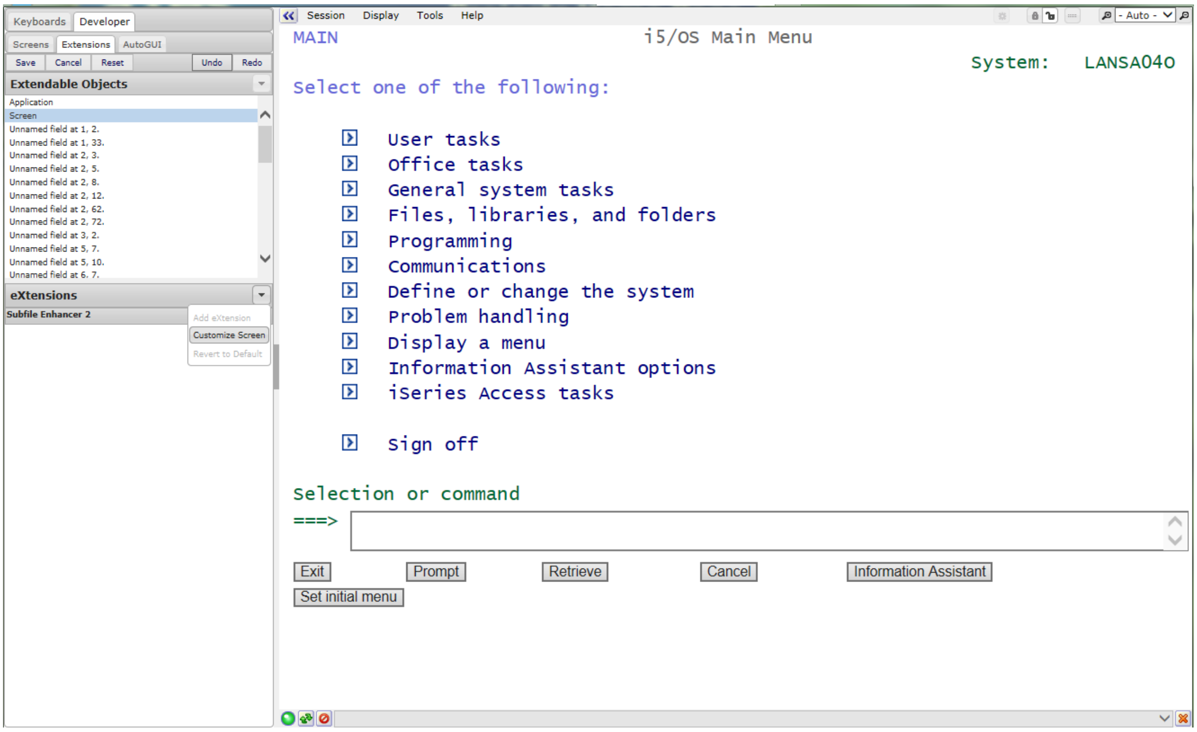
Sign on as an aXes developer and select your project from the list on the right.

On the next screen, click on the **Work as TS2 Developer** option. This will start an aXes developer session.

Objectives	This tutorial teaches you how to customize 5250 screens by hiding screen elements and adding new screen elements using aXes eXtensions.
Prerequisites	Tutorial 1 - Screen and Field Identification Before attempting this tutorial, please complete all of the steps in Tutorial 1.
Outline	Using aXes Developer Simple customization Adding value using customization Adding value using scripting Testing customizations as a user Screen identifier/design concepts

Sign on to a 5250 session and display the Main Menu.

The white background around the open padlock in the tool bar at the top right of the screen panel indicates that the screen is unlocked. The closed pad lock is greyed.



```
MAIN                          i5/OS Main Menu
                                                        System:   LANSA040
Select one of the following:

    User tasks
    Office tasks
    General system tasks
    Files, libraries, and folders
    Programming
    Communications
    Define or change the system
    Problem handling
    Display a menu
    Information Assistant options
    iSeries Access tasks

    Sign off

Selection or command
===>
```

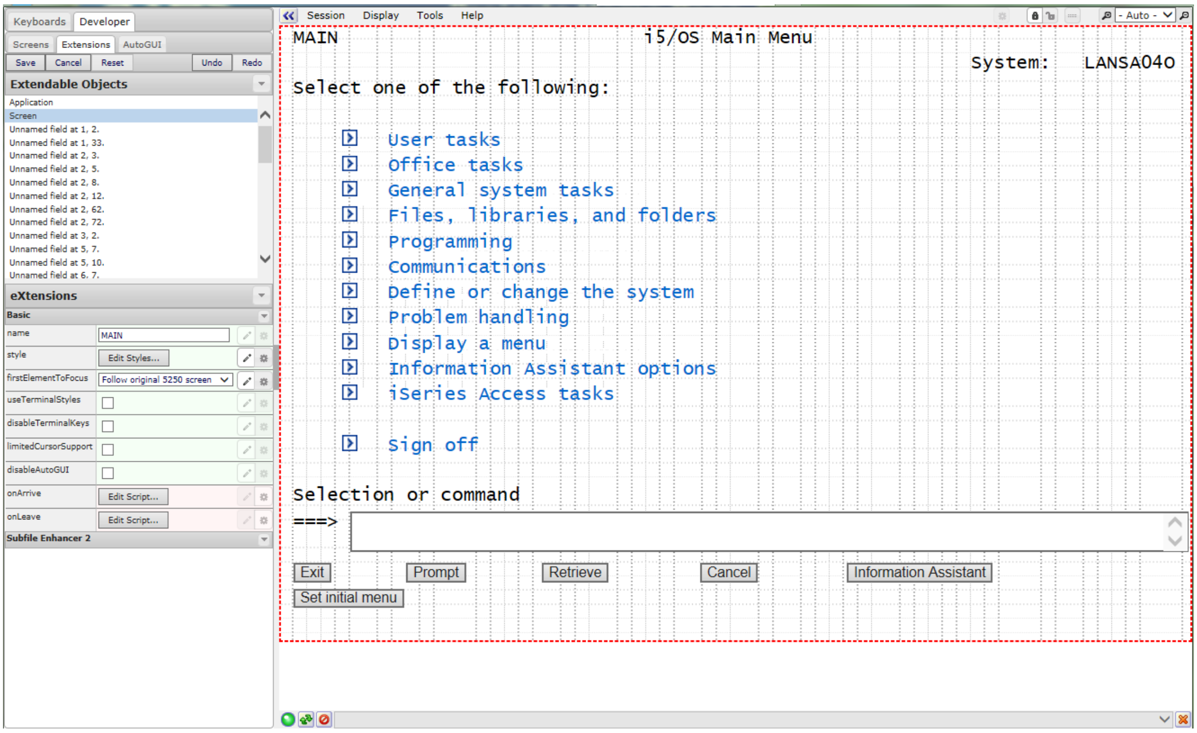
Click on the **eXtensions tab** and then lock the screen by clicking the **closed padlock icon** on the top right of the screen.

Selecting objects to customize

Click the **Screen** object in the **Extendable Objects** list.

Click the **eXtensions action button** (in the eXtensions bar) and choose the **Customize Screen** option from the eXtensions Actions list.

The eXtensions panel shows the basic object properties.



Extendable Objects

- Application
- Screen
- Unnamed field at 1, 2.
- Unnamed field at 1, 33.
- Unnamed field at 2, 3.
- Unnamed field at 2, 5.
- Unnamed field at 2, 8.
- Unnamed field at 2, 12.
- Unnamed field at 2, 62.
- Unnamed field at 2, 72.
- Unnamed field at 3, 2.
- Unnamed field at 5, 7.
- Unnamed field at 5, 10.
- Unnamed field at 6, 7.

eXtensions

Basic

name	MAIN
style	Edit Styles...
firstElementToFocus	Follow original 5250 screen
useTerminalStyles	□
disableTerminalKeys	□
limitedCursorSupport	□
disableAutoGUI	□
onArrive	Edit Script...
onLeave	Edit Script...

Subfile Enhancer 2

```
MAIN                                    i5/OS Main Menu
                                                              System:    LANSA040
Select one of the following:

       User tasks
       Office tasks
       General system tasks
       Files, libraries, and folders
       Programming
       Communications
       Define or change the system
       Problem handling
       Display a menu
       Information Assistant options
       iSeries Access tasks

       Sign off

Selection or command
===>
```

What is the thin red line?

If you look closely at your 5250 screen panel, you may see red, dotted lines around the screen. These lines represent the boundaries of this 5250 screen.

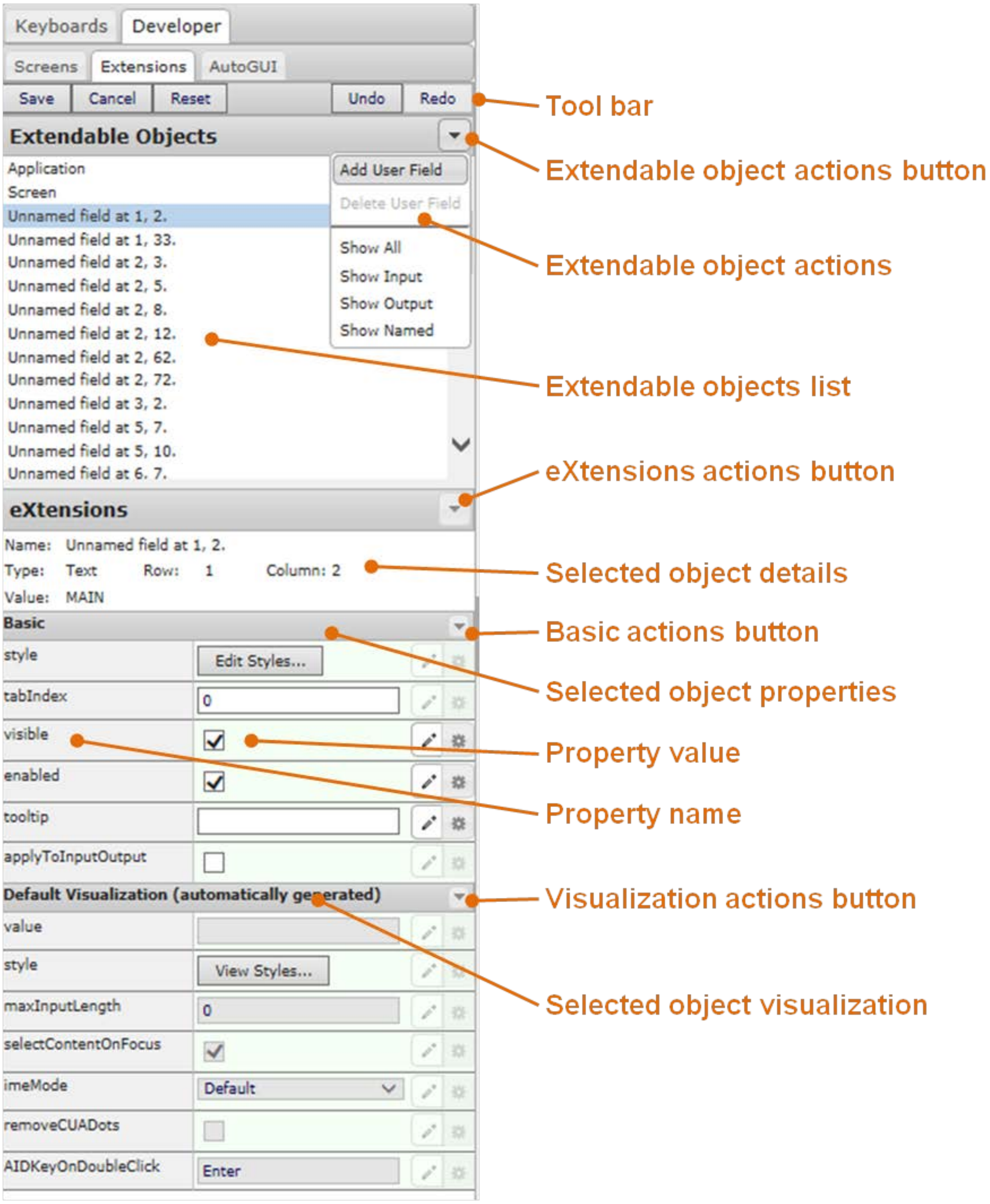
You can move the boundaries to create more space on your 5250 screen.

Tutorial 10 – 5250 Screen Styling explains how to move the screen boundaries.

> Warning: Do not place any screen element outside the screen boundaries.

aXes Developer window

The aXes Developer window changes layout and content depending on the object selection. To understand how this works, it is worth looking at the layout of a typical aXes Developer window in a little more detail.



Name: Unnamed field at 1, 2.
Type: Text Row: 1 Column: 2
Value: MAIN

style	Edit Styles...
tabIndex	0
visible	☑
enabled	☑
tooltip	
applyToInputOutput	□

value	
style	View Styles...
maxInputLength	0
selectContentOnFocus	☑
imeMode	Default
removeCUADots	□
AIDKeyOnDoubleClick	Enter

The body of the aXes Developer window contains a list of screen objects and property sheets. If you have used almost any other visual development tools, you will have seen similar property sheets.

The following table explains the parts of the aXes Developer window.

Basic actions button	The button opens a list of actions available for the basic properties.
Extendable object actions button	The button opens a list of actions available for a selected object.

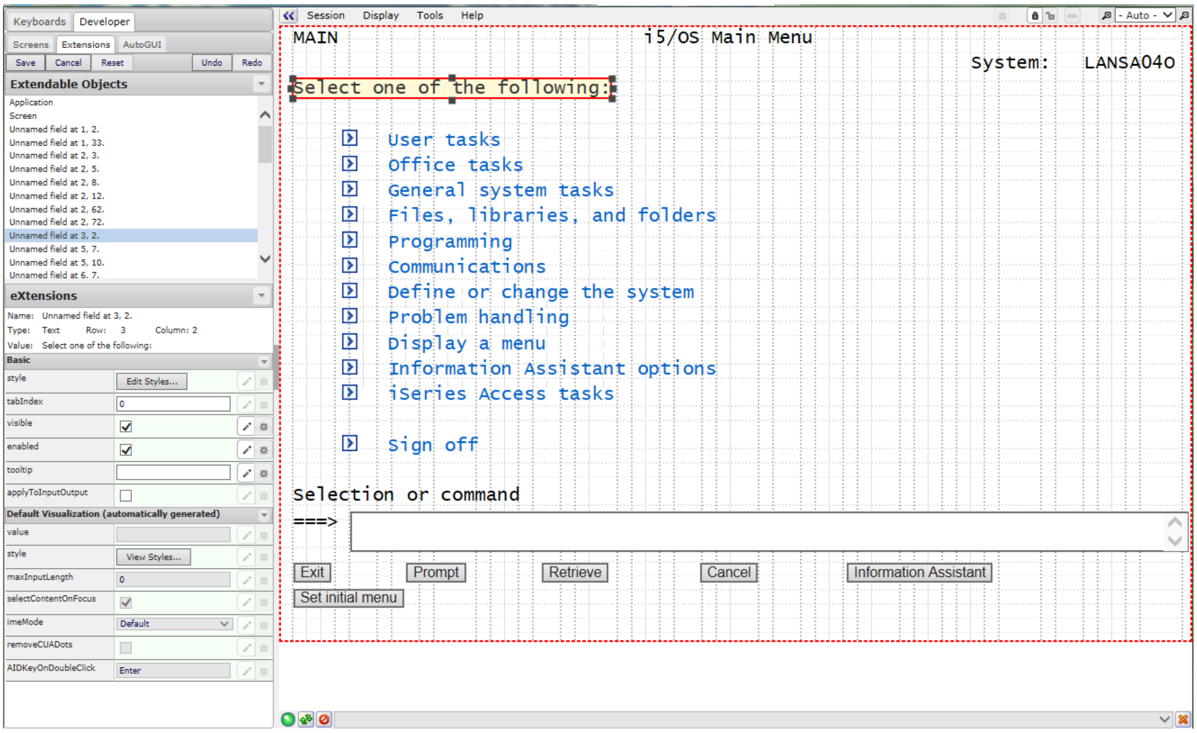
Extendable objects	The Extendable Objects panel contains a list of objects available for customization using extensions.
eXtensions	The eXtensions panel contains details of the selected object, properties and property values for extending the selected object. For example, the Visible property shows or hides the selected object.
eXtensions actions button	The button opens a list of actions available for an extension.
Property name / value	Each property has a name (on the left) and a value (on the right). Change the value (on the right) to change the behaviour of a screen or screen element.
Selected object details	The details describe the object type and placement on the screen.
Selected object properties	eXtensions Basic The Basic panel contains properties defining how selected object behaves.
Selected object visualization	eXtensions Default Visualization The Default Visualization panel contains properties defining the appearance of the selected object.
Tool bar	The tool bar contains buttons including save, cancel, undo, redo and reset.
Visualization actions button	The button opens a list of actions available for the visualization properties.

Simple customization

Hide screen elements

With the Main Menu 5250 screen and the aXes Developer window displayed, click on the text **Select one of the following:** to select it.

Your screen should now look like this:



Name: Unnamed field at 3, 2.
Type: Text Row: 3 Column: 2
Value: Select one of the following:

style	Edit Styles...
tabIndex	0
visible	☑
enabled	☑
tooltip	
applyToInputOutput	□
value	
style	View Styles...
maxInputLength	0
selectContentOnFocus	☑
imeMode	Default
removeCUADots	□
AIDKeyOnDoubleClick	Enter

```
MAIN                                i5/OS Main Menu
                                                             System:   LANSA040
Select one of the following:

     User tasks
     Office tasks
     General system tasks
     Files, libraries, and folders
     Programming
     Communications
     Define or change the system
     Problem handling
     Display a menu
     Information Assistant options
     iSeries Access tasks

     Sign off

Selection or command
===>
```

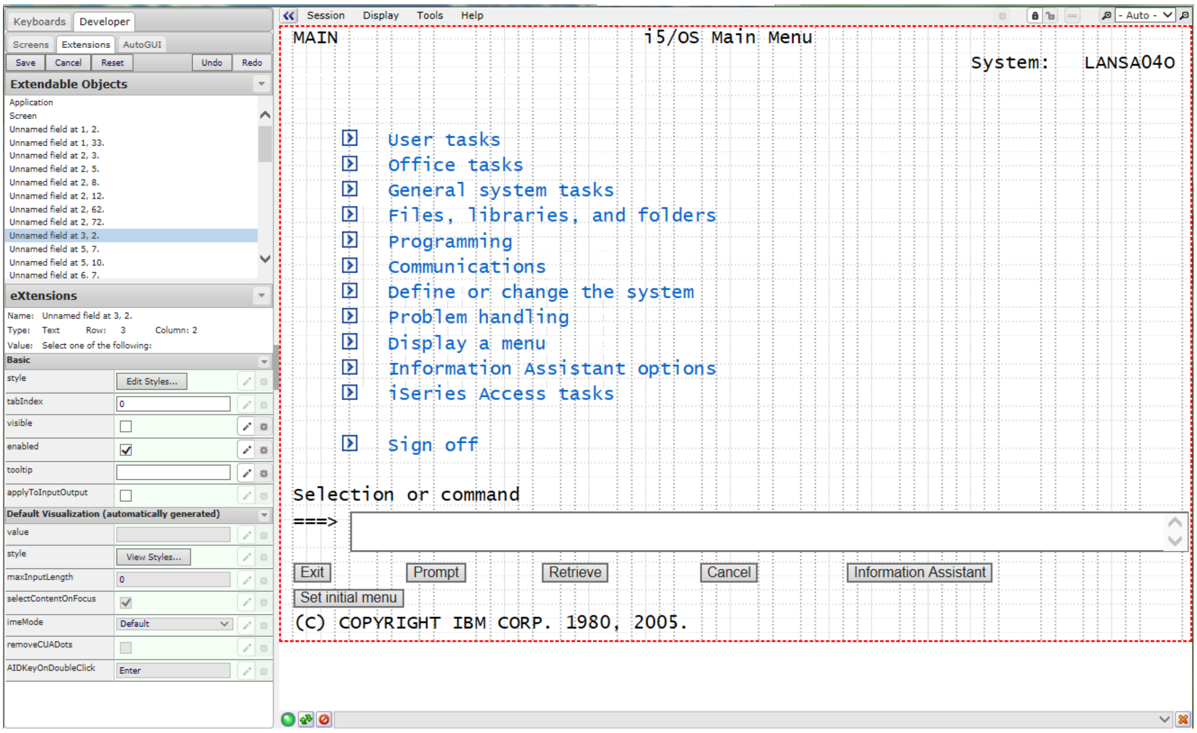
The aXes Developer window shows the selected element, i.e. the output text, "Select one of the following:" inside a selection box with sizing handles.

In the eXtensions area of the aXes Developer window, uncheck the **Visible** property.

Do not remove the default visualisation.

Click the Save button.

The 5250 screen no longer shows the text.



Application
Screen
Unnamed field at 1, 2.
Unnamed field at 1, 33.
Unnamed field at 2, 3.
Unnamed field at 2, 5.
Unnamed field at 2, 8.
Unnamed field at 2, 12.
Unnamed field at 2, 62.
Unnamed field at 2, 72.
Unnamed field at 3, 2.
Unnamed field at 5, 7.
Unnamed field at 5, 10.
Unnamed field at 6, 7.

eXtensions

Name: Unnamed field at 3, 2.
Type: Text Row: 3 Column: 2
Value: Select one of the following:

Basic	
style	Edit Styles...
tabIndex	0
visible	□
enabled	☑
tooltip	
applyToInputOutput	□
Default Visualization (automatically generated)	
value	
style	View Styles...
maxInputLength	0
selectContentOnFocus	☑
imeMode	Default
removeCUADots	□
AIDKeyOnDoubleClick	Enter

```
MAIN                                i5/OS Main Menu
                                                     System:   LANSA040

      User tasks
      Office tasks
      General system tasks
      Files, libraries, and folders
      Programming
      Communications
      Define or change the system
      Problem handling
      Display a menu
      Information Assistant options
      iSeries Access tasks

      Sign off

Selection or command
===>

Exit    Prompt    Retrieve    Cancel    Information Assistant
Set initial menu
(C) COPYRIGHT IBM CORP. 1980, 2005.
```

In this example, aXes does not display the hidden text. Nevertheless, the text remains as a screen element.

Hiding a screen element does not delete the element from the screen.

You have now completed the simplest task of screen enhancement – hiding a screen element.

Screen enhancement techniques

In most cases, enhancing 5250 screens uses three main techniques:

1. Change visualization

   Causing 5250 screen elements be visualized in a different way, e.g. as a check box, drop-down list, hidden, etc.

   Tutorial 3 deals with this area in depth.

2. Add new capabilities

   Add new capabilities to 5250 screen elements, e.g. transforming a date field into a date picker.

   Tutorial 3 also deals with this topic.

3. Introduce new elements

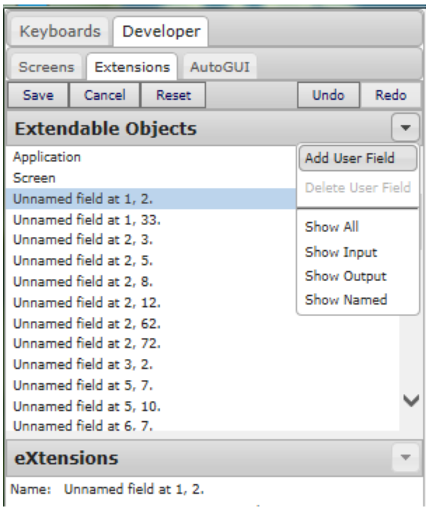
   Add brand new elements to the 5250 screen to improve its appearance or allow it to perform new operations that add value to the underlying application, e.g. adding push buttons and scripts.

   Adding value is the focus of the rest of this tutorial.

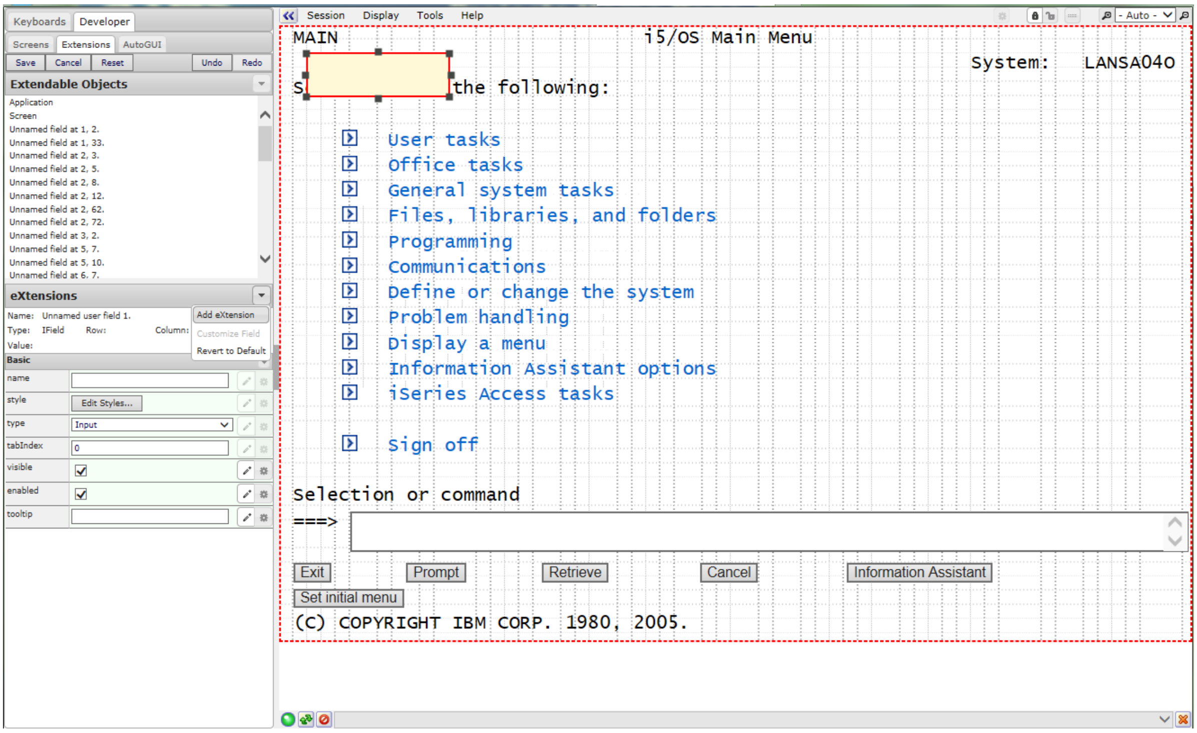
Add a group box

You are now going to add a group box to the MAIN screen.

Click the **Extendable Objects action button** (on the right-hand side of the Extendable Objects bar), then choose **Add User Field** from drop-down list.



A new user field will appear on the screen as a box with sizing handles.



Application
Screen
Unnamed field at 1, 2.
Unnamed field at 1, 33.
Unnamed field at 2, 3.
Unnamed field at 2, 5.
Unnamed field at 2, 8.
Unnamed field at 2, 12.
Unnamed field at 2, 62.
Unnamed field at 2, 72.
Unnamed field at 3, 2.
Unnamed field at 5, 7.
Unnamed field at 5, 10.
Unnamed field at 6, 7.

Name: Unnamed user field 1.
Type: IField Row: Column:
Value:

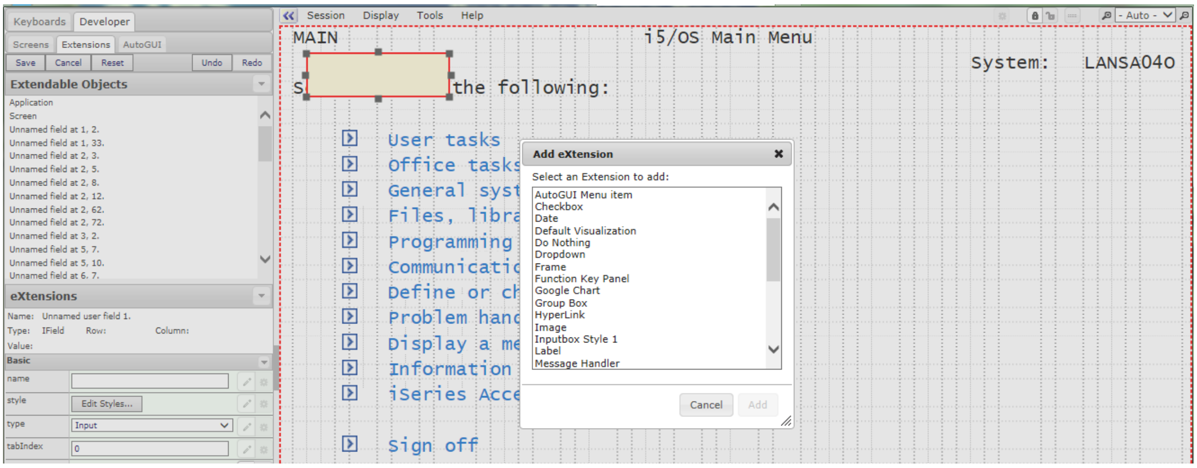
name	
style	Edit Styles...
type	Input
tabIndex	0
visible	☑
enabled	☑
tooltip	

```
MAIN                              i5/OS Main Menu
                                                          System:   LANSA040
S          the following:

     User tasks
     Office tasks
     General system tasks
     Files, libraries, and folders
     Programming
     Communications
     Define or change the system
     Problem handling
     Display a menu
     Information Assistant options
     iSeries Access tasks

     Sign off

Selection or command
===>

(C) COPYRIGHT IBM CORP. 1980, 2005.
```

Next, you define what type of user field you wish to use.

Click the **eXtensions action button** (on the right-hand side of the eXtensions bar) and choose **Add eXtension** from the drop-down list.

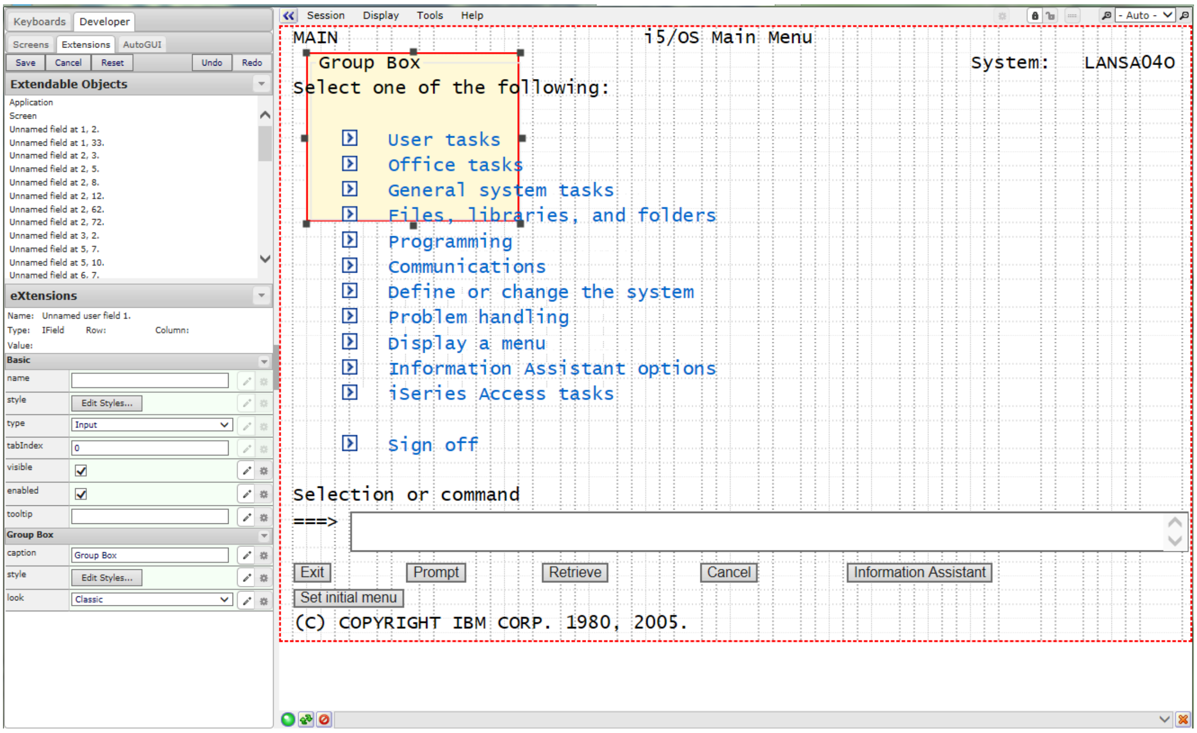
aXes will display a list of eXtensions.



Select an Extension to add:

AutoGUI Menu item
Checkbox
Date
Default Visualization
Do Nothing
Dropdown
Frame
Function Key Panel
Google Chart
Group Box
HyperLink
Image
Inputbox Style 1
Label
Message Handler

Choose **Group Box** from the list and click the **Add button**.



Application
Screen
Unnamed field at 1, 2.
Unnamed field at 1, 33.
Unnamed field at 2, 3.
Unnamed field at 2, 5.
Unnamed field at 2, 8.
Unnamed field at 2, 12.
Unnamed field at 2, 62.
Unnamed field at 2, 72.
Unnamed field at 3, 2.
Unnamed field at 5, 7.
Unnamed field at 5, 10.
Unnamed field at 6, 7.

eXtensions

Name: Unnamed user field 1.
Type: IField Row: Column:
Value:

Basic	
name	
style	Edit Styles...
type	Input
tabIndex	0
visible	☑
enabled	☑
tooltip	
Group Box	
caption	Group Box
style	Edit Styles...
look	Classic

```
MAIN                              i5/OS Main Menu
  Group Box                                                      System:   LANSA040
Select one of the following:

    ▷   User tasks
    ▷   Office tasks
    ▷   General system tasks
    ▷   Files, libraries, and folders
    ▷   Programming
    ▷   Communications
    ▷   Define or change the system
    ▷   Problem handling
    ▷   Display a menu
    ▷   Information Assistant options
    ▷   iSeries Access tasks

    ▷   Sign off

Selection or command
===>

Exit      Prompt      Retrieve      Cancel      Information Assistant
Set initial menu
(C) COPYRIGHT IBM CORP. 1980, 2005.
```

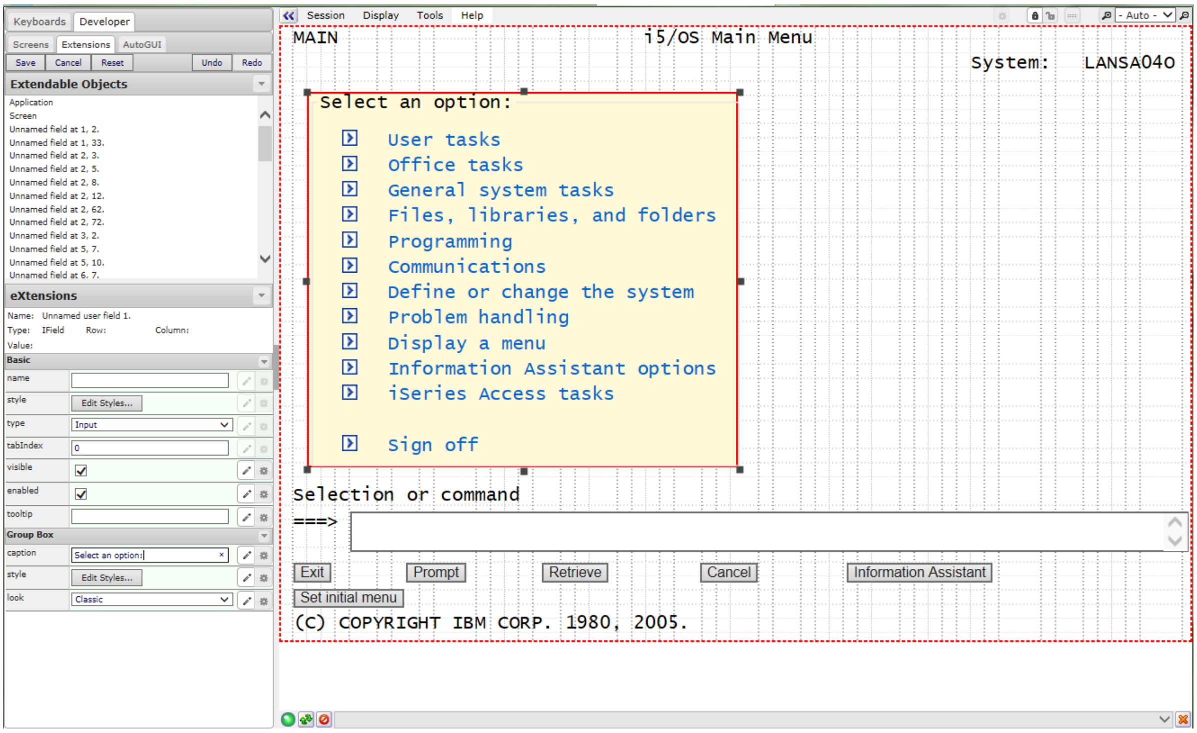
The group box will become larger and include the text, Group Box.

Change its caption property value to **Select an option:**

You can click on and drag the sizing handles to resize the group box.

Enlarge the group box so that it surrounds the menu items.

When you finish, the screen will look similar to this:



Extendable Objects

Application
Screen
Unnamed field at 1, 2.
Unnamed field at 1, 33.
Unnamed field at 2, 3.
Unnamed field at 2, 5.
Unnamed field at 2, 8.
Unnamed field at 2, 12.
Unnamed field at 2, 62.
Unnamed field at 2, 72.
Unnamed field at 3, 2.
Unnamed field at 5, 7.
Unnamed field at 5, 10.
Unnamed field at 6, 7.

eXtensions

Name: Unnamed user field 1.
Type: IField Row: Column:
Value:

Basic	
name	
style	Edit Styles...
type	Input
tabIndex	0
visible	☑
enabled	☑
tooltip	
Group Box	
caption	Select an option:
style	Edit Styles...
look	Classic

```
MAIN                            i5/OS Main Menu
                                                  System:   LANSA040
 Select an option:

  ▣   User tasks
  ▣   Office tasks
  ▣   General system tasks
  ▣   Files, libraries, and folders
  ▣   Programming
  ▣   Communications
  ▣   Define or change the system
  ▣   Problem handling
  ▣   Display a menu
  ▣   Information Assistant options
  ▣   iSeries Access tasks

  ▣   Sign off

Selection or command
===>

(C) COPYRIGHT IBM CORP. 1980, 2005.
```

You have added a group box to customize the MAIN screen.

Using screen customization to add value

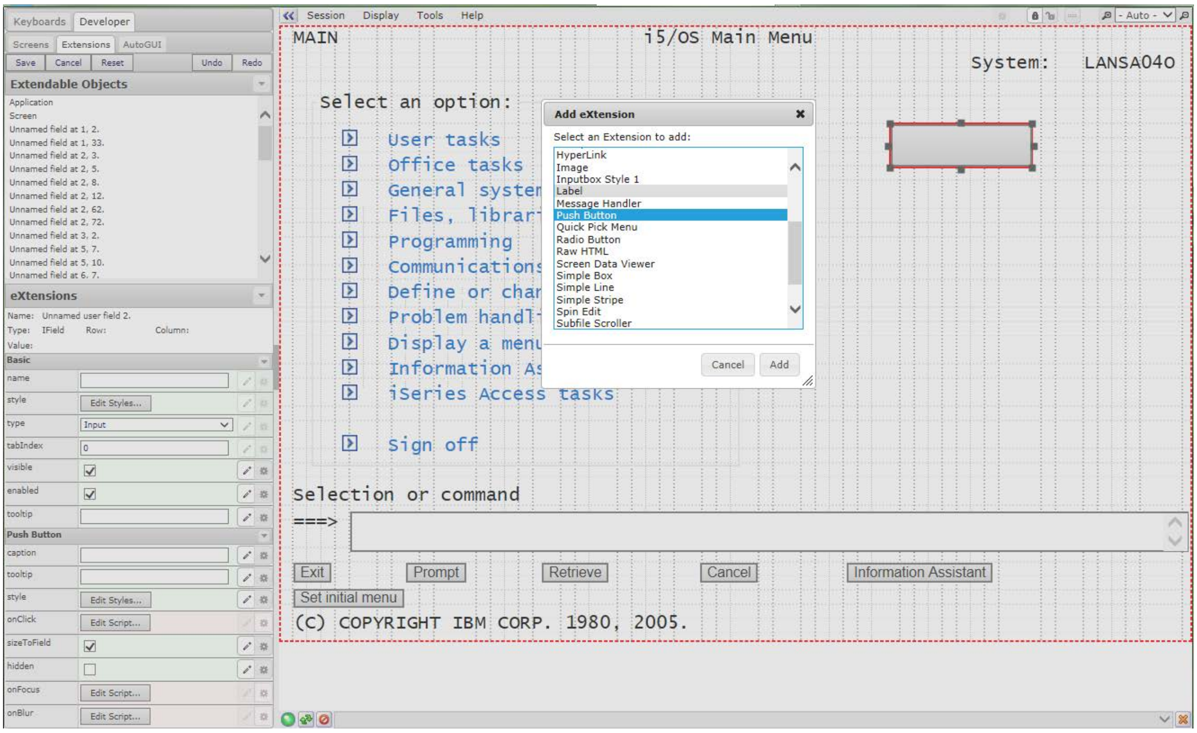
In the preceding section, you added a group box to the MAIN screen. This adds some value to the application from the visual perspective, but adds nothing from the usability perspective.

You are now going to add new buttons to the MAIN screen that add usability value to the 5250 screen.

Add a button

Click the **Extendable Objects** action button, then choose **Add User Field** from drop-down list.

When aXes displays the list of eXtensions, choose **Push Button** and then click the Add button.



Name: Unnamed user field 2.
Type: IField Row: Column:
Value:

Basic	
name	
style	Edit Styles...
type	Input
tabIndex	0
visible	☑
enabled	☑
tooltip	
Push Button	
caption	
tooltip	
style	Edit Styles...
onClick	Edit Script...
sizeToField	☑
hidden	☐
onFocus	Edit Script...
onBlur	Edit Script...

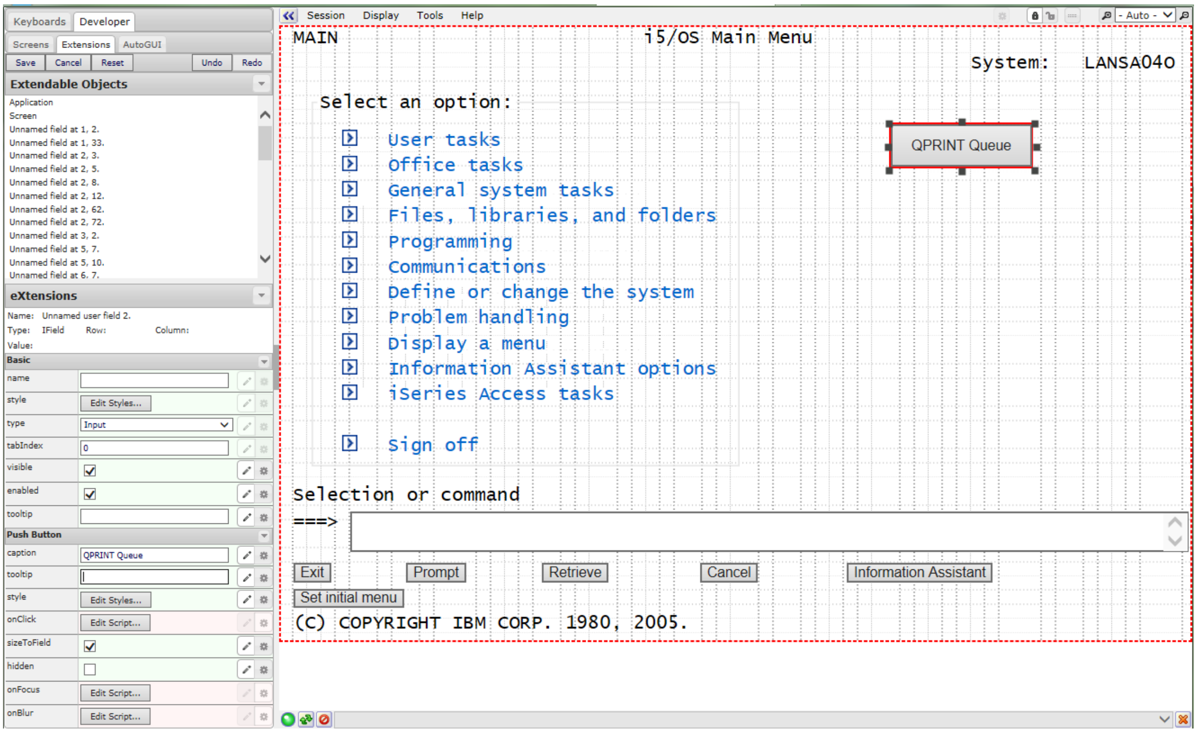
```
MAIN                         i5/OS Main Menu
                                                   System:   LANSA040
 Select an option:

   User tasks
   Office tasks
   General system
   Files, library
   Programming
   Communications
   Define or char
   Problem handl
   Display a menu
   Information As
   iSeries Access tasks

   Sign off

Selection or command
===>

(C) COPYRIGHT IBM CORP. 1980, 2005.
```

Move the new button to the right-hand side of the screen.

Change the button's caption to **QPRINT Queue**.

The caption property is in the lower panel of the aXes Developer window.

The screen should look like this:



Extendable Objects

Application
Screen
Unnamed field at 1, 2.
Unnamed field at 1, 33.
Unnamed field at 2, 3.
Unnamed field at 2, 5.
Unnamed field at 2, 8.
Unnamed field at 2, 12.
Unnamed field at 2, 62.
Unnamed field at 2, 72.
Unnamed field at 3, 2.
Unnamed field at 5, 7.
Unnamed field at 5, 10.
Unnamed field at 6, 7.

eXtensions

Name: Unnamed user field 2.
Type: IField Row: Column:
Value:

Basic	
name	
style	Edit Styles...
type	Input
tabIndex	0
visible	☑
enabled	☑
tooltip	
Push Button	
caption	QPRINT Queue
tooltip	
style	Edit Styles...
onClick	Edit Script...
sizeToField	☑
hidden	☐
onFocus	Edit Script...
onBlur	Edit Script...

```
MAIN                          i5/OS Main Menu
                                                   System:   LANSA040

  Select an option:

     User tasks
     Office tasks
     General system tasks
     Files, libraries, and folders
     Programming
     Communications
     Define or change the system
     Problem handling
     Display a menu
     Information Assistant options
     iSeries Access tasks

     Sign off

Selection or command
===>

(C) COPYRIGHT IBM CORP. 1980, 2005.
```

Click the **Save** button to save your MAIN screen customizations.

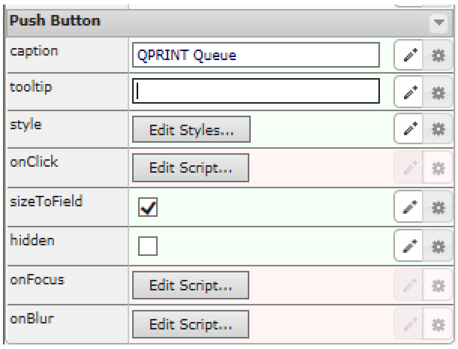
You have added a push button to customize the MAIN screen.

Using scripting to add value

Add buttons and scripts

This section explains how to add a script to a button.

Click on your new **QPRINT Queue button** to select it, and look at its property sheet:



Push Button	
caption	QPRINT Queue
tooltip	
style	Edit Styles...
onClick	Edit Script...
sizeToField	☑
hidden	□
onFocus	Edit Script...
onBlur	Edit Script...

The caption, tool tip, style, sizeToField, and hidden properties are static and are evaluated literally from what is displayed, i.e. the caption is taken literally from the text you typed in as QPRINT Queue. You can edit these by clicking the edit icon (pencil image) to the far right of the property.

The onClick, onFocus and onBlur properties are action properties and perform a JavaScript operation. The onClick property tells aXes what to do when a user clicks the QPRINT Queue button, and in this example, the JavaScript code SENDKEY(ENV.defaultKey); will be executed.

> Almost all properties can be evaluated as JavaScript code. This is a very powerful, flexible and dynamic feature that is dealt with in more detail in following tutorials.

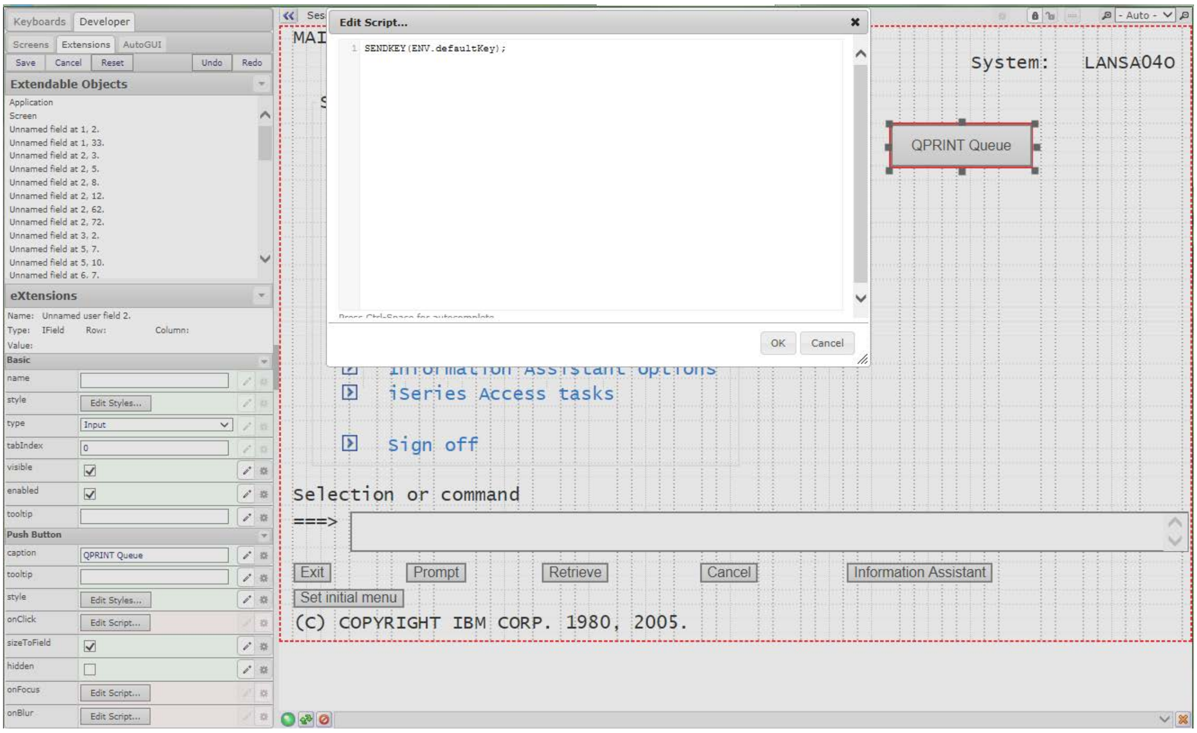
Without going into the details right now, as the name suggests, the SENDKEY(ENV.defaultKey); JavaScript function call sends a key stroke corresponding to the Environment default key into the 5250 session. Typically, this is the Enter key, unless a different default key has been specified.

Clicking the QPRINT Queue button acts just like pressing the Enter key.

You are now going to change what happens when the QPRINT Queue button is clicked.

Click the **Edit Script** button adjacent to the onClick property.

A small editor window like this should appear:



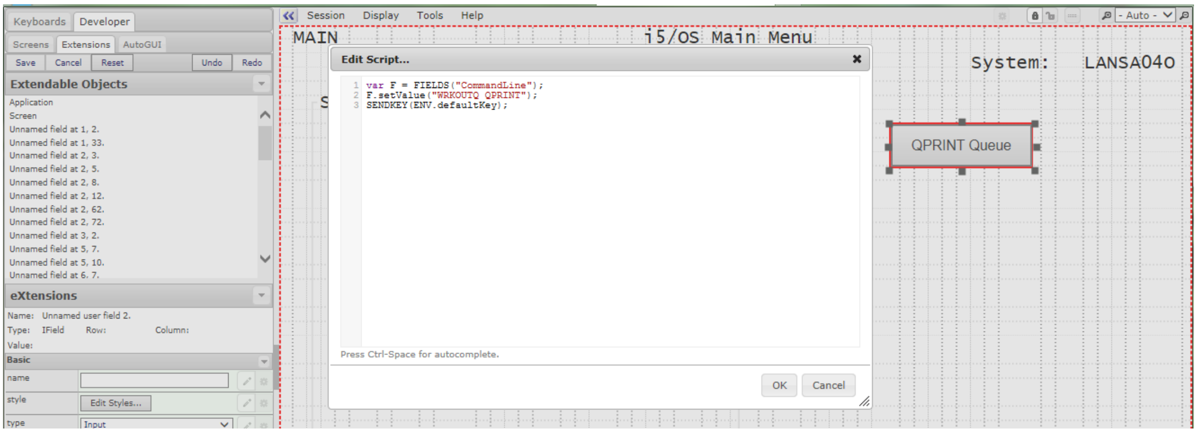
```
SENDKEY(ENV.defaultKey);
```

Select and delete the **SENDKEY(ENV.defaultKey);** code.

Copy and paste the following code into the edit window.

```
var F = FIELDS("CommandLine");
F.setValue("WRKOUTQ QPRINT");
SENDKEY(ENV.defaultKey);
```

The editor window should look like the next screen.



```
var F = FIELDS("CommandLine");
F.setValue("WRKOUTQ QPRINT");
SENDKEY(ENV.defaultKey);
```

Click OK in the editor window to apply your changes to the property value.

What does this code do? It locates a reference to the field you have named **CommandLine** on the current MAIN screen (remember, you assigned this name in Tutorial 1). It then sets the CommandLine screen field to the value "WRKOUTQ QPRINT", and finally presses the Enter key.

In short, pressing the QPRINT Queue button should now execute the IBM i WRKOUTQ QPRINT command.

This is a short cut way to display the IBM i output queue named QPRINT.

It adds value to the application because it makes displaying the output queue faster, especially when you display the output queue QPRINT frequently.

Next, to facilitate the following tutorials you need to add two more button eXtensions to the MAIN screen.

Follow the same steps you used when adding the **QPRINT Queue** button.

First, add a button with the caption **Add aXesDemo to Library List** that executes this code when clicked:

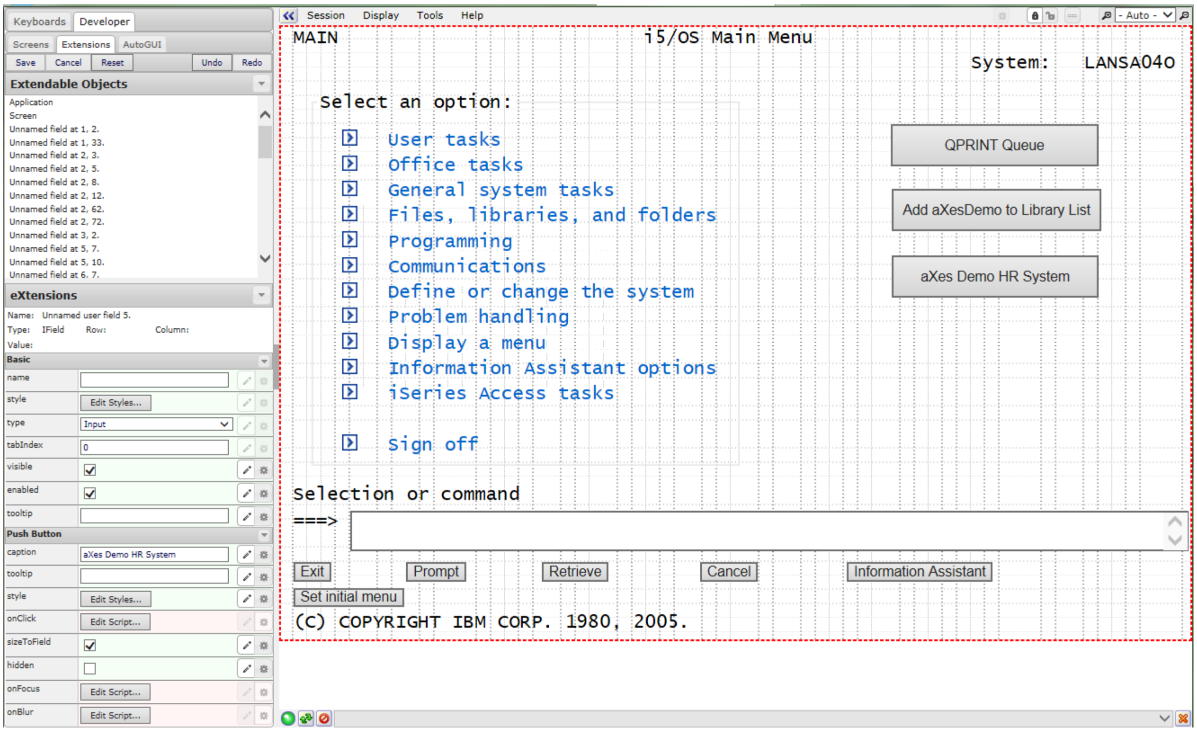
```
var F = FIELDS("CommandLine");
F.setValue("ADDLIBLE AXESDEMO");
SENDKEY(ENV.defaultKey);
```

Then, add another new button with the caption **aXes Demo HR System** that executes this code when clicked:

```
var F = FIELDS("CommandLine");
F.setValue("CALL XHRRPGTRN");
SENDKEY(ENV.defaultKey);
```

Save your screen customizations.

Your customized version of the MAIN screen should now look something like this:



```
Application
Screen
Unnamed field at 1, 2.
Unnamed field at 1, 33.
Unnamed field at 2, 3.
Unnamed field at 2, 5.
Unnamed field at 2, 8.
Unnamed field at 2, 12.
Unnamed field at 2, 62.
Unnamed field at 2, 72.
Unnamed field at 3, 2.
Unnamed field at 5, 7.
Unnamed field at 5, 10.
Unnamed field at 6, 7.
```

eXtensions

Name: Unnamed user field 5.
Type: IField Row: Column:
Value:

Basic	
name	
style	Edit Styles...
type	Input
tabIndex	0
visible	☑
enabled	☑
tooltip	
Push Button	
caption	aXes Demo HR System
tooltip	
style	Edit Styles...
onClick	Edit Script...
sizeToField	☑
hidden	□
onFocus	Edit Script...
onBlur	Edit Script...

```
MAIN                            i5/OS Main Menu
                                                          System:   LANSA040

 Select an option:

     User tasks
     Office tasks
     General system tasks
     Files, libraries, and folders
     Programming
     Communications
     Define or change the system
     Problem handling
     Display a menu
     Information Assistant options
     iSeries Access tasks

     Sign off

Selection or command
===>

(C) COPYRIGHT IBM CORP. 1980, 2005.
```

Testing customizations

Always test your customizations as a user.

You can test your screen customizations while logged on to aXes as a developer, but you must always also test your application while logged on as a user.

To test as a user, open the Projects Home Page.

Select your project, and then use the **Work as a User (aXes-TS2)** option to start aXes.

Log on and verify that the customizations on the MAIN screen display correctly and try out the new buttons.

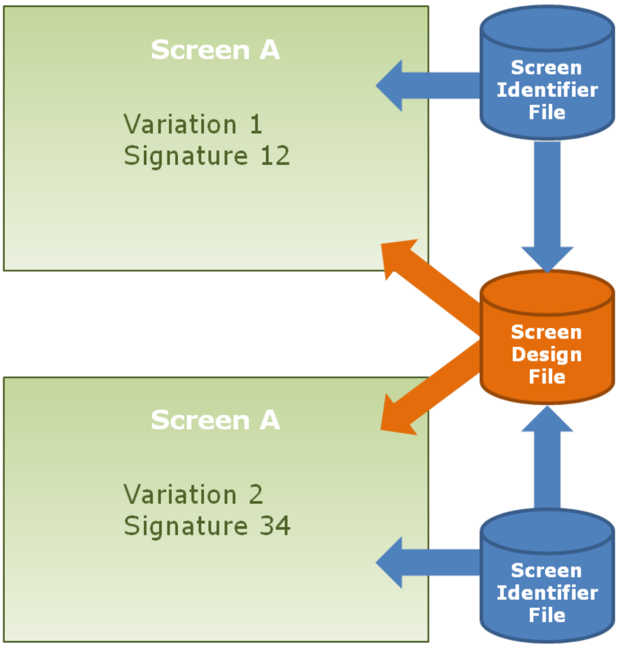
Screen identifier/design concepts

When you use aXes eXtensions you need to understand the terms **screen identifier** and **screen design**.

Screen identifier	When you save customizations, screen identification information goes into a **.scn** file (for example 14.scn). This file is referred to as a screen identifier file.
Screen design	The design customizations go into a file named: **screen_<screenname>.js** This file is referred to as a screen design file.

A screen identifier uniquely identifies one variation of a screen. A variation can be, for example, a screen in display mode, and another variation can be the same screen in update mode. Both variations have their own screen identifiers.

If a screen has been customized, the screen identifier points to the screen design. When there are variations of a screen, all screen identifiers point to the same screen design.



aXes® Documentation Library

eXtensions Tutorial 3 - Advanced Screen Enhancement

Objectives

This tutorial teaches you how to customize 5250 screens by moving and hiding screen elements, and adding new screen elements using aXes eXtensions including dates, dropdown lists, group boxes, hyperlinks, images, push buttons, and radio buttons.

Prerequisites

Tutorial 2 – Basic Screen Enhancement

Before attempting this tutorial, please complete all of the steps in Tutorial 2.

Outline

What this tutorial covers

Screen and field names

What’s the plan?

Getting assistance

Setting up your styles

Defining styles

Why are styles important?

Adding a stripe

Moving elements, changing labels and captions

Move and resize screen elements

Hiding screen elements

Applying styles

Tooltips

Dates

Radio buttons

Drop down lists

Using row and other scripting objects

Push buttons

Multi-lingual text

Group boxes

Images

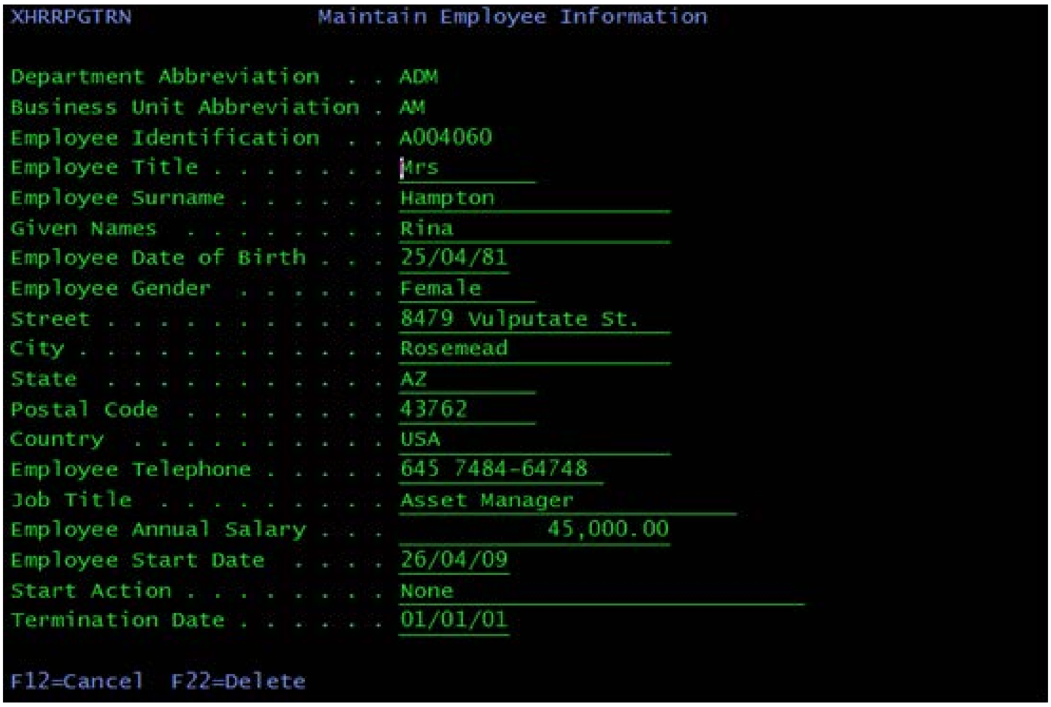
Hyperlinks

Finish the screen customization

Testing customizations

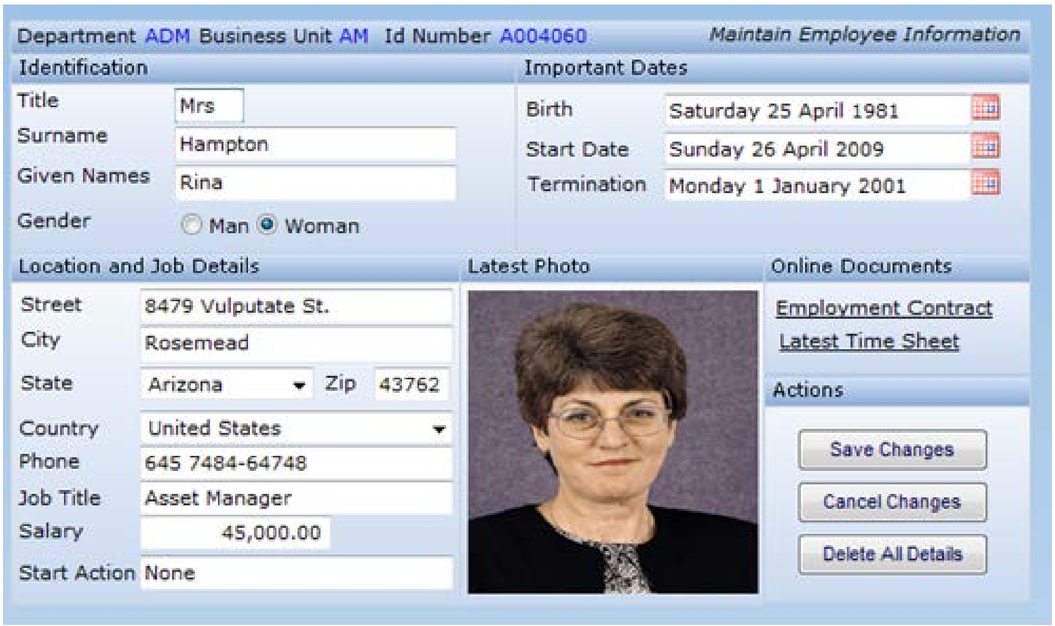
What this tutorial covers

Tutorial 3 focuses on customizing this 5250 screen:



```
XHRRPGTRN                Maintain Employee Information

Department Abbreviation  . . ADM
Business Unit Abbreviation . AM
Employee Identification  . . A004060
Employee Title . . . . . . . Mrs
Employee Surname . . . . . . Hampton
Given Names  . . . . . . . . Rina
Employee Date of Birth . . . 25/04/81
Employee Gender  . . . . . . Female
Street . . . . . . . . . . . 8479 Vulputate St.
City . . . . . . . . . . . . Rosemead
State  . . . . . . . . . . . AZ
Postal Code  . . . . . . . . 43762
Country  . . . . . . . . . . USA
Employee Telephone . . . . . 645 7484-64748
Job Title  . . . . . . . . . Asset Manager
Employee Annual Salary . . .            45,000.00
Employee Start Date  . . . . 26/04/09
Start Action . . . . . . . . None
Termination Date . . . . . . 01/01/01

F12=Cancel  F22=Delete
```

By the end of the tutorial, you will be able to make it look and act like this:



Department ADM Business Unit AM Id Number A004060 *Maintain Employee Information*

Identification

Title: Mrs
Surname: Hampton
Given Names: Rina
Gender: ○ Man ● Woman

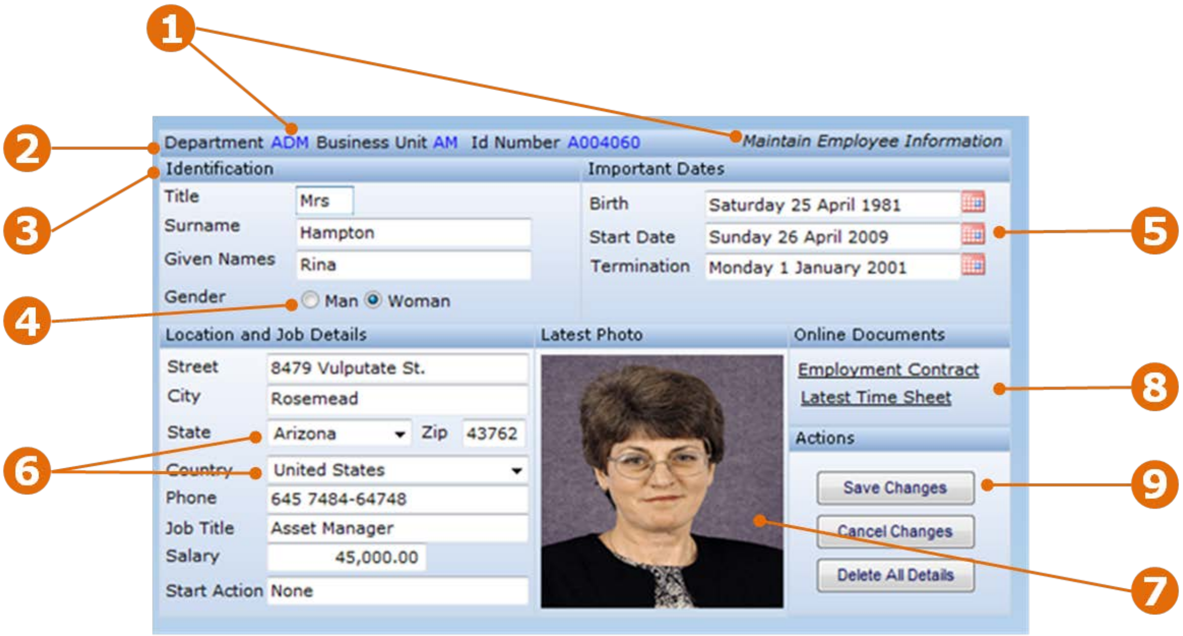
Important Dates

Birth: Saturday 25 April 1981
Start Date: Sunday 26 April 2009
Termination: Monday 1 January 2001

Location and Job Details

Street: 8479 Vulputate St.
City: Rosemead
State: Arizona
Zip: 43762
Country: United States
Phone: 645 7484-64748
Job Title: Asset Manager
Salary: 45,000.00
Start Action: None

The screen customization includes these areas of visual customization and functional enrichment:



Department ADM Business Unit AM Id Number A004060 *Maintain Employee Information*

Identification

Title: Mrs

Surname: Hampton

Given Names: Rina

Gender: ○ Man ● Woman

Important Dates

Birth: Saturday 25 April 1981

Start Date: Sunday 26 April 2009

Termination: Monday 1 January 2001

Location and Job Details

Street: 8479 Vulputate St.

City: Rosemead

State: Arizona Zip: 43762

Country: United States

Phone: 645 7484-64748

Job Title: Asset Manager

Salary: 45,000.00

Start Action: None

Latest Photo

Online Documents

Employment Contract

Latest Time Sheet

Actions

	Features	Tutorial Topics
1	Styles and styling	Defining and using styles. How to style screen elements by their role.
2	Moving, hiding and enhancing	Understand how to move things around the screen. Hide and alter screen content. Use tool tips (hints) as a substitute for long labels.
3	Group Boxes	Use group boxes and stripes to visually group and draw attention to information.
4	Radio Buttons	How to use radio buttons.
5	Dates	How to change the way that dates are displayed and adding calendar pickers.
6	Drop Downs	How to add drop downs or combo boxes.
7	Images	How to add images.
8	Hyperlinks	How to add hyperlinks to display associated information.
9	Buttons	How to add buttons and script what happens when users click them.

Screen and field names

This section defines screen and field names used in all tutorials.

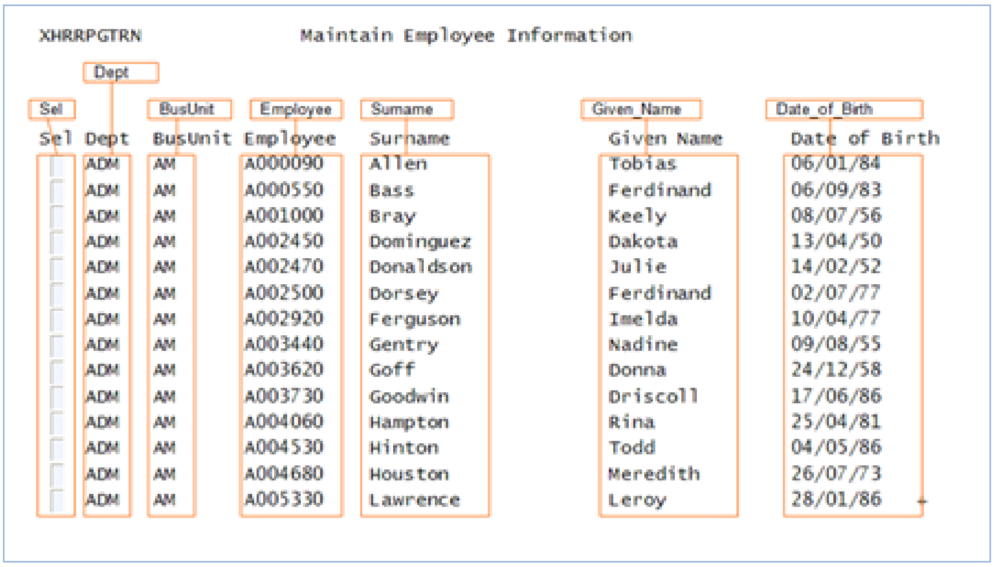
You don’t have to read this section just now. It is a reference point for this and following tutorials.

The screens are XHRRPGTRN_Select and XHRRPGTRN_Maint.

You can display the screens by adding AXESDEMO to your library list, and then calling the program XHRRPGTRN.

XHRRPGTRN_Select and its field names

This is the XHRRPGTRN_Select screen, with its fields named as indicated:

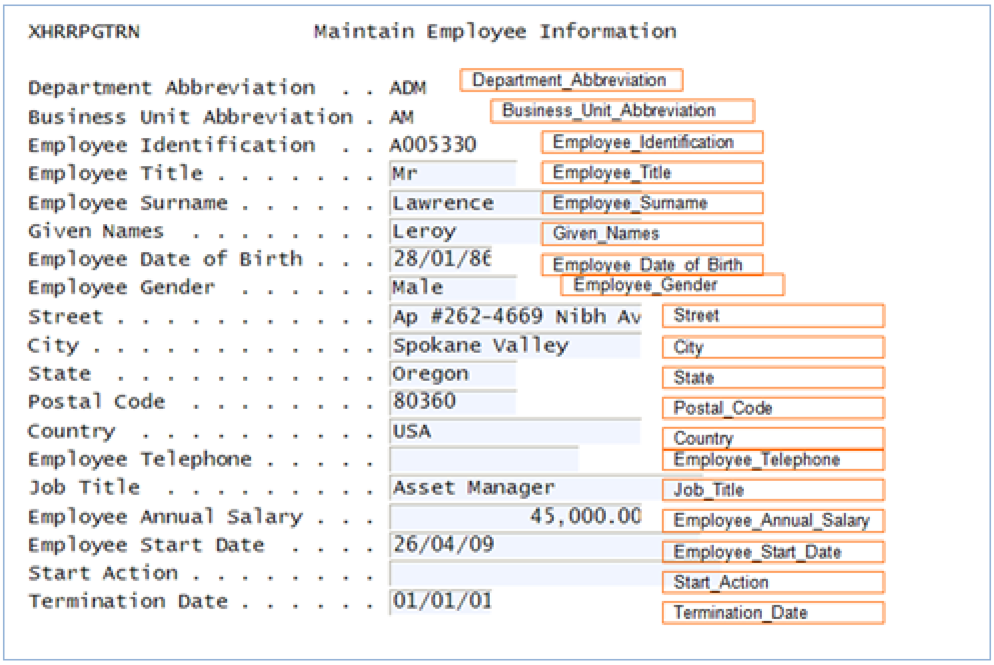


XHRRPGTRN Maintain Employee Information

Sel	Dept	BusUnit	Employee	Surname	Given Name	Date of Birth
□	ADM	AM	A000090	Allen	Tobias	06/01/84
□	ADM	AM	A000550	Bass	Ferdinand	06/09/83
□	ADM	AM	A001000	Bray	Keely	08/07/56
□	ADM	AM	A002450	Dominguez	Dakota	13/04/50
□	ADM	AM	A002470	Donaldson	Julie	14/02/52
□	ADM	AM	A002500	Dorsey	Ferdinand	02/07/77
□	ADM	AM	A002920	Ferguson	Imelda	10/04/77
□	ADM	AM	A003440	Gentry	Nadine	09/08/55
□	ADM	AM	A003620	Goff	Donna	24/12/58
□	ADM	AM	A003730	Goodwin	Driscoll	17/06/86
□	ADM	AM	A004060	Hampton	Rina	25/04/81
□	ADM	AM	A004530	Hinton	Todd	04/05/86
□	ADM	AM	A004680	Houston	Meredith	26/07/73
□	ADM	AM	A005330	Lawrence	Leroy	28/01/86 +

XHRRPGTRN_Maint and its field names

This is the XHRRPGTRN_Maint screen, with its fields named as indicated:



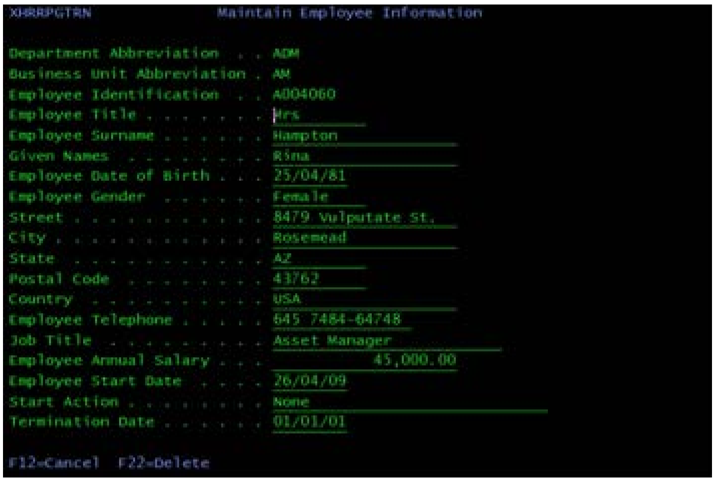
XHRRPGTRN   Maintain Employee Information

Department Abbreviation . .	ADM	Department_Abbreviation
Business Unit Abbreviation .	AM	Business_Unit_Abbreviation
Employee Identification . .	A005330	Employee_Identification
Employee Title	Mr	Employee_Title
Employee Surname	Lawrence	Employee_Surname
Given Names	Leroy	Given_Names
Employee Date of Birth . . .	28/01/86	Employee_Date_of_Birth
Employee Gender	Male	Employee_Gender
Street	Ap #262-4669 Nibh Av	Street
City	Spokane Valley	City
State	Oregon	State
Postal Code	80360	Postal_Code
Country	USA	Country
Employee Telephone		Employee_Telephone
Job Title	Asset Manager	Job_Title
Employee Annual Salary . . .	45,000.00	Employee_Annual_Salary
Employee Start Date	26/04/09	Employee_Start_Date
Start Action		Start_Action
Termination Date	01/01/01	Termination_Date

What’s the plan?

It’s important to have a 5250 screen customization and enhancement plan. Creating a plan will ensure that the look and feel of the screens remains consistent after customization, and help you to allocate a priority to the screens you intend to customize – work on the most important screens first.

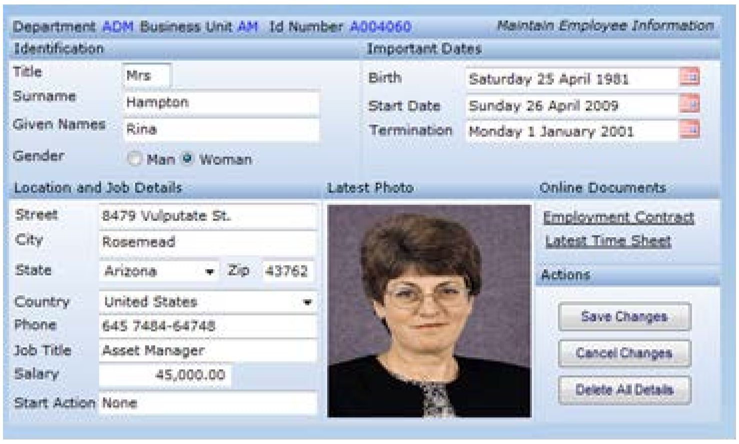
In this tutorial, the plan is to customize this 5250 screen:



```
XHRRPGTRN                 Maintain Employee Information

Department Abbreviation  . .  ADM
Business Unit Abbreviation .  AM
Employee Identification  . .  A004060
Employee Title . . . . . . .  Mrs
Employee Surname . . . . . .  Hampton
Given Names  . . . . . . . .  Rina
Employee Date of Birth . . .  25/04/81
Employee Gender  . . . . . .  Female
Street . . . . . . . . . . .  8479 Vulputate St.
City . . . . . . . . . . . .  Rosemead
State  . . . . . . . . . . .  AZ
Postal Code  . . . . . . . .  43762
Country  . . . . . . . . . .  USA
Employee Telephone . . . . .  645 7484-64748
Job Title  . . . . . . . . .  Asset Manager
Employee Annual Salary . . .              45,000.00
Employee Start Date  . . . .  26/04/09
Start Action . . . . . . . .  None
Termination Date . . . . . .  01/01/01

F12=Cancel  F22=Delete
```

The customization will add these visual and functional eXtensions to the screen: styles, a stripe, date pickers, a radio button, drop downs, buttons, group boxes, an image and hyperlinks.

The outcome of the customizations will look like this screen.



Department ADM Business Unit AM Id Number A004060 *Maintain Employee Information*

Identification

Title	Mrs
Surname	Hampton
Given Names	Rina
Gender	○ Man ● Woman

Important Dates

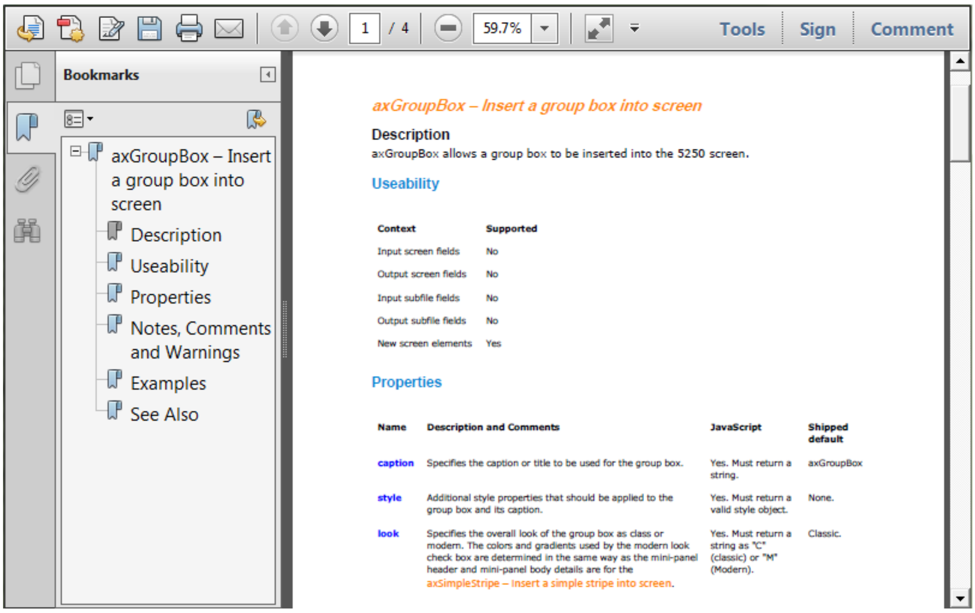
Birth	Saturday 25 April 1981
Start Date	Sunday 26 April 2009
Termination	Monday 1 January 2001

Location and Job Details

Street	8479 Vulputate St.
City	Rosemead
State	Arizona Zip 43762
Country	United States
Phone	645 7484-64748
Job Title	Asset Manager
Salary	45,000.00
Start Action	None

Getting assistance

While completing these tutorials you can get information from the product documentation. This is a sample page from the eXtensions Guide.



axGroupBox – Insert a group box into screen

Description

axGroupBox allows a group box to be inserted into the 5250 screen.

Useability

Context	Supported
Input screen fields	No
Output screen fields	No
Input subfile fields	No
Output subfile fields	No
New screen elements	Yes

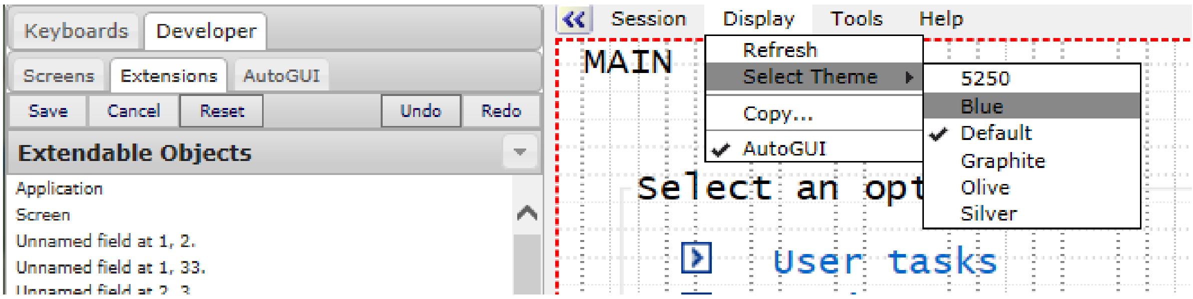
Properties

Name	Description and Comments	JavaScript	Shipped default
caption	Specifies the caption or title to be used for the group box.	Yes. Must return a string.	axGroupBox
style	Additional style properties that should be applied to the group box and its caption.	Yes. Must return a valid style object.	None.
look	Specifies the overall look of the group box as class or modern. The colors and gradients used by the modern look check box are determined in the same way as the mini-panel header and mini-panel body details are for the axSimpleStripe – Insert a simple stripe into screen.	Yes. Must return a string as "C" (classic) or "M" (Modern).	Classic.

The aXes Technical Documentation Index provides a list of the documents.

Setting up your styles

Up until now, you may have used one of the shipped basic 5250 themes, e.g. blue or default, to change the appearance of your aXes 5250 application.



Once you start customizing an application you should stop using the basic 5250 themes and develop your own customized role-based styles and themes.

> What follows in this tutorial covers the basics of screen styling.
>
> Later you should review Tutorial 10 – 5250 Screen Styling to obtain an in depth understanding of what you can do with customized styles and themes.

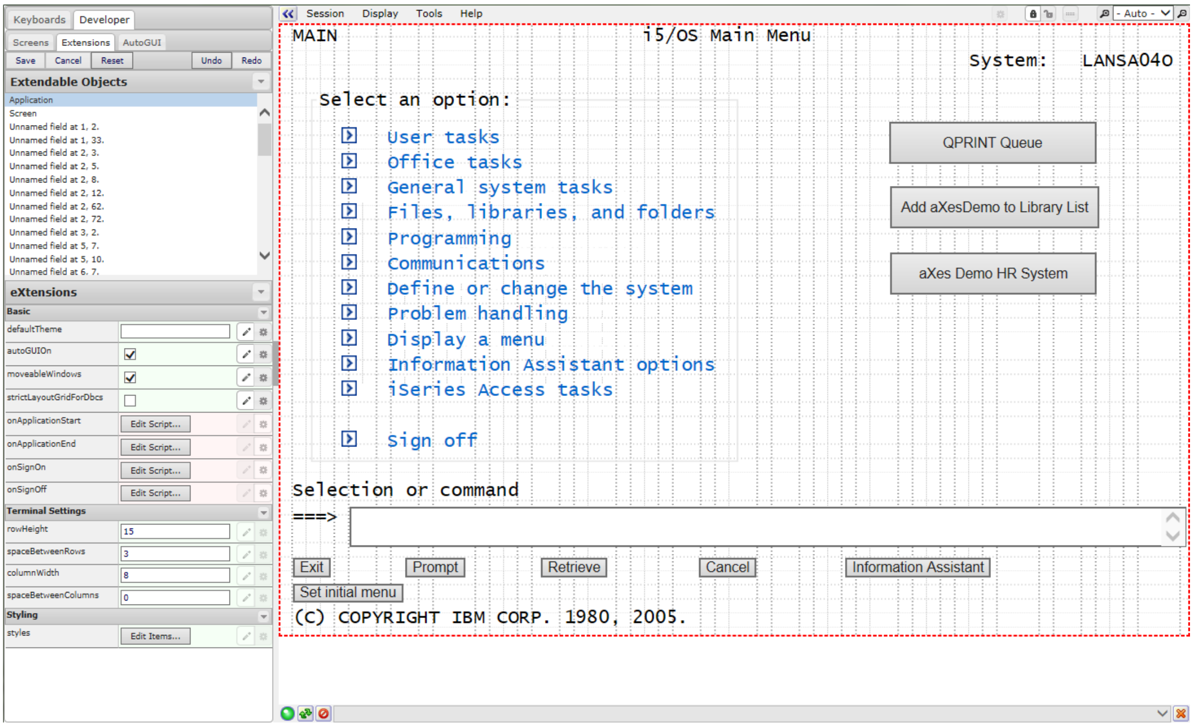
Defining styles

To get started with customized styles and themes you need to define some styles.

The next task will add styles for background, font, key information and screen title. You will use the styles later in the tutorial.

Open the Projects Home Page.

Select your project, and then use the **Work as TS2 Developer** option and sign on to a 5250 session.



defaultTheme
autoGUIOn ☑
moveableWindows ☑
strictLayoutGridForDbcs ☐
onApplicationStart
onApplicationEnd
onSignOn
onSignOff
rowHeight 15
spaceBetweenRows 3
columnWidth 8
spaceBetweenColumns 0
styles

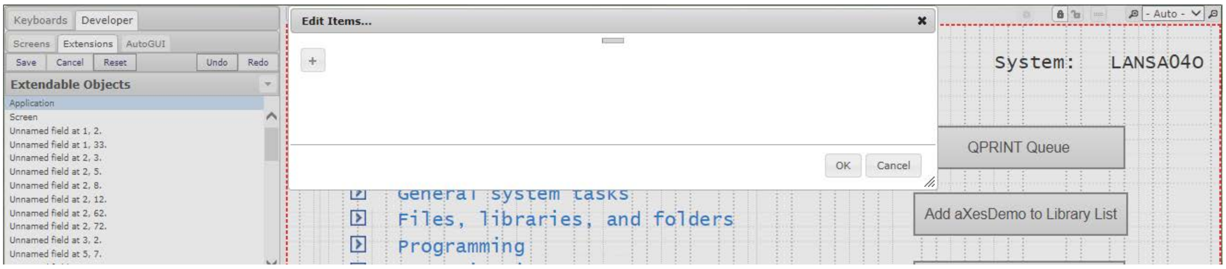
```
MAIN                          i5/OS Main Menu
                                                    System:   LANSA040

 Select an option:

    User tasks
    Office tasks
    General system tasks
    Files, libraries, and folders
    Programming
    Communications
    Define or change the system
    Problem handling
    Display a menu
    Information Assistant options
    iSeries Access tasks

    Sign off

Selection or command
===>

(C) COPYRIGHT IBM CORP. 1980, 2005.
```

In the aXes Developer window, click **Application** in the **Extendable Objects** list to view the application properties. No need to put in edit mode, it's already editable.

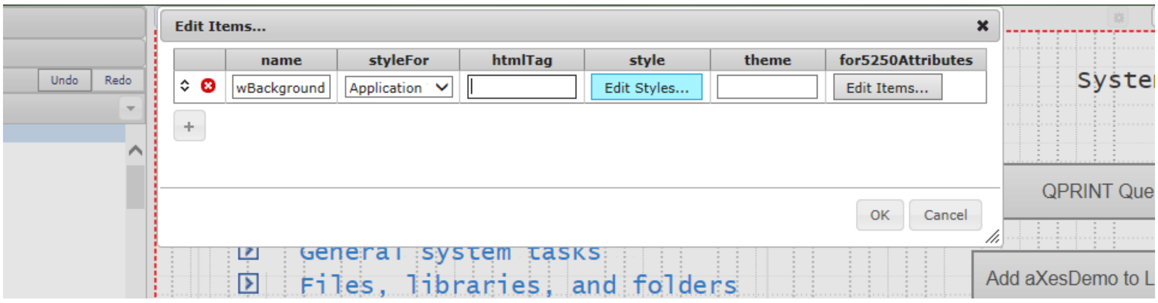
Locate the **Styling** bar at the bottom of the aXes Developer window.

Click on the **Edit Items** button adjacent to the **styles** label.

aXes will display the Edit Items window.



Click on the **plus button** [+] to add a style item.

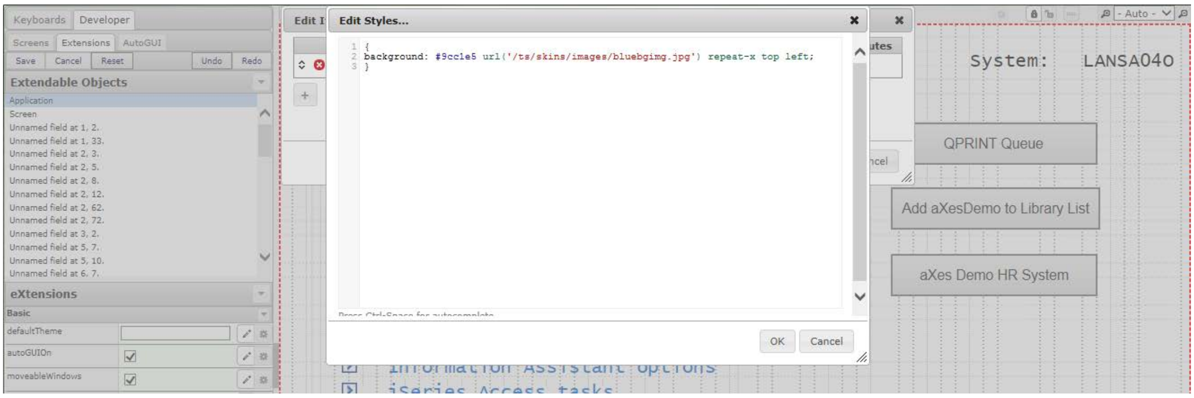


name	styleFor	htmlTag	style	theme	for5250Attributes
wBackground	Application		Edit Styles...		Edit Items...

Use the following property values for the **background style**.

Name	BasicWindowBackground
styleFor	Application
htmlTag	
style	
theme	
for5250Attributes	

Click the **Edit Style** button in the **style** property.



```
{
background: #9cc1e5 url('/ts/skins/images/bluebgimg.jpg') repeat-x top left;
}
```

Copy this code into the Edit Styles window, then click OK.

```
{
background: #9cc1e5 url('/ts/skins/images/bluebgimg.jpg')
repeat-x top left;
}
```

This will set the image (bluebgimg.jpg) as the screen background.

Next, add the font, key information and screen title styles.

Click on the plus button [+] to add an item for the font style.

Use the following property values for the **font style**.

Name	BasicFont
styleFor	All
htmlTag	

style	Click the Edit Style button, copy this code into the Edit Styles window, then click OK.
	`{`
	`font-family:Verdana;`
	`font-size:9pt;`
	`}`

theme	
for5250Attributes	

Click on the plus button [+] to add an item for the key information style.

Use the following property values for the **key information style**.

Name	KeyInformation
styleFor	
htmlTag	

style	Click the Edit Style button, copy this code into the Edit Styles window, then click OK.
	`{`
	`color:blue;`
	`}`

theme	
for5250Attributes	

Click on the plus button [+] to add an item for the screen title style.

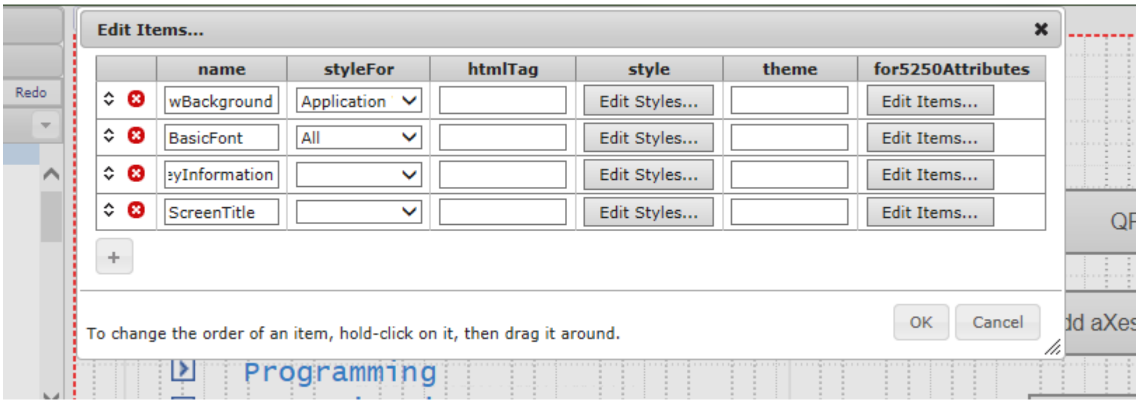
Use the following property values for the **screen title style**.

Name	ScreenTitle
styleFor	
htmlTag	

style	Click the Edit Style button, copy this code into the Edit Styles window, then click OK.
	{
	font-style:italic;
	}

theme	
for5250Attributes	

When you finish adding the styles the Edit Items window should look like this:



	name	styleFor	htmlTag	style	theme	for5250Attributes
	wBackground	Application		Edit Styles...		Edit Items...
	BasicFont	All		Edit Styles...		Edit Items...
	yInformation			Edit Styles...		Edit Items...
	ScreenTitle			Edit Styles...		Edit Items...

To change the order of an item, hold-click on it, then drag it around.

Click the OK button to close the Edit Items window.

To see the effect of your styles:

Sign off from your aXes development session.

Close the browser window.

Start your aXes development session again.

You should see the result of the BasicWindowBackground style.

Why are styles important?

You define styles so that screen elements can be quickly associated with their visual display characteristics or rules, e.g. all screen titles are to be displayed in italics, and screens will have a blue background.

> Styling is a sophisticated feature and there is another tutorial dedicated to styling.

The benefits of using styles are compelling.

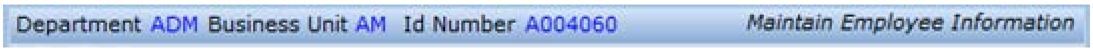
Standardization	Styles produce standardization, which is essential to produce a common look and feel, and to prevent unconstrained style use of styles.
Single point of change	Styles provide a single point of change. What do you do when the marketing people decide screen titles should be italic/green this week and bold/blue next week? Using styles simplifies the change, e.g. This week - font-color: green; font-style: italic; Next week - font-color: blue; font-style: bold;

If by default a style is to be applied to all instances of an eXtension (e.g. we want all buttons to have the same appearance), the style should be applied to the button eXtension template instead of applying the change to each button instance, as this saves you from having to assign a style to individual buttons.

Please refer to FAQ – Customizing eXtensions for more information on how to customize an eXtension template.

Adding a stripe

In our customization plan there is a "bar" or "stripe" across the top of the 5250 screen. After you have added the stripe to the top of your 5250 screen, it will look like this:



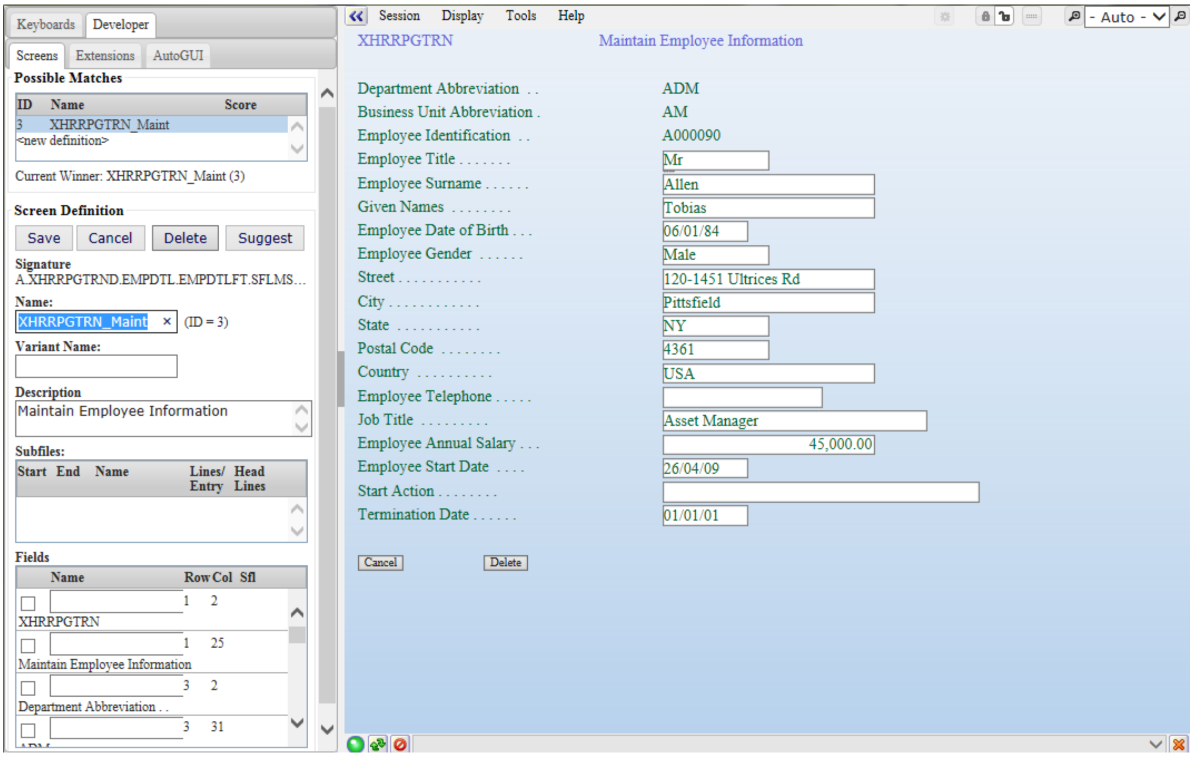
Department ADM Business Unit AM Id Number A004060 Maintain Employee Information

Open the Projects Home Page.

Select your project, and then use the Work as Developer option, and sign on to a 5250 session.

Add library AXESDEMO to your library list and call program XHRRPGTRN using the buttons you created in the previous tutorial.

Select one of the displayed employees with an X and press enter. The resulting 5250 screen will look something like this:



Possible Matches

ID	Name	Score
3	XHRRPGTRN_Maint	
	<new definition>	

Current Winner: XHRRPGTRN_Maint (3)

Screen Definition

Signature
A.XHRRPGTRND.EMPDTL.EMPDTLFT.SFLMS...

Name:
XHRRPGTRN_Maint (ID = 3)

Variant Name:

Description
Maintain Employee Information

Subfiles:

Start	End	Name	Lines/ Entry	Head Lines

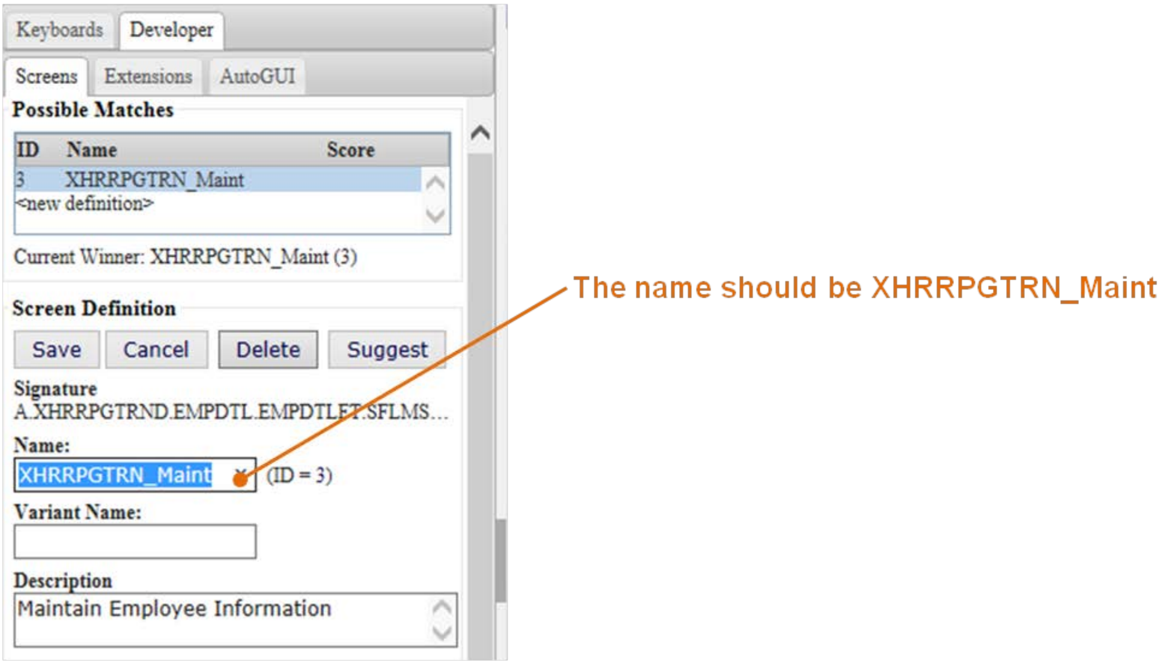
Fields

	Name	Row	Col	Sfl
□	XHRRPGTRN	1	2	
□	Maintain Employee Information	1	25	
□	Department Abbreviation . .	3	2	
□	ADM	3	31	

XHRRPGTRN Maintain Employee Information

Department Abbreviation . .	ADM
Business Unit Abbreviation .	AM
Employee Identification . .	A000090
Employee Title	Mr
Employee Surname	Allen
Given Names	Tobias
Employee Date of Birth . . .	06/01/84
Employee Gender	Male
Street	120-1451 Ultrices Rd
City	Pittsfield
State	NY
Postal Code	4361
Country	USA
Employee Telephone	
Job Title	Asset Manager
Employee Annual Salary . . .	45,000.00
Employee Start Date	26/04/09
Start Action	
Termination Date	01/01/01

You identified this screen as XHRRPGTRN_Maint in a previous tutorial.



Possible Matches

ID	Name	Score
3	XHRRPGTRN_Maint	
<new definition>		

Current Winner: XHRRPGTRN_Maint (3)

Screen Definition

Signature
A.XHRRPGTRND.EMPDTL.EMPDTLET.SFLMS…

Name:
XHRRPGTRN_Maint (ID = 3)

Variant Name:

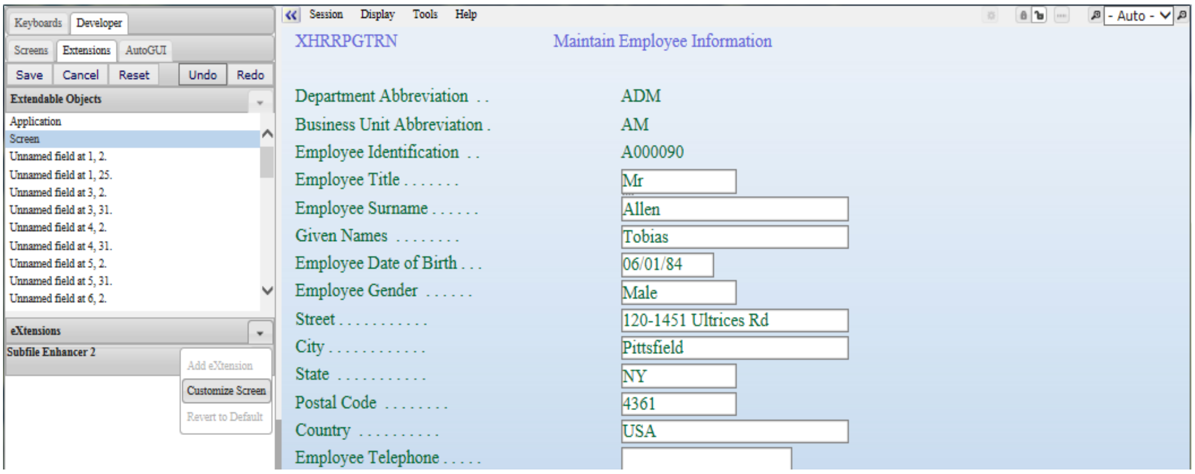
Description
Maintain Employee Information

The name should be XHRRPGTRN_Maint

Check that the name in the Screen Definition shown on the Screens tab in the Developer window is XHRRPGTRN_Maint.

> If the name is not XHRRPGTRN_Maint, or you do not understand why it is XHRRPGTRN_Maint, please complete all the preceding tutorials before going any further in this tutorial.

Click **Screen** in the **Extendable Objects** list and then choose **Customize Screen** from the eXtensions actions button (in the eXtensions bar of the Developer window).

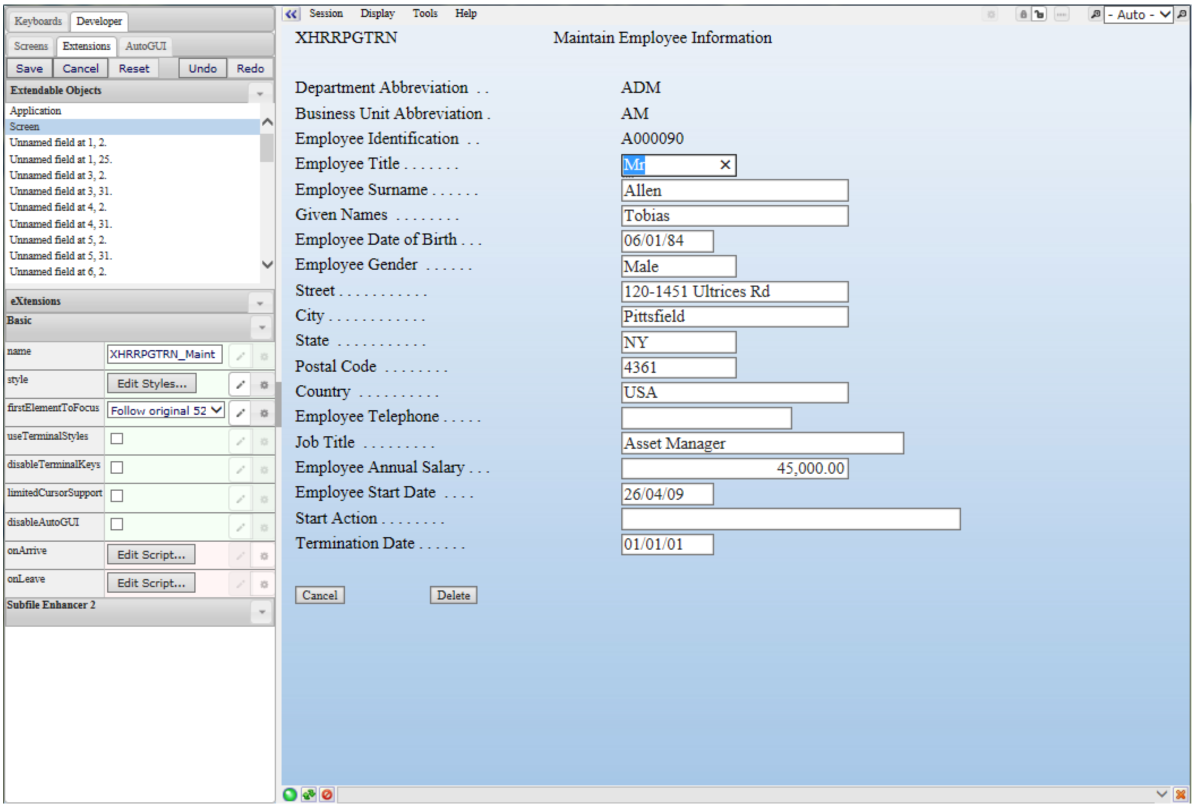


XHRRPGTRN Maintain Employee Information

Department Abbreviation . . ADM
Business Unit Abbreviation . AM
Employee Identification . . A000090
Employee Title . . . . Mr
Employee Surname . . . . . . Allen
Given Names . . . . Tobias
Employee Date of Birth . . . 06/01/84
Employee Gender . . . . . . Male
Street . . . . 120-1451 Ultrices Rd
City . . . . Pittsfield
State . . . . NY
Postal Code 4361
Country . . . . USA
Employee Telephone . . . . .

By default, screen XHRRPGTRN_Maint uses the same styles as the basic, un-customized 5250 screens. Since we are going to heavily enhance this screen, you need to prevent this by changing the useTerminalStyles property to false.

The **useTerminalStyles** property is in the Basic properties below the eXtensions bar. Uncheck the useTerminalStyles property.



Extendable Objects
Application
Screen
Unnamed field at 1, 2.
Unnamed field at 1, 25.
Unnamed field at 3, 2.
Unnamed field at 3, 31.
Unnamed field at 4, 2.
Unnamed field at 4, 31.
Unnamed field at 5, 2.
Unnamed field at 5, 31.
Unnamed field at 6, 2.

eXtensions
Basic

name	XHRRPGTRN_Maint
style	Edit Styles...
firstElementToFocus	Follow original 52
useTerminalStyles	□
disableTerminalKeys	□
limitedCursorSupport	□
disableAutoGUI	□
onArrive	Edit Script...
onLeave	Edit Script...

Subfile Enhancer 2

XHRRPGTRN Maintain Employee Information

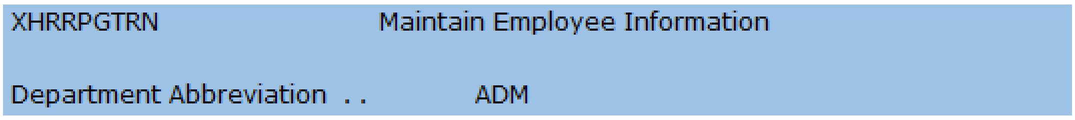
Department Abbreviation . .	ADM
Business Unit Abbreviation .	AM
Employee Identification . .	A000090
Employee Title	Mr
Employee Surname	Allen
Given Names	Tobias
Employee Date of Birth . . .	06/01/84
Employee Gender	Male
Street	120-1451 Ultrices Rd
City	Pittsfield
State	NY
Postal Code	4361
Country	USA
Employee Telephone	
Job Title	Asset Manager
Employee Annual Salary . . .	45,000.00
Employee Start Date	26/04/09
Start Action	
Termination Date	01/01/01

Cancel Delete

After unchecking the use Terminal Styles property, click the Save button.

You should see your screen change as the BasicFont style takes effect.

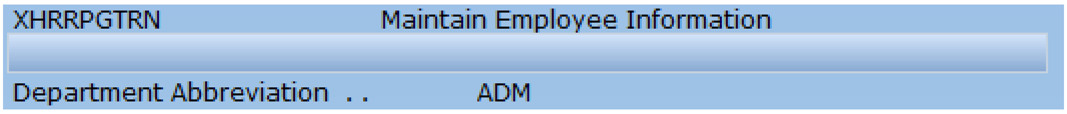
Now you are going to change the top of the screen from this:



XHRRPGTRN Maintain Employee Information

Department Abbreviation . . ADM

To this:



XHRRPGTRN Maintain Employee Information

Department Abbreviation . . ADM

Do this by adding a new element to the screen.

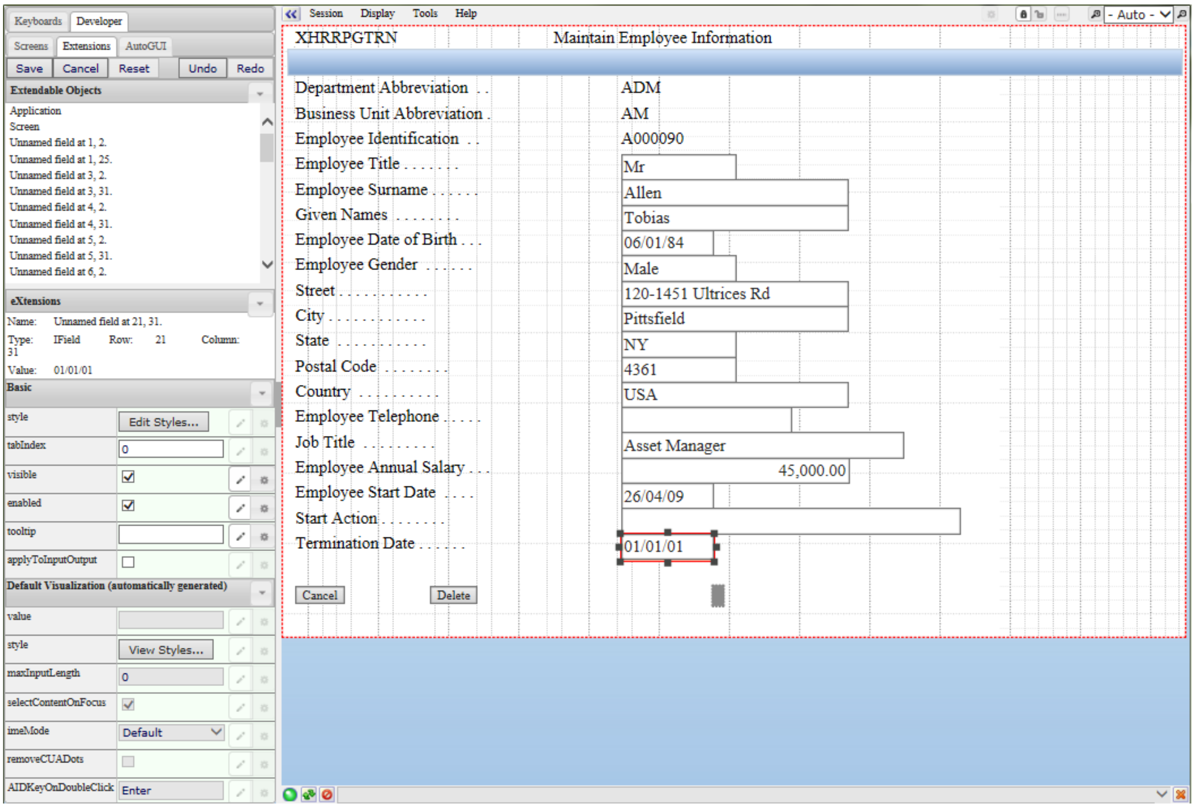
Add an eXtension and make the new element into a Simple Stripe.

Change the stripe's context to Mini-Panel Heading.

Move and position the stripe so that it sits below the screen name, above the Department Abbreviation label, and spans the screen width.

Save your changes.

The customized screen will look similar to this:



Application
Screen
Unnamed field at 1, 2.
Unnamed field at 1, 25.
Unnamed field at 3, 2.
Unnamed field at 3, 31.
Unnamed field at 4, 2.
Unnamed field at 4, 31.
Unnamed field at 5, 2.
Unnamed field at 5, 31.
Unnamed field at 6, 2.

eXtensions

Name: Unnamed field at 21, 31.
Type: IField Row: 21 Column: 31
Value: 01/01/01

Basic

style	Edit Styles...
tabIndex	0
visible	☑
enabled	☑
tooltip	
applyToInputOutput	□

Default Visualization (automatically generated)

value	
style	View Styles...
maxInputLength	0
selectContentOnFocus	☑
imeMode	Default
removeCUADots	□
AIDKeyOnDoubleClick	Enter

XHRRPGTRN Maintain Employee Information

Department Abbreviation . .	ADM
Business Unit Abbreviation .	AM
Employee Identification . .	A000090
Employee Title	Mr
Employee Surname	Allen
Given Names	Tobias
Employee Date of Birth . . .	06/01/84
Employee Gender	Male
Street	120-1451 Ultrices Rd
City	Pittsfield
State	NY
Postal Code	4361
Country	USA
Employee Telephone	
Job Title	Asset Manager
Employee Annual Salary . . .	45,000.00
Employee Start Date	26/04/09
Start Action	
Termination Date	01/01/01

Moving elements, changing labels and captions

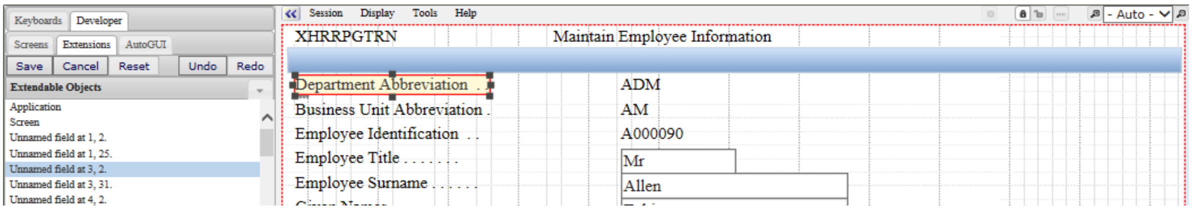
The objective of this task is to change the screen appearance by manipulating screen elements e.g. labels and fields.

Move and resize screen elements

With the aXes Developer window and the XHRRPGTRN_Maint screen displayed.

The label "Department Abbreviation .." on the 5250 screen can be made shorter.

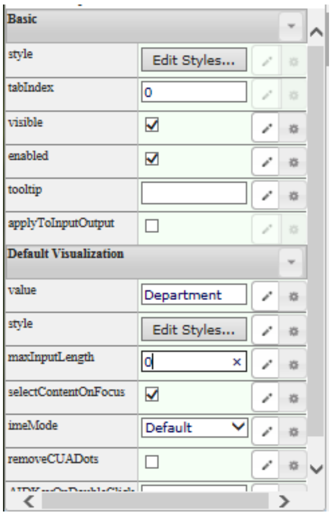
Click on the label **Department Abbreviation** to select it and choose Customize:



XHRRPGTRN Maintain Employee Information

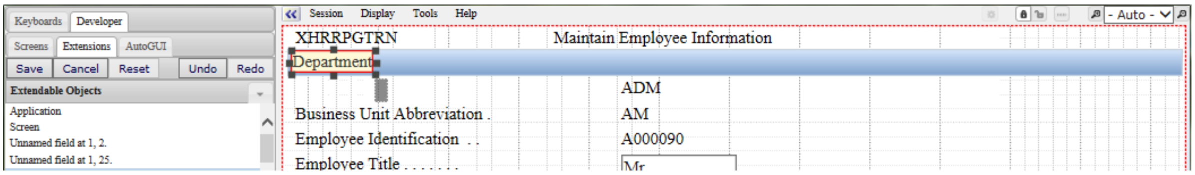
Department Abbreviation . . ADM
Business Unit Abbreviation . AM
Employee Identification . . A000090
Employee Title . . . . Mr
Employee Surname . . . . . . Allen

Change the **value** property to the word **Department** (the value property is in the Default Visualization).



Basic	
style	Edit Styles...
tabIndex	0
visible	☑
enabled	☑
tooltip	
applyToInputOutput	□
Default Visualization	
value	Department
style	Edit Styles...
maxInputLength	0
selectContentOnFocus	☑
imeMode	Default
removeCUADots	□

Now move the label up onto the stripe:



XHRRPGTRN Maintain Employee Information

Department

ADM
Business Unit Abbreviation . AM
Employee Identification . . A000090
Employee Title . . . . Mr

Your screen should show the word Department in the bar at the top left-hand side of the screen.

Save your changes.

How to move screen elements

You can move elements around either by dragging them with the mouse, or by using the arrow keys (you must select an element before you can move it).

To move multiple elements as a group, draw a box around them (in screen edit mode). Any element that is fully or partially within the box will be selected. All the selected elements can be moved (or sized) together.

The keyboard shortcuts for moving screen elements are:

Right Arrow	Move to right
Left Arrow	Move to left
Up Arrow	Move up
Down Arrow	Move down

How to resize screen elements

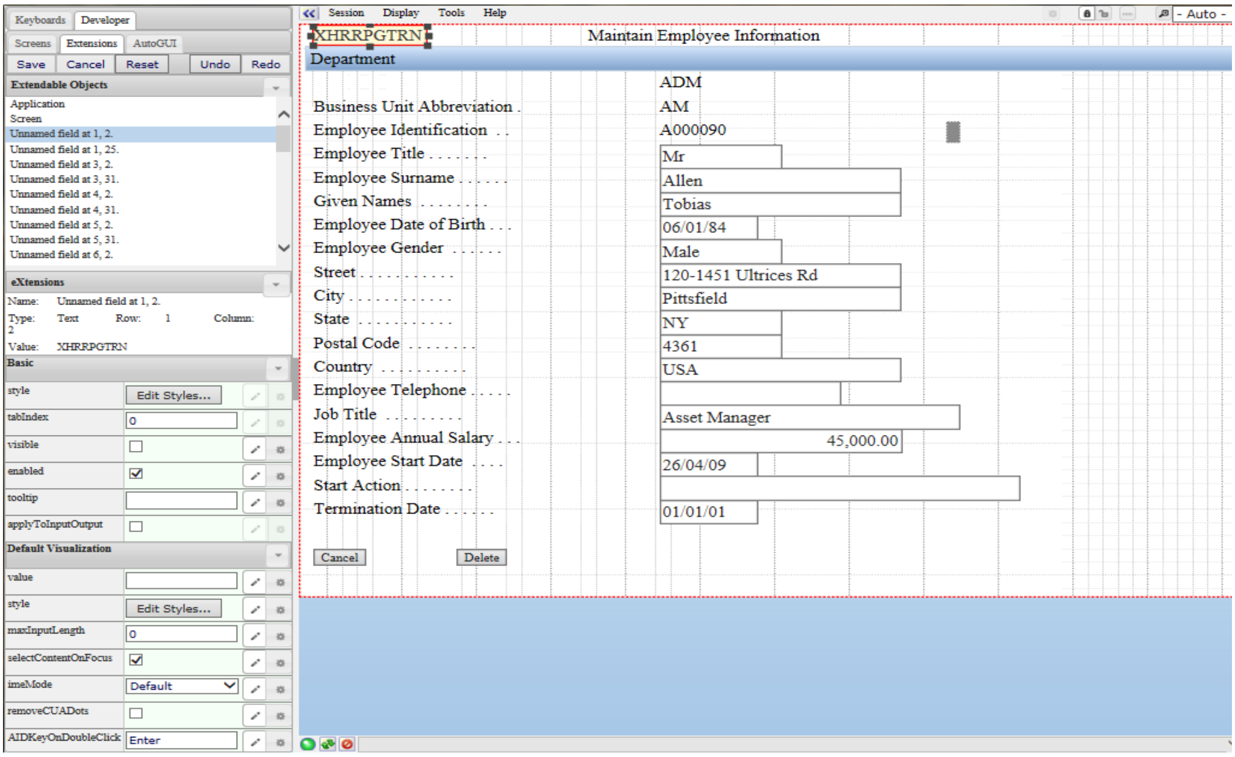
To resize an element, or a group of elements, use the mouse to click on one of the element's sizing handles and drag the element to the size you want.

Hiding screen elements

The objective of this task is to hide the screen identifier.

With the aXes Developer window and the XHRRPGTRN_Maint screen displayed.

Click on the screen identifier **XHRRPGTRN** to select it and choose Customize from the eXtensions actions button (in the eXtensions bar of the Developer window).



Extendable Objects

- Application
- Screen
- Unnamed field at 1, 2.
- Unnamed field at 1, 25.
- Unnamed field at 3, 2.
- Unnamed field at 3, 31.
- Unnamed field at 4, 2.
- Unnamed field at 4, 31.
- Unnamed field at 5, 2.
- Unnamed field at 5, 31.
- Unnamed field at 6, 2.

eXtensions

Name: Unnamed field at 1, 2.
Type: Text Row: 1 Column: 2
Value: XHRRPGTRN

Basic

style	Edit Styles...
tabIndex	0
visible	□
enabled	☑
tooltip	
applyToInputOutput	□

Default Visualization

value	
style	Edit Styles...
maxInputLength	0
selectContentOnFocus	☑
imeMode	Default
removeCUADots	□
AIDKeyOnDoubleClick	Enter

XHRRPGTRN Maintain Employee Information

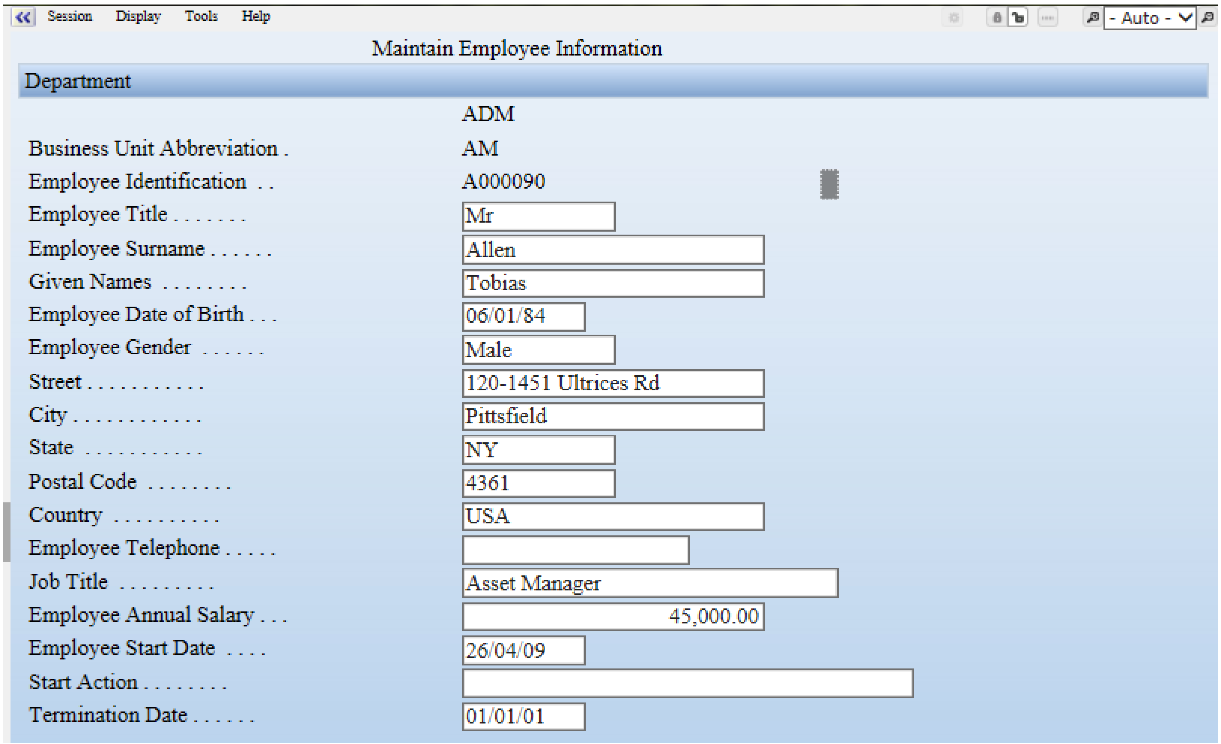
Department

	ADM
Business Unit Abbreviation .	AM
Employee Identification . .	A000090
Employee Title	Mr
Employee Surname	Allen
Given Names	Tobias
Employee Date of Birth . . .	06/01/84
Employee Gender	Male
Street	120-1451 Ultrices Rd
City	Pittsfield
State	NY
Postal Code	4361
Country	USA
Employee Telephone	
Job Title	Asset Manager
Employee Annual Salary . . .	45,000.00
Employee Start Date	26/04/09
Start Action	
Termination Date	01/01/01

Cancel Delete

Uncheck the **visible** property, then save your changes and unlock the screen.

The screen identifier, XHRRPGTRN, will no longer appear on the screen.



Maintain Employee Information

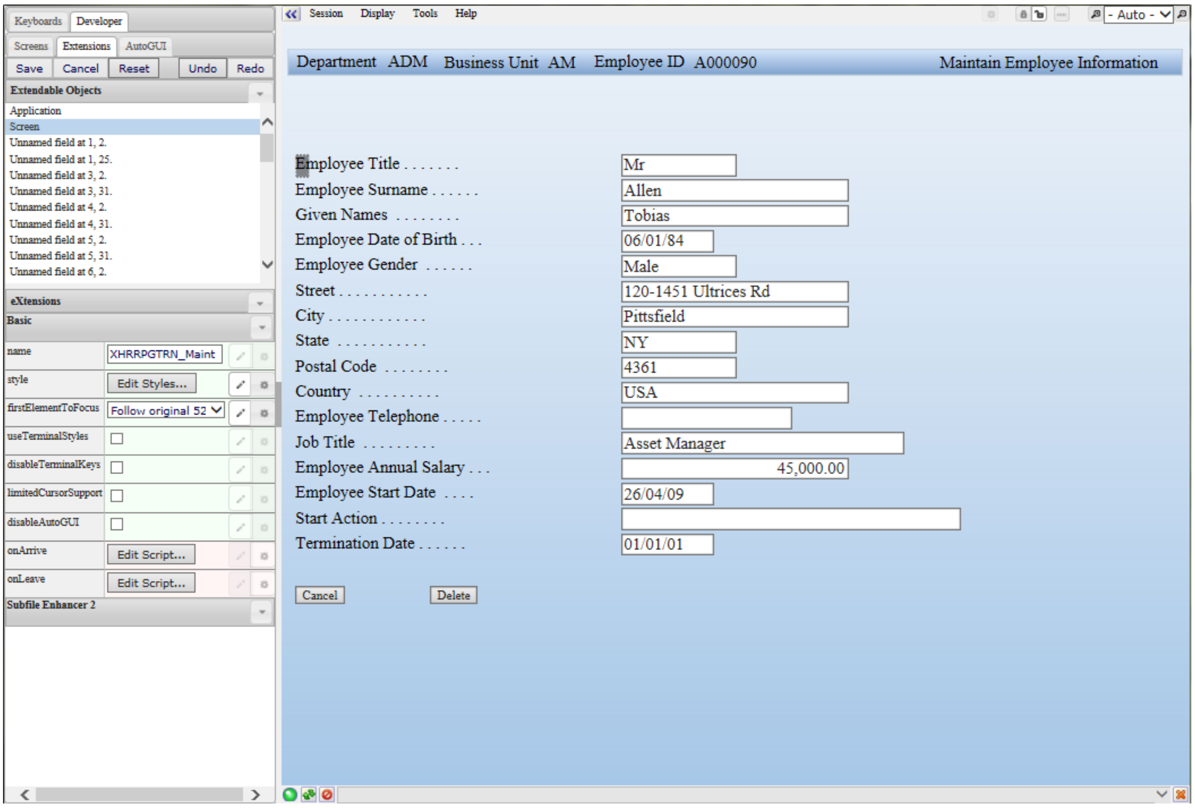
Department

	ADM
Business Unit Abbreviation .	AM
Employee Identification . .	A000090
Employee Title	Mr
Employee Surname	Allen
Given Names	Tobias
Employee Date of Birth . . .	06/01/84
Employee Gender	Male
Street	120-1451 Ultrices Rd
City	Pittsfield
State	NY
Postal Code	4361
Country	USA
Employee Telephone	
Job Title	Asset Manager
Employee Annual Salary . . .	45,000.00
Employee Start Date	26/04/09
Start Action	
Termination Date	01/01/01

You have now used three essential skills of screen enhancement:

- Simplification or replacement of long text
- Moving screen elements
- Hiding screen elements

Using these skills, you should be able to rearrange the 5250 screen elements so that the top of the screen looks something like this:



name	XHRRPGTRN_Maint
style	Edit Styles...
firstElementToFocus	Follow original 52
useTerminalStyles	□
disableTerminalKeys	□
limitedCursorSupport	□
disableAutoGUI	□
onArrive	Edit Script...
onLeave	Edit Script...

Department ADM Business Unit AM Employee ID A000090 Maintain Employee Information

Employee Title	Mr
Employee Surname	Allen
Given Names	Tobias
Employee Date of Birth . . .	06/01/84
Employee Gender	Male
Street	120-1451 Ultrices Rd
City	Pittsfield
State	NY
Postal Code	4361
Country	USA
Employee Telephone	
Job Title	Asset Manager
Employee Annual Salary . . .	45,000.00
Employee Start Date	26/04/09
Start Action	
Termination Date	01/01/01

Move the business unit label and field, the employee identification label and field, and the screen title into the bar beside department.

> The exact location of a customized screen element is visible in its basic style property. You can manually change these values to move elements around the screen.
>
> Sometimes the manual approach is an easier and more accurate way to very precisely align fields.

Save your changes.

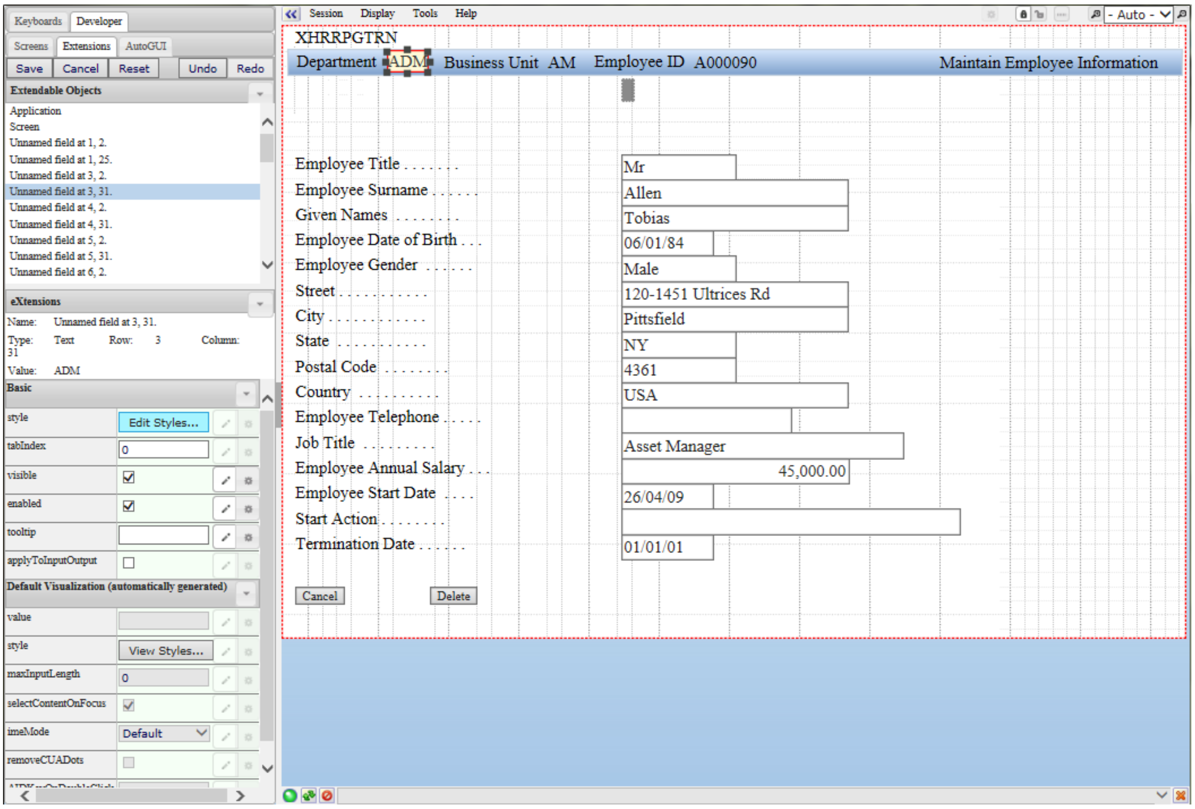
Applying Styles

The top bar of the 5250 screen now contains three pieces of key information (department, business unit and employee ID), and the screen title. These fields can now be associated with a style to give them their specific visual characteristics.

With the aXes Developer window and the XHRRPGTRN_Maint screen displayed.

Click on the department code "ADM" (or whatever code is displayed) to select it.

In the Basic properties sheet, click the Edit Styles button adjacent to the style property.



Application
Screen
Unnamed field at 1, 2.
Unnamed field at 1, 25.
Unnamed field at 3, 2.
Unnamed field at 3, 31.
Unnamed field at 4, 2.
Unnamed field at 4, 31.
Unnamed field at 5, 2.
Unnamed field at 5, 31.
Unnamed field at 6, 2.

eXtensions

Name: Unnamed field at 3, 31.
Type: Text Row: 3 Column: 31
Value: ADM

Basic

style	Edit Styles...
tabIndex	0
visible	☑
enabled	☑
tooltip	
applyToInputOutput	□

Default Visualization (automatically generated)

value	
style	View Styles...
maxInputLength	0
selectContentOnFocus	☑
imeMode	Default
removeCUADots	□

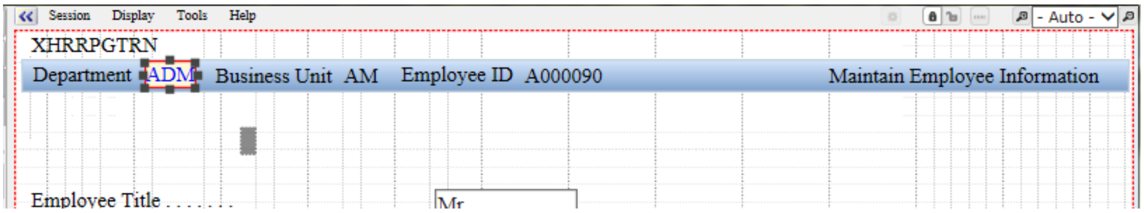
XHRRPGTRN

Department ADM Business Unit AM Employee ID A000090 Maintain Employee Information

Employee Title	Mr
Employee Surname	Allen
Given Names	Tobias
Employee Date of Birth . . .	06/01/84
Employee Gender	Male
Street	120-1451 Ultrices Rd
City	Pittsfield
State	NY
Postal Code	4361
Country	USA
Employee Telephone	
Job Title	Asset Manager
Employee Annual Salary . . .	45,000.00
Employee Start Date	26/04/09
Start Action	
Termination Date	01/01/01

Add a key information style entry at the top: **_base: KeyInformation;**

Click OK and save your changes. You can see the effect of the changes.



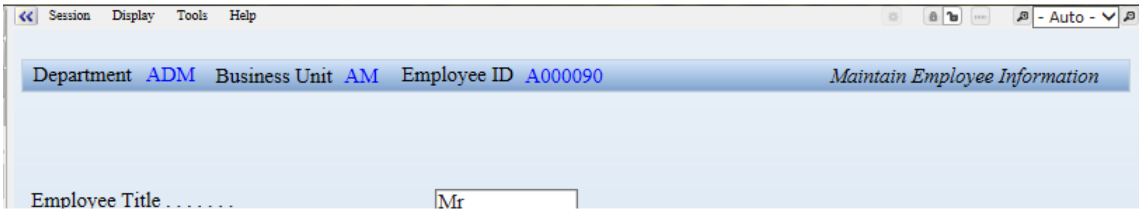
The colour of the Department Abbreviation text has changed to blue. You defined the colour when you created the KeyInformation style earlier in this tutorial.

Apply the KeyInformation style to the Business Unit field.

Apply the KeyInformation style to the Employee ID field.

Next, select the screen title and apply the style **_base: ScreenTitle;**

The top of the 5250 screen will look similar to this:



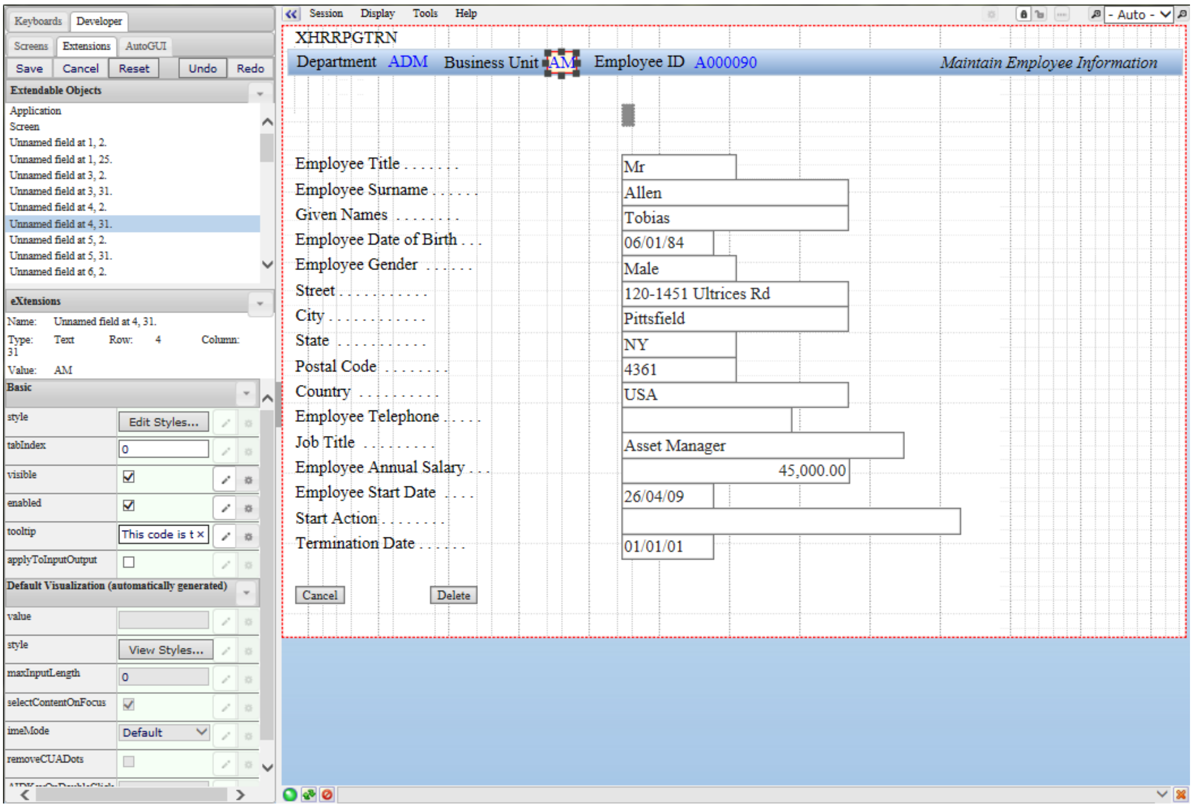
The text in the fields is blue and the screen title text is italic.

Using styles will save you time and effort. If you customize 20 screens and then decide that you want to change the screen title text to be italic and blue, you only have to change the style (i.e. the ScreenTitle style), not individual style properties in all 20 screens.

Tooltips

You can add hints or explanations to fields by adding a tooltip. Tooltips provide more meaningful details than a short or abbreviated label.

With the aXes Developer window and the XHRRPGTRN_Maint screen displayed, click on the **Business Unit** code, then choose customize and select **Tooltip** from the Add eXtensions list.



Extendable Objects

- Application
- Screen
- Unnamed field at 1, 2.
- Unnamed field at 1, 25.
- Unnamed field at 3, 2.
- Unnamed field at 3, 31.
- Unnamed field at 4, 2.
- Unnamed field at 4, 31.
- Unnamed field at 5, 2.
- Unnamed field at 5, 31.
- Unnamed field at 6, 2.

eXtensions

Name: Unnamed field at 4, 31.
Type: Text Row: 4 Column: 31
Value: AM

Basic

style	Edit Styles...
tabIndex	0
visible	☑
enabled	☑
tooltip	This code is t
applyToInputOutput	☐

Default Visualization (automatically generated)

value	
style	View Styles...
maxInputLength	0
selectContentOnFocus	☑
imeMode	Default
removeCUADots	☐

XHRRPGTRN

Department ADM Business Unit AM Employee ID A000090 *Maintain Employee Information*

Employee Title	Mr
Employee Surname	Allen
Given Names	Tobias
Employee Date of Birth . . .	06/01/84
Employee Gender	Male
Street	120-1451 Ultrices Rd
City	Pittsfield
State	NY
Postal Code	4361
Country	USA
Employee Telephone	
Job Title	Asset Manager
Employee Annual Salary . . .	45,000.00
Employee Start Date	26/04/09
Start Action	
Termination Date	01/01/01

Select the tooltip property in the Basic property sheet, and set it to this text (copy and paste from this):

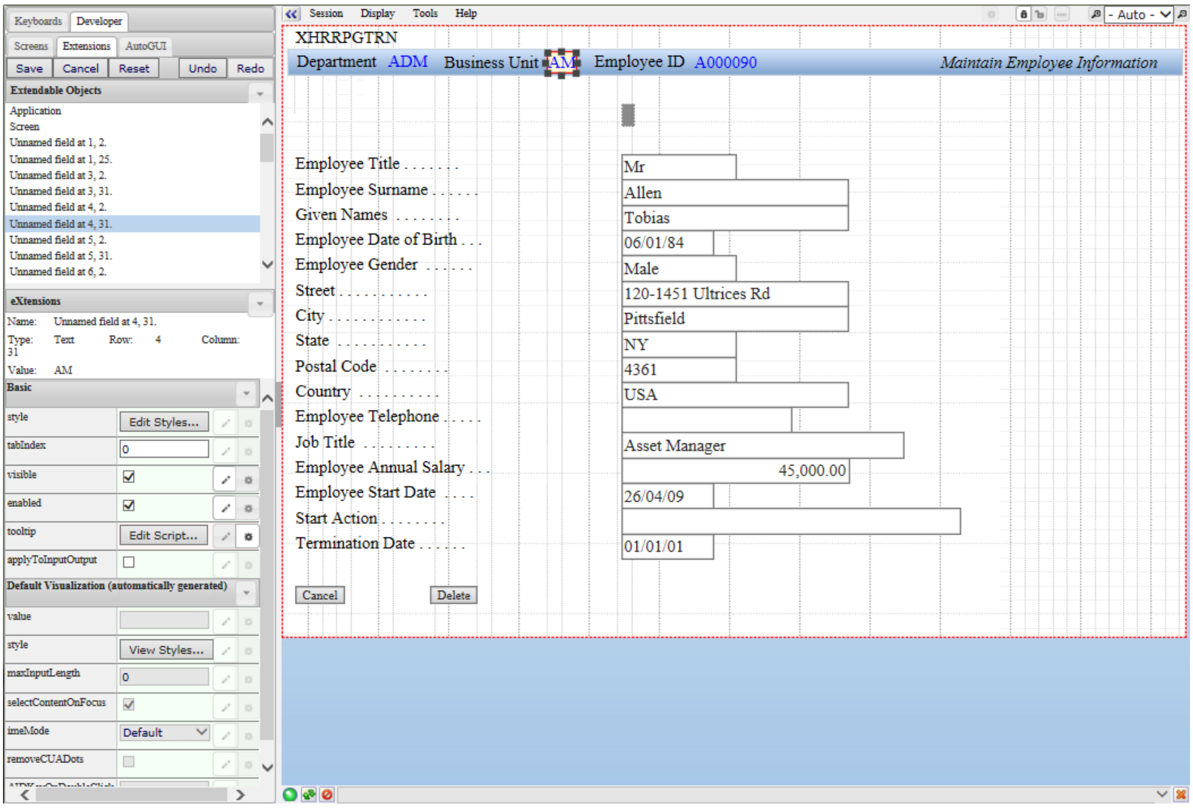
```
This code is the business unit that the employee currently
works for. It is often referred to as their "BU" code. Their id
badges must always display this code or they may be refused
admittance to company premises.
```

This tooltip is just plain text. When you add the text, the property field should have the pencil symbol on the right hand side. The gear symbol will be greyed.

You can change the type of property value you wish to enter by clicking on the gear symbol on the right hand side of the tooltip property. The gear symbol indicates that this property will be evaluated by executing the script that is displayed.

If you want to dynamically create the tooltip by executing a script, then the script you need for this example is:

```
ENV.returnValue = "This code is the business unit that the
employee currently works for. It is often referred to as their
\"BU\" code. Their id badges must always display this code or
they may be refused admittance to company premises.";
```



Extendable Objects

Application
Screen
Unnamed field at 1, 2.
Unnamed field at 1, 25.
Unnamed field at 3, 2.
Unnamed field at 3, 31.
Unnamed field at 4, 2.
Unnamed field at 4, 31.
Unnamed field at 5, 2.
Unnamed field at 5, 31.
Unnamed field at 6, 2.

eXtensions

Name: Unnamed field at 4, 31.
Type: Text Row: 4 Column: 31
Value: AM

Basic

style	Edit Styles...
tabIndex	0
visible	☑
enabled	☑
tooltip	Edit Script...
applyToInputOutput	☐

Default Visualization (automatically generated)

value	
style	View Styles...
maxInputLength	0
selectContentOnFocus	☑
imeMode	Default
removeCUADots	☐

XHRRPGTRN

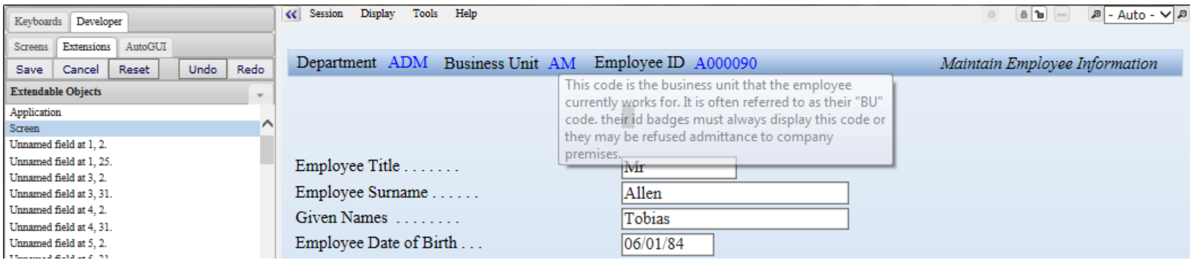
Department ADM Business Unit AM Employee ID A000090 *Maintain Employee Information*

Employee Title	Mr
Employee Surname	Allen
Given Names	Tobias
Employee Date of Birth . . .	06/01/84
Employee Gender	Male
Street	120-1451 Ultrices Rd
City	Pittsfield
State	NY
Postal Code	4361
Country	USA
Employee Telephone	
Job Title	Asset Manager
Employee Annual Salary . . .	45,000.00
Employee Start Date	26/04/09
Start Action	
Termination Date	01/01/01

Cancel Delete

Save your changes.

When you hover the mouse over the business unit code, you will now see the tooltip:



Extendable Objects

Application
Screen
Unnamed field at 1, 2.
Unnamed field at 1, 25.
Unnamed field at 3, 2.
Unnamed field at 3, 31.
Unnamed field at 4, 2.
Unnamed field at 4, 31.
Unnamed field at 5, 2.

Department ADM Business Unit AM Employee ID A000090 *Maintain Employee Information*

This code is the business unit that the employee currently works for. It is often referred to as their "BU" code. their id badges must always display this code or they may be refused admittance to company premises.

Employee Title	Mr
Employee Surname	Allen
Given Names	Tobias
Employee Date of Birth . . .	06/01/84

Managing long text in tooltips

If you require a lot of text for your tooltips (i.e. hints or explanations), create the tooltips in a word processing tool (e.g. Microsoft Word) where you can perform a spelling and grammar check.

When the text is ready, copy and paste it into your screen definitions, or use CTEXT("Tooltip.0001") type scripting references to externalize the definitions and multilingual translations.

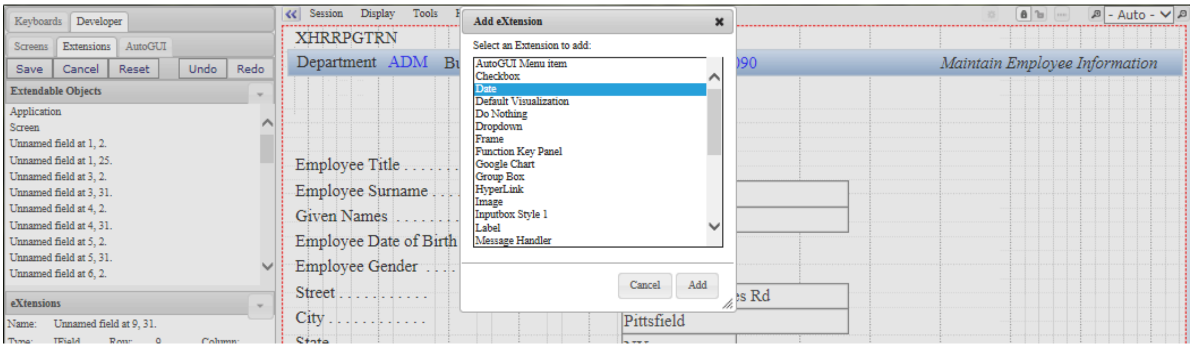
> You must use a Unicode capable text editor when editing multilingual strings.
>
> The multilingual section of this tutorial provides more details.

Dates

This task adds a calendar date picker to the date fields.

With the aXes Developer window and the XHRRPGTRN_Maint screen displayed, click **Employee Date of Birth**, remove the default visualization, and choose Customize.

Select **Date** from the Add eXtension list.



Application
Screen
Unnamed field at 1, 2.
Unnamed field at 1, 25.
Unnamed field at 3, 2.
Unnamed field at 3, 31.
Unnamed field at 4, 2.
Unnamed field at 4, 31.
Unnamed field at 5, 2.
Unnamed field at 5, 31.
Unnamed field at 6, 2.

Name: Unnamed field at 9, 31.

XHRRPGTRN

Department ADM

Maintain Employee Information

Employee Title . . . .
Employee Surname . . . . .
Given Names . . . .
Employee Date of Birth
Employee Gender . . . .
Street . . . .
City . . . .

Pittsfield

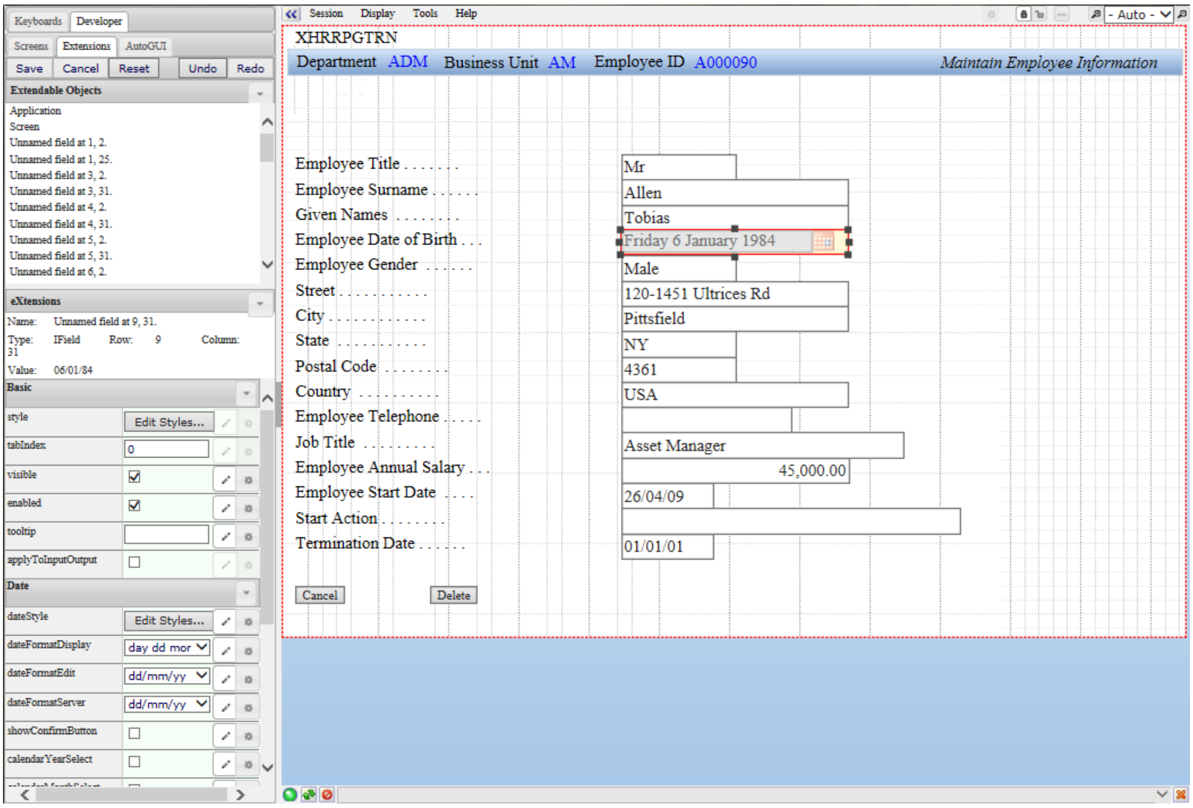
Add eXtension

Select an Extension to add:

AutoGUI Menu item
Checkbox
Date
Default Visualization
Do Nothing
Dropdown
Frame
Function Key Panel
Google Chart
Group Box
HyperLink
Image
Inputbox Style 1
Label
Message Handler

Cancel Add

Click the Add button and the date eXtension will appear on the screen.



Application
Screen
Unnamed field at 1, 2.
Unnamed field at 1, 25.
Unnamed field at 3, 2.
Unnamed field at 3, 31.
Unnamed field at 4, 2.
Unnamed field at 4, 31.
Unnamed field at 5, 2.
Unnamed field at 5, 31.
Unnamed field at 6, 2.

Name: Unnamed field at 9, 31.
Type: IField Row: 9 Column: 31
Value: 06/01/84

Basic

style	Edit Styles...
tabIndex	0
visible	☑
enabled	☑
tooltip	
applyToInputOutput	□

Date

dateStyle	Edit Styles...
dateFormatDisplay	day dd mor
dateFormatEdit	dd/mm/yy
dateFormatServer	dd/mm/yy
showConfirmButton	□
calendarYearSelect	□

XHRRPGTRN

Department ADM Business Unit AM Employee ID A000090

Maintain Employee Information

Employee Title	Mr
Employee Surname	Allen
Given Names	Tobias
Employee Date of Birth . . .	Friday 6 January 1984
Employee Gender	Male
Street	120-1451 Ultrices Rd
City	Pittsfield
State	NY
Postal Code	4361
Country	USA
Employee Telephone	
Job Title	Asset Manager
Employee Annual Salary . . .	45,000.00
Employee Start Date	26/04/09
Start Action	
Termination Date	01/01/01

Cancel Delete

In the date property sheet, change the **dateFormatDisplay** and the **dateFormatServer** property values.

dateFormatDisplay	day dd month yyyy
dateFormatServer	dd/mm/yy

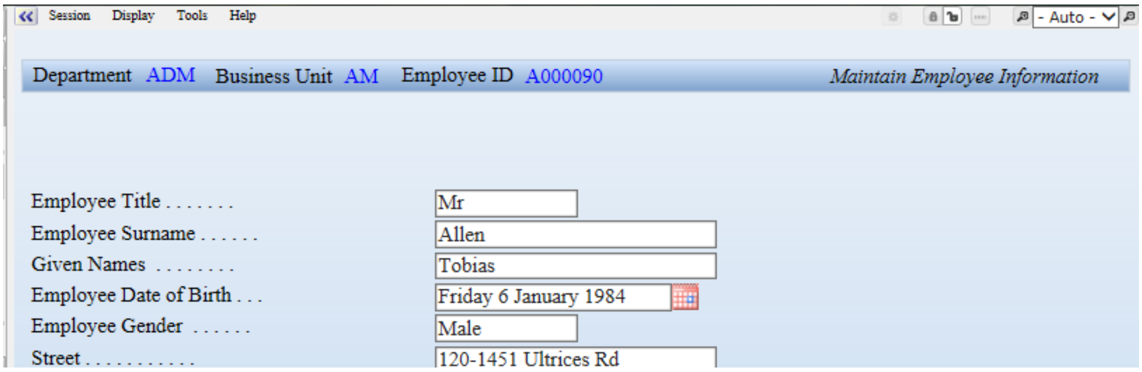
The dateFormatServer property must match the format of the date as it appears on the uncustomized screen. We used dd/mm/yy to match the date format shown on the uncustomized screen used in the tutorial. On your machine, the format may be different.

> If you cannot find a format that matches the format used to display your dates, you can change the dateFormatServer property to a script, and set it to any valid jQuery date format, using the script:
>
> ENV.returnValue = "dd-mm-y";
>
> See http://docs.jquery.com/UI/Datepicker/formatDate for valid jQuery date formats.

Widen the field to the right so that you can see the full date.

Save your changes.

The Employee Date of Birth field is now displayed like this:



Department ADM Business Unit AM Employee ID A000090 *Maintain Employee Information*

Employee Title	Mr
Employee Surname	Allen
Given Names	Tobias
Employee Date of Birth . . .	Friday 6 January 1984
Employee Gender	Male
Street	120-1451 Ultrices Rd

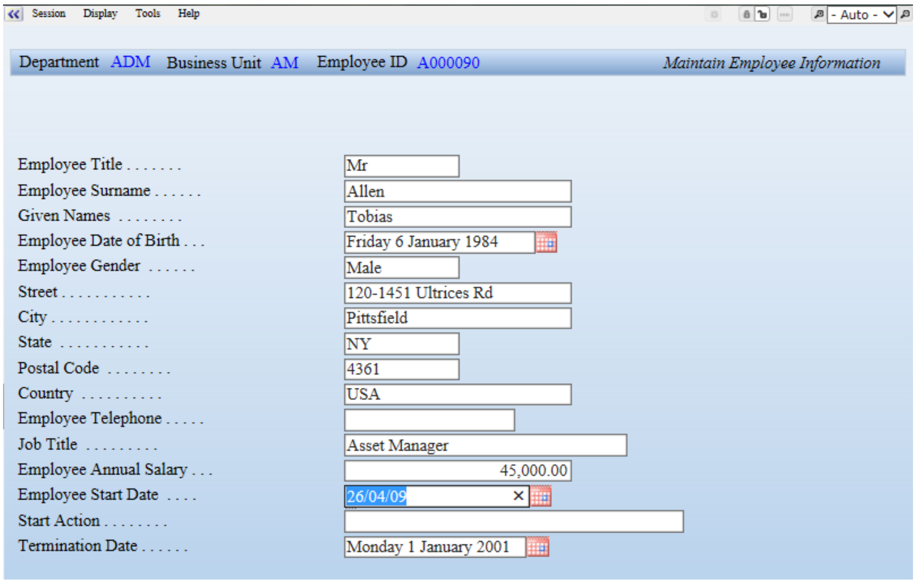
The date is displayed in a long format and has a date picker (calendar) that can be invoked by clicking the red calendar icon.

> You can control date options using their various properties.
>
> The defaults for the various date formats can be set by changing the global defaults used by the Date eXtension or even programmatically by scripting. You can do this for most properties in most eXtensions. Changing the default value for a property in an eXtension is more efficient than changing the property repeatedly.

Repeat this process for the Employee Start Date and the Termination Date.

Save your screen changes.

After the changes, the screen will look something like this:



Department ADM Business Unit AM Employee ID A000090 *Maintain Employee Information*

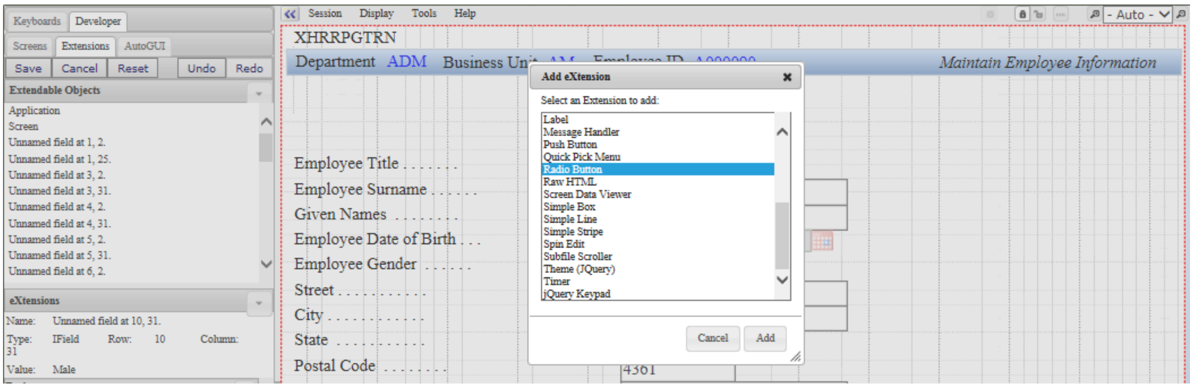
Employee Title	Mr
Employee Surname	Allen
Given Names	Tobias
Employee Date of Birth . . .	Friday 6 January 1984
Employee Gender	Male
Street	120-1451 Ultrices Rd
City	Pittsfield
State	NY
Postal Code	4361
Country	USA
Employee Telephone	
Job Title	Asset Manager
Employee Annual Salary . . .	45,000.00
Employee Start Date	26/04/09
Start Action	
Termination Date	Monday 1 January 2001

Radio Buttons

This task will transform the Employee Gender field into radio buttons.

With the aXes Developer window and the XHRRPGTRN_Maint screen displayed, click **Employee Gender**, remove the default visualization, and choose Customize.

Select **Radio Button** from the Add eXtension list.



Name: Unnamed field at 10, 31.
Type: IField Row: 10 Column: 31
Value: Male

XHRRPGTRN
Department ADM Business Unit AM Employee ID A000090 *Maintain Employee Information*

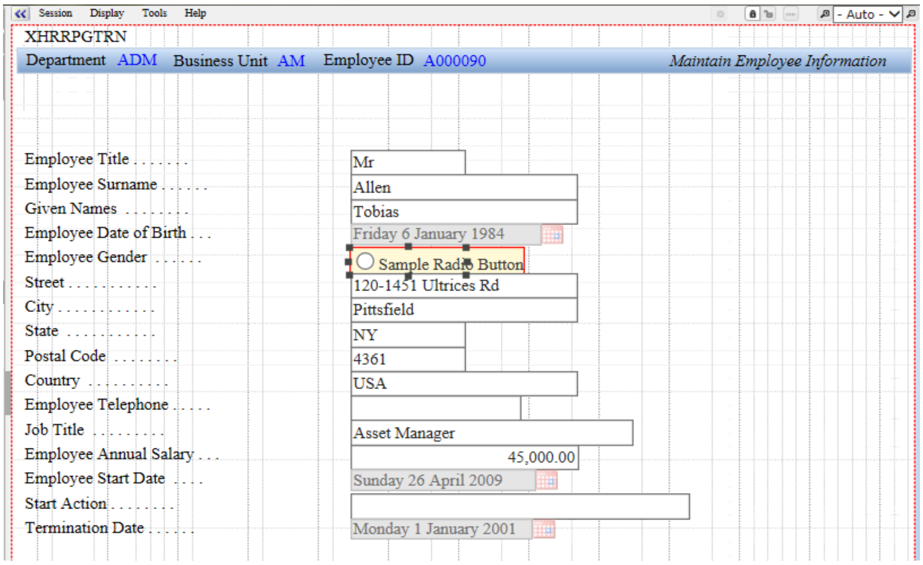
Employee Title . . . .
Employee Surname . . . . . .
Given Names . . . .
Employee Date of Birth . . .
Employee Gender . . . . . .
Street . . . .
City . . . .
State . . . .
Postal Code . . . .

Select an Extension to add:

- Label
- Message Handler
- Push Button
- Quick Pick Menu
- Radio Button
- Raw HTML
- Screen Data Viewer
- Simple Box
- Simple Line
- Simple Stripe
- Spin Edit
- Subfile Scroller
- Theme (JQuery)
- Timer
- jQuery Keypad

Click the Add button.

aXes will insert a radio button extension and the screen will look like this:



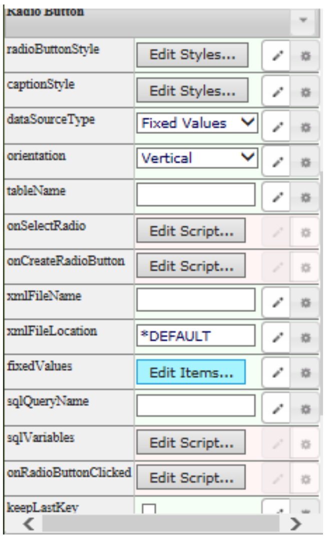
XHRRPGTRN
Department ADM Business Unit AM Employee ID A000090 *Maintain Employee Information*

Employee Title	Mr
Employee Surname	Allen
Given Names	Tobias
Employee Date of Birth . . .	Friday 6 January 1984
Employee Gender	○ Sample Radio Button
Street	120-1451 Ultrices Rd
City	Pittsfield
State	NY
Postal Code	4361
Country	USA
Employee Telephone	
Job Title	Asset Manager
Employee Annual Salary . . .	45,000.00
Employee Start Date	Sunday 26 April 2009
Start Action	
Termination Date	Monday 1 January 2001

Next, we need a value for each radio button.

Radio button values can be created from multiple types of input sources. In this case, we are going to use the simplest source, a table of fixed values.

Look at radio button property sheet:



Radio Button

radioButtonStyle	Edit Styles...
captionStyle	Edit Styles...
dataSourceType	Fixed Values
orientation	Vertical
tableName	
onSelectRadio	Edit Script...
onCreateRadioButton	Edit Script...
xmlFileName	
xmlFileLocation	*DEFAULT
fixedValues	Edit Items...
sqlQueryName	
sqlVariables	Edit Script...
onRadioButtonClicked	Edit Script...
keepLastKey	□

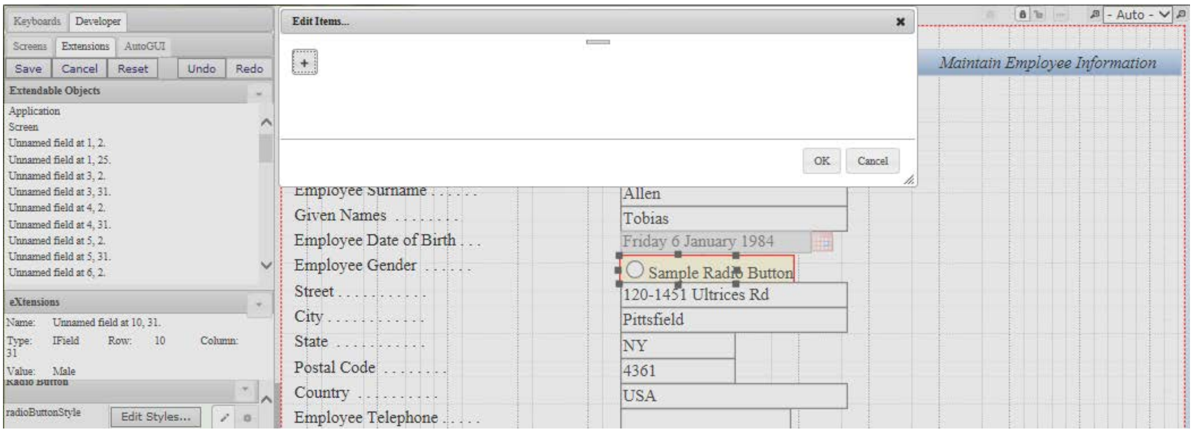
Change the value of the **dataSourceType** property to **Fixed Values**.

Change the value of the **orientation** property from vertical to **Horizontal**.

Then, click the **Edit Items** button adjacent to the **fixedValues** property.

The Edit Items window will appear where you will define a value and label text for each radio button.

The empty Edit Items window looks like this:



Name: Unnamed field at 10, 31.
Type: IField Row: 10 Column: 31
Value: Male

Radio Button
radioButtonStyle Edit Styles...

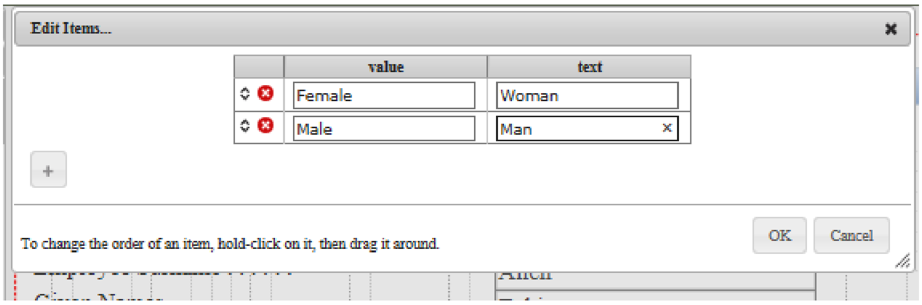
Employee Surname . . . . . . Allen
Given Names . . . . Tobias
Employee Date of Birth . . . Friday 6 January 1984
Employee Gender . . . . . . ○ Sample Radio Button
Street . . . . 120-1451 Ultrices Rd
City . . . . Pittsfield
State . . . . NY
Postal Code 4361
Country . . . . USA
Employee Telephone . . . . .

The value determines what will be checked for and put into the 5250 entry field by the radio button. The label text determines what is displayed beside the radio button.

Click the plus button [+] in the Edit Items window and add two items:

Item	Value (inserted in the real 5250 field)	Text (displayed beside the radio button)
1	Male	Man
2	Female	Woman

After you finish the Edit Item windows should look like this:



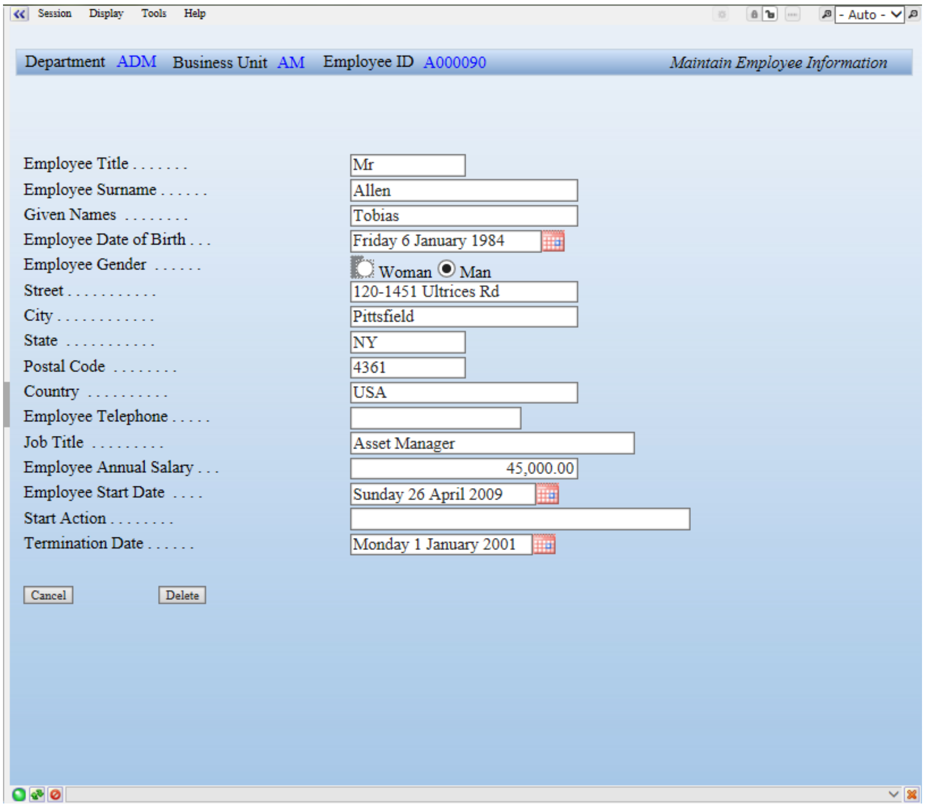
value	text
Female	Woman
Male	Man

To change the order of an item, hold-click on it, then drag it around.

Click OK to close the Edit Items window.

Save your changes.

Your screen should look something like this:



Department ADM Business Unit AM Employee ID A000090 *Maintain Employee Information*

Employee Title	Mr
Employee Surname	Allen
Given Names	Tobias
Employee Date of Birth . . .	Friday 6 January 1984
Employee Gender	○ Woman ● Man
Street	120-1451 Ultrices Rd
City	Pittsfield
State	NY
Postal Code	4361
Country	USA
Employee Telephone	
Job Title	Asset Manager
Employee Annual Salary . . .	45,000.00
Employee Start Date	Sunday 26 April 2009
Start Action	
Termination Date	Monday 1 January 2001

Drop down lists

This task will transform the State and Country fields into drop downs (i.e. list boxes).

The data used to fill drop downs can come from static tables, dynamic tables (created by SQL commands), XML documents or simple static lists.

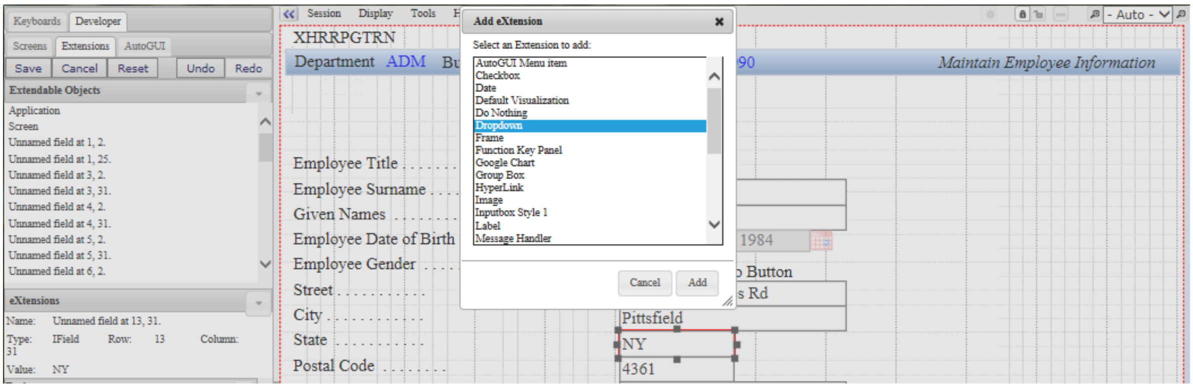
> A separate tutorial covers the possibilities for supplying data to fill drop down lists and ways that you can use them.

In this tutorial you are going to use server based static tables shipped with aXes as the data source for the drop down lists.

State field drop down

With the aXes Developer window and the XHRRPGTRN_Maint screen displayed, click **State**, remove the default visualization, and choose Customize.

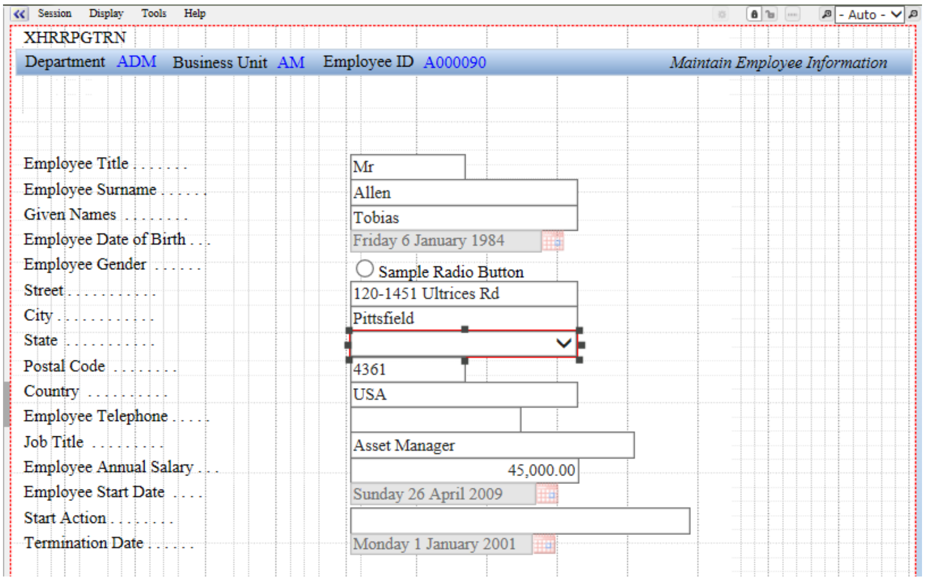
Select **Dropdown** from the Add eXtension list.



Name: Unnamed field at 13, 31.
Type: IField Row: 13 Column: 31
Value: NY

Click the Add button.

aXes will insert a dropdown extension and the screen will look like this:



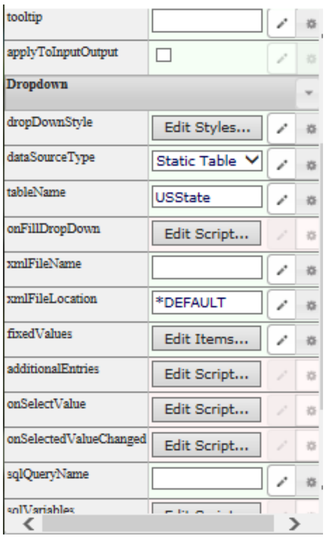
XHRRPGTRN

Department ADM Business Unit AM Employee ID A000090 *Maintain Employee Information*

Employee Title	Mr
Employee Surname	Allen
Given Names	Tobias
Employee Date of Birth . . .	Friday 6 January 1984
Employee Gender	○ Sample Radio Button
Street	120-1451 Ultrices Rd
City	Pittsfield
State	
Postal Code	4361
Country	USA
Employee Telephone	
Job Title	Asset Manager
Employee Annual Salary . . .	45,000.00
Employee Start Date	Sunday 26 April 2009
Start Action	
Termination Date	Monday 1 January 2001

Next, we need to define a data source type and a data source.

Look at the dropdown property sheet:



tooltip	
applyToInputOutput	□
Dropdown	
dropDownStyle	Edit Styles...
dataSourceType	Static Table
tableName	USState
onFillDropDown	Edit Script...
xmlFileName	
xmlFileLocation	*DEFAULT
fixedValues	Edit Items...
additionalEntries	Edit Script...
onSelectValue	Edit Script...
onSelectedValueChanged	Edit Script...
sqlQueryName	
sqlVariables	Edit Script...

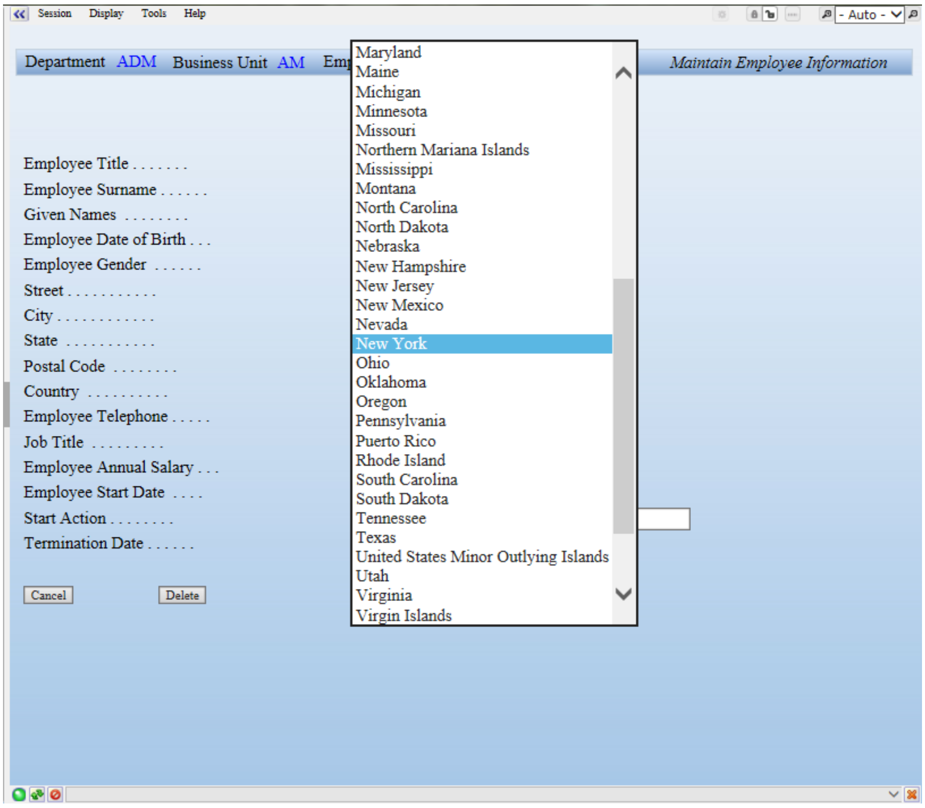
Change the value of the **dataSourceType** property to **Static Table**.

Change the value of the **tableName** propertyto **USState**.

Widen the state field on the 5250 screen so that the state names fit in the window width.

Save your changes.

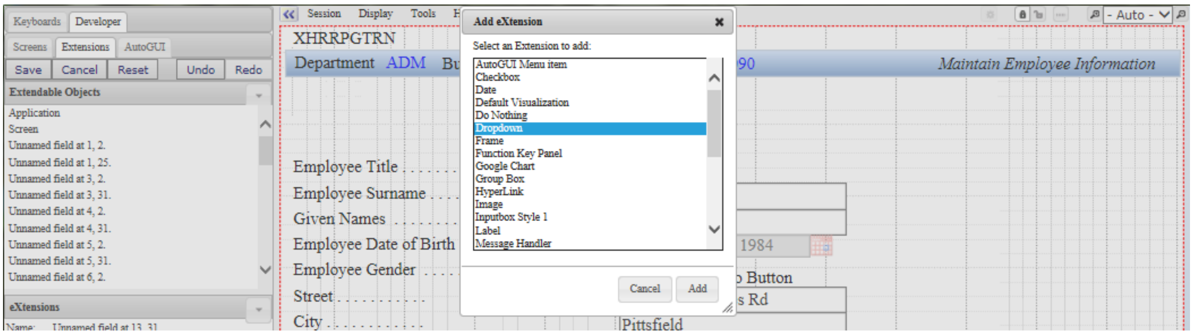
Your screen will now show a state code drop down like this:


Session
Display
Tools
Help
- Auto -
Department ADM
Business Unit AM
Emp
Maintain Employee Information
Employee Title . . . .
Employee Surname . . . . . .
Given Names . . . .
Employee Date of Birth . . .
Employee Gender . . . . . .
Street . . . .
City . . . .
State . . . .
Postal Code . . . .
Country . . . .
Employee Telephone . . . . .
Job Title . . . .
Employee Annual Salary . . .
Employee Start Date . . . .
Start Action . . . .
Termination Date . . . . . .
Cancel
Delete
Maryland
Maine
Michigan
Minnesota
Missouri
Northern Mariana Islands
Mississippi
Montana
North Carolina
North Dakota
Nebraska
New Hampshire
New Jersey
New Mexico
Nevada
New York
Ohio
Oklahoma
Oregon
Pennsylvania
Puerto Rico
Rhode Island
South Carolina
South Dakota
Tennessee
Texas
United States Minor Outlying Islands
Utah
Virginia
Virgin Islands

Country field drop down

With the aXes Developer window and the XHRRPGTRN_Maint screen displayed, click **Country**, remove the default visualization, and choose Customize.

Select **Dropdown** from the Add eXtension list.



Click the Add button.

aXes will insert a dropdown extension.

Next, we need to define a data source type and a data source.

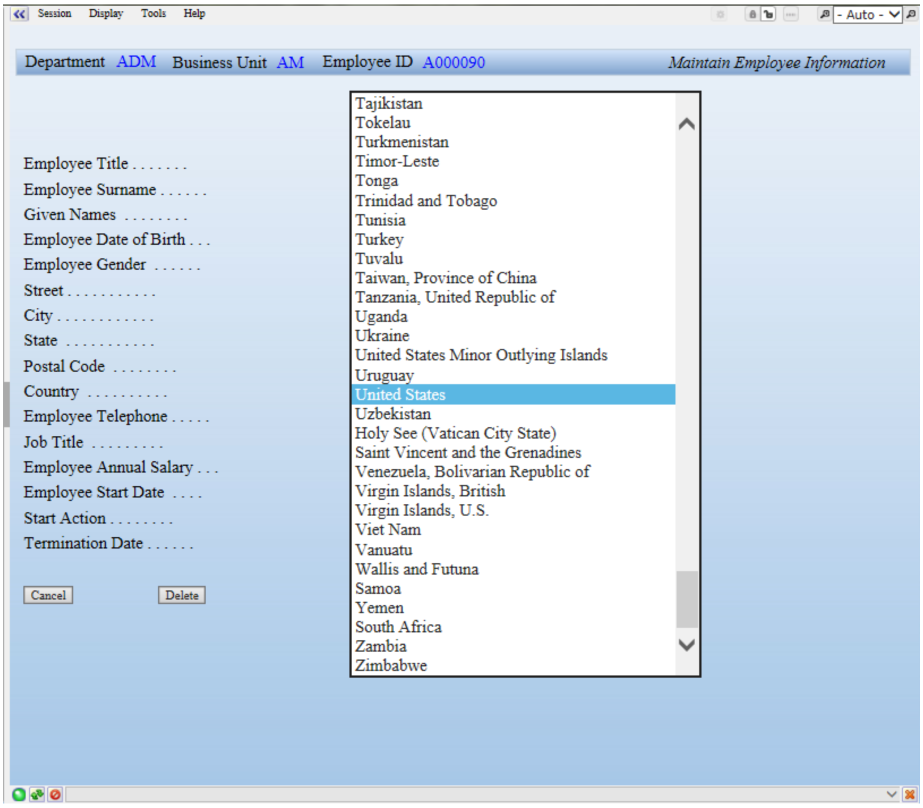
On the dropdown property sheet , change the value of the **dataSourceType** property to **Static Table**.

Change the value of the **tableName** property to **ISOCountry** (the table name is case sensitive, so ensure you use the exact spelling and upper and lower case).

Widen the state field on the 5250 screen so that the country names fit in the window width.

Save your changes.

Your screen will now show a state code drop down like this:



Department ADM Business Unit AM Employee ID A000090 *Maintain Employee Information*

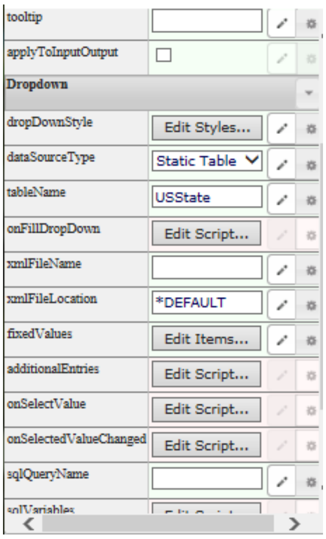
Employee Title . . . .
Employee Surname . . . . . .
Given Names . . . .
Employee Date of Birth . . .
Employee Gender . . . . . .
Street . . . .
City . . . .
State . . . .
Postal Code . . . .
Country . . . .
Employee Telephone . . . . .
Job Title . . . .
Employee Annual Salary . . .
Employee Start Date . . . .
Start Action . . . .
Termination Date . . . . . .

Tajikistan
Tokelau
Turkmenistan
Timor-Leste
Tonga
Trinidad and Tobago
Tunisia
Turkey
Tuvalu
Taiwan, Province of China
Tanzania, United Republic of
Uganda
Ukraine
United States Minor Outlying Islands
Uruguay
United States
Uzbekistan
Holy See (Vatican City State)
Saint Vincent and the Grenadines
Venezuela, Bolivarian Republic of
Virgin Islands, British
Virgin Islands, U.S.
Viet Nam
Vanuatu
Wallis and Futuna
Samoa
Yemen
South Africa
Zambia
Zimbabwe

Dropdowns have many options for sources of data, presenting the data, and can initiate actions when a user selects a drop down item. Using these features drop downs can be used to achieve clever and useful screen interactions. For example, imagine you want the country drop down to show the country code and the full name of the country.

Put the screen in edit mode and select the country code drop down.

Look at the dropdown property sheet:

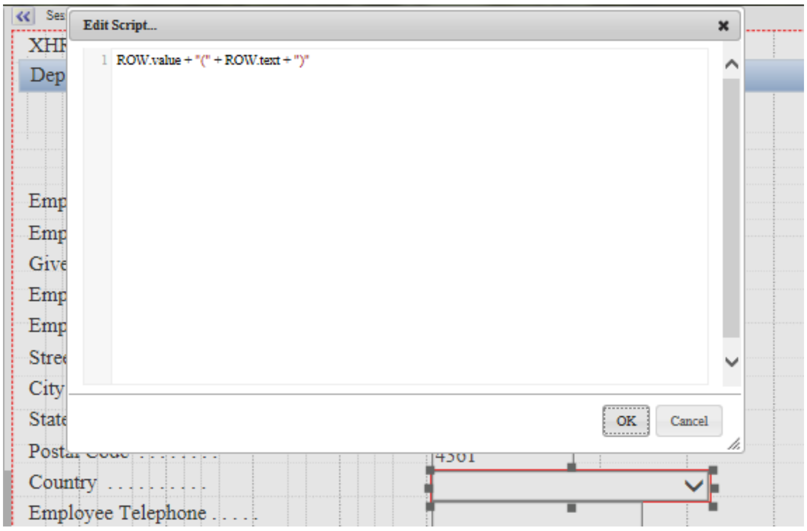


tooltip	
applyToInputOutput	□
Dropdown	
dropDownStyle	Edit Styles...
dataSourceType	Static Table
tableName	USState
onFillDropDown	Edit Script...
xmlFileName	
xmlFileLocation	*DEFAULT
fixedValues	Edit Items...
additionalEntries	Edit Script...
onSelectValue	Edit Script...
onSelectedValueChanged	Edit Script...
sqlQueryName	
sqlVariables	Edit Script...

Click the Edit Script button adjacent to the **onFillDropDown** property.

Copy this code into the Edit Script window.

```
ROW.value + " (" + ROW.text + ")"
```

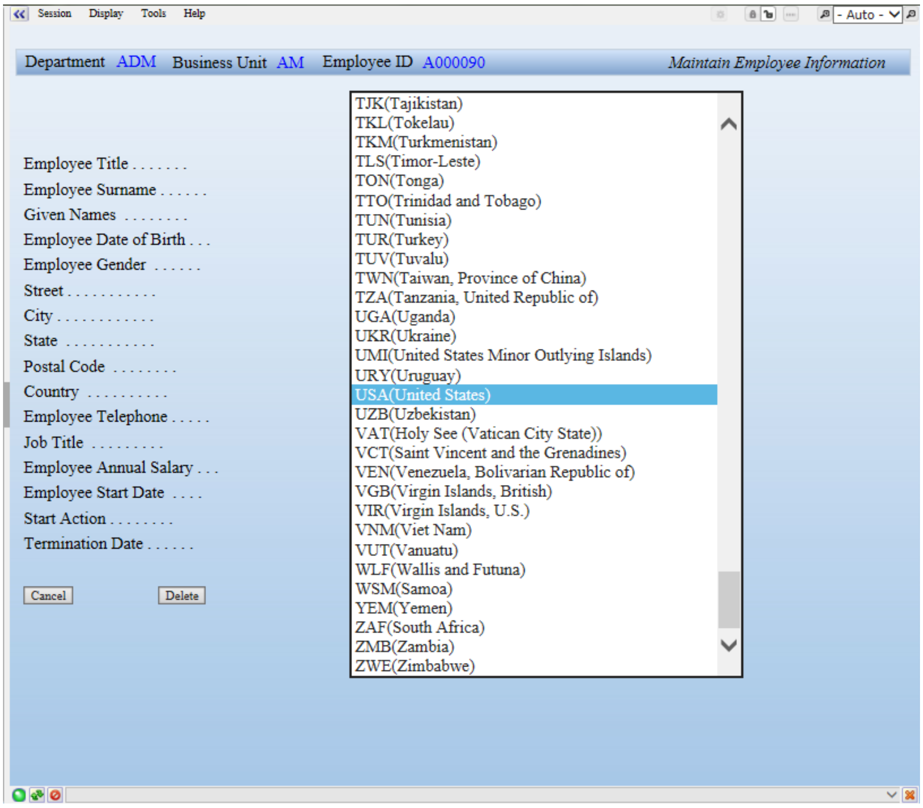


Edit Script...

```
ROW.value + " (" + ROW.text + ")"
```

Click OK and save your changes.

The Country drop down will now look like this:



Department ADM Business Unit AM Employee ID A000090 *Maintain Employee Information*

Employee Title . . . .
Employee Surname . . . . . .
Given Names . . . .
Employee Date of Birth . . .
Employee Gender . . . . . .
Street . . . .
City . . . .
State . . . .
Postal Code . . . .
Country . . . .
Employee Telephone . . . . .
Job Title . . . .
Employee Annual Salary . . .
Employee Start Date . . . .
Start Action . . . .
Termination Date . . . . . .

TJK(Tajikistan)
TKL(Tokelau)
TKM(Turkmenistan)
TLS(Timor-Leste)
TON(Tonga)
TTO(Trinidad and Tobago)
TUN(Tunisia)
TUR(Turkey)
TUV(Tuvalu)
TWN(Taiwan, Province of China)
TZA(Tanzania, United Republic of)
UGA(Uganda)
UKR(Ukraine)
UMI(United States Minor Outlying Islands)
URY(Uruguay)
USA(United States)
UZB(Uzbekistan)
VAT(Holy See (Vatican City State))
VCT(Saint Vincent and the Grenadines)
VEN(Venezuela, Bolivarian Republic of)
VGB(Virgin Islands, British)
VIR(Virgin Islands, U.S.)
VNM(Viet Nam)
VUT(Vanuatu)
WLF(Wallis and Futuna)
WSM(Samoa)
YEM(Yemen)
ZAF(South Africa)
ZMB(Zambia)
ZWE(Zimbabwe)

As you may imagine, this ability to script what happens as the drop down is filled and what happens when a user selects an item can be used in many clever and useful ways.

Using ROW and other scripting objects

In the preceding example the ROW scripting object was used. In scripts that process table information, it is shorthand scripting used to refer to the "current" ROW in the table.

There are other shorthand objects you can use in your scripting:

Objects	Description	Example of Common Methods

FIELD	The specific field bound to the extension.	FIELD.getValue()
		FIELD.setValue("ADM")

FIELDS	Gets a reference to any named field on the current screen.	FIELDS(“CustomerName”).getValue()
CTEXT	Shortcut for TEXT.cust	CTEXT(“label001”). Retrieves the customer text associated with the key value "label001".

Objects	Description	Example of Common Methods

SENDKEY	Send key	SENDKEY(“Enter”)
		SENDKEY("F4")

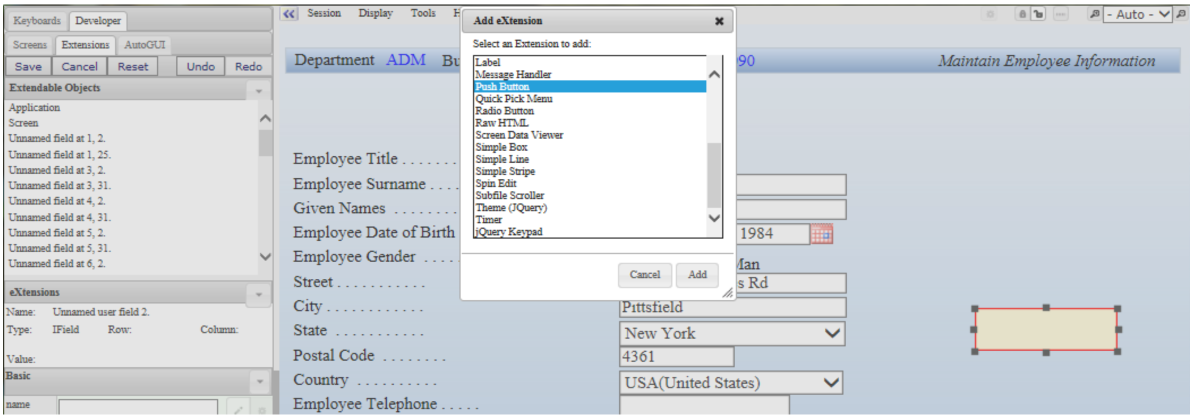
TRACE	Traces values into the trace window.	TRACE("Button 1 clicked")
DEBUG	Shortcut for AXES.Debug.output	
LANGUAGE	Current language code	

Push Buttons

This task will hide the function keys and replace them with push buttons.

With the aXes Developer window and the XHRRPGTRN_Maint screen displayed, click the Extendable Objects action button, then choose Add User Field from the Add eXtension window.

When aXes displays the list of eXtensions, choose **Push Button** and then click the Add button.



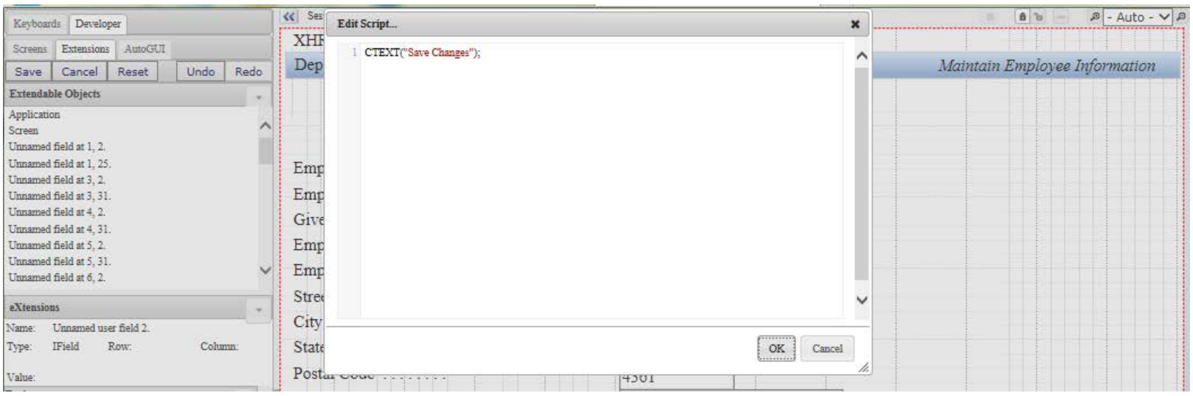
Repeat these steps to add a user field, choosing the push button eXtension, until you have three push buttons on the screen.

Select each button in turn and edit the script for the **caption** and **onClick** properties.

First Button	caption	`CTEXT("Save Changes");`
	onClick	`SENDKEY("Enter");`
Second Button	caption	`CTEXT("Cancel Changes");`
	onClick	`SENDKEY("F12");`
Third Button	caption	`CTEXT("Delete All Details");`
	onClick	`SENDKEY("F22");`

Click the Edit Script button adjacent to the property to open the Edit Script window. You can copy and paste the script from the table above.

This is an example of editing the script for the caption property of the Save Changes button.



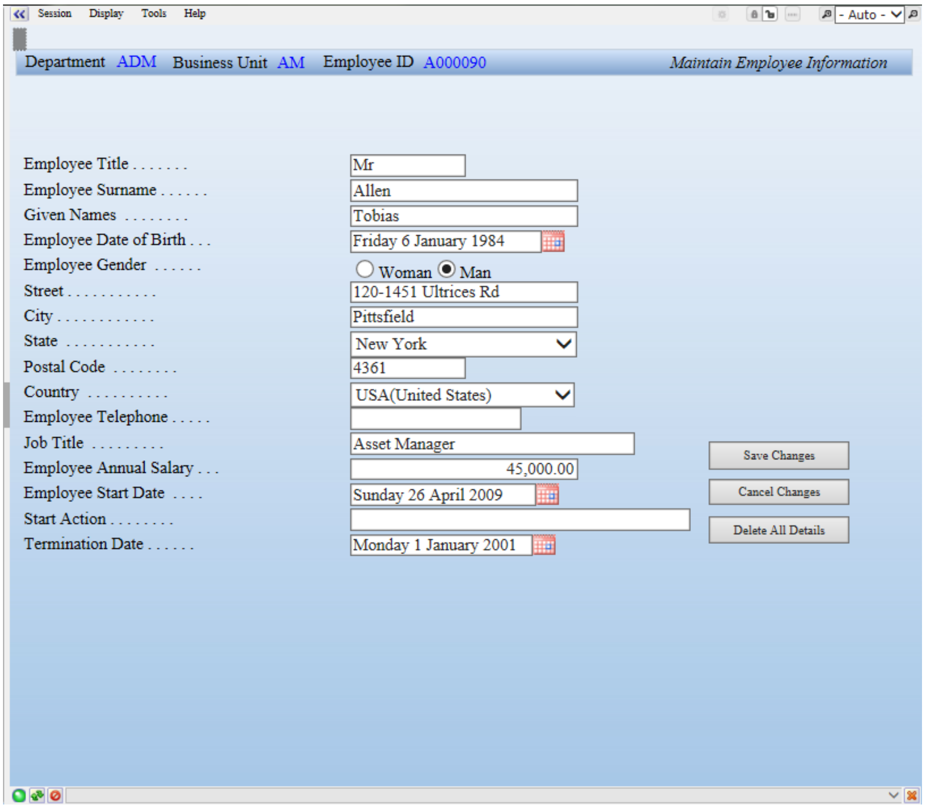
```
1  CTEXT("Save Changes");
```

Move the buttons to the bottom right-hand side of the screen.

Select the **F12=Cancel** object and hide it by unchecking its visible property.

Select the **F22=Delete** object and hide it by unchecking its visible property.

Your screen will look like this:



Department ADM Business Unit AM Employee ID A000090 *Maintain Employee Information*

Employee Title	Mr
Employee Surname	Allen
Given Names	Tobias
Employee Date of Birth . . .	Friday 6 January 1984
Employee Gender	○ Woman ● Man
Street	120-1451 Ultrices Rd
City	Pittsfield
State	New York
Postal Code	4361
Country	USA(United States)
Employee Telephone	
Job Title	Asset Manager
Employee Annual Salary . . .	45,000.00
Employee Start Date	Sunday 26 April 2009
Start Action	
Termination Date	Monday 1 January 2001

Save your changes.

The F12=Cancel and F22=Delete text at the bottom of the 5250 screen is screen text, just like the text "Employee Title". There is nothing special at all about this text. Its presence or absence in no way influences what function keys are actually enabled by the screen.

The Save button was added for the Enter function key. In most good GUI designs all function keys generally have an associated button.

Multi-lingual text

If you are not concerned about multi-lingual applications, skip to the next section.

You can use multi-lingual text for buttons, tooltips, labels, etc.

You must use a Unicode capable text editor when editing multi-lingual text. Almost all PC based text editors recognize and can handle Unicode files.

Typically, you can check this by looking at the “Save as” dialog options. For example, Notepad shows this in the Encoding option.

When you added the Save Changes button, you used the script, CTEXT("Save Changes"); to set the button's caption. This indicates that the button's caption is found by executing the CTEXT("Save Changes") scripting function.

The CTEXT() function looks up the customer multi-lingual text for the key "Save Changes" and returns the associated language text.

Customer multi-lingual text is stored in the **Texts_Cust_LL.txt** file. The **LL** represents a language code, e.g. English is en, French is fr, German is de, and Japanese is ja. The name of the multi-lingual text file for Japanese is **Texts_Cust_ja.txt.**

At execution time CTEXT("key") looks up the key supplied and returns a value from data loaded from the current language file. If no entry is found, the "key" is returned as the string.

See the deployment tutorial for more details about using multi-lingual text.

The technique of using CTEXT("key") for multilingual text can and should be used for literal text values everywhere in eXtensions intended for use in multi-lingual applications.

The choice of the CTEXT "key" here makes the button captions correct even when there are no key values in the Texts_Cust_LL.txt files. This is because the key and the English translations are the same values.

This approach is fine with short to medium text strings, but using CTEXT("Enter your password or click cancel to cancel and then log on again") would be not be advisable.

You should probably look to developing a "key" naming standard like CTEXT("Message_0001") or CTEXT("LongString.001") instead. This means that you have to provide English (or native) language translations of what Message_0001 (say) translates to.

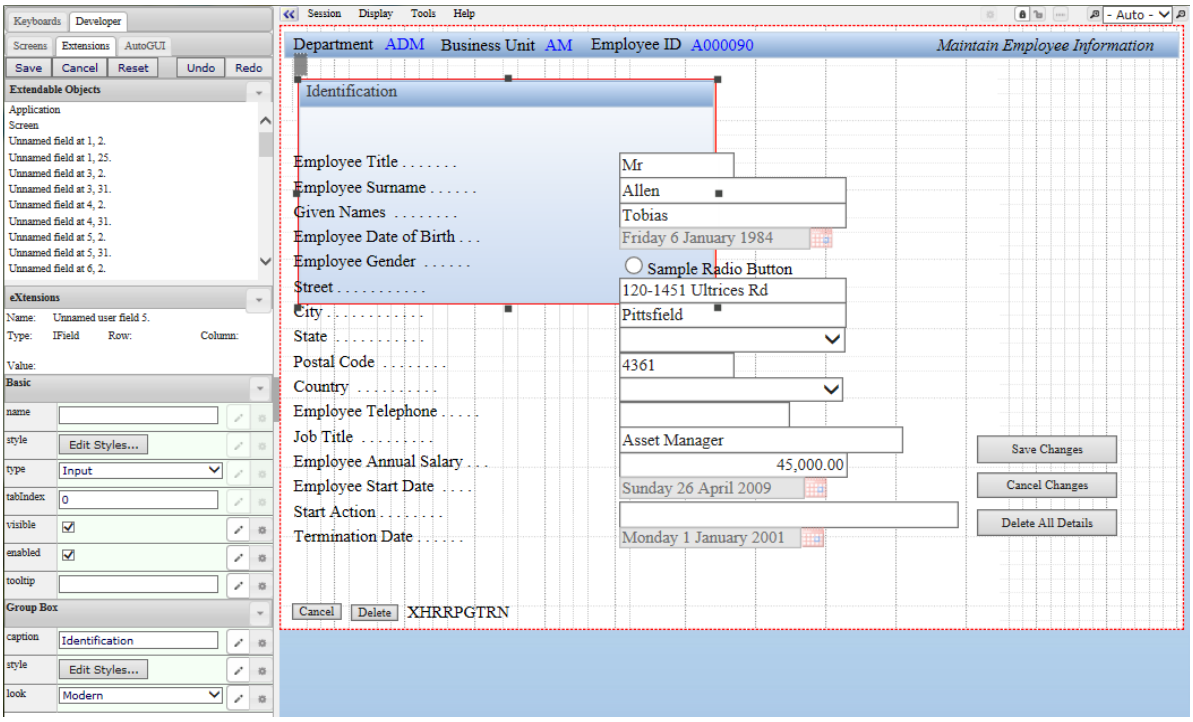
If you change a Texts_Cust_LL.txt file you should clear your browser cache to pick up the new file version.

Group boxes

This task will rearrange data on the 5250 screen into groups.

With the aXes Developer window and the XHRRPGTRN_Maint screen displayed, click the Extendable Objects action button, then choose Add User Field from the Add eXtension window.

When aXes displays the list of eXtensions, choose **Group Box** and then click the Add button.



Name: Unnamed user field 5.
Type: IField Row: Column:
Value:

name	
type	Input
tabIndex	0
visible	☑
enabled	☑
tooltip	
caption	Identification
look	Modern

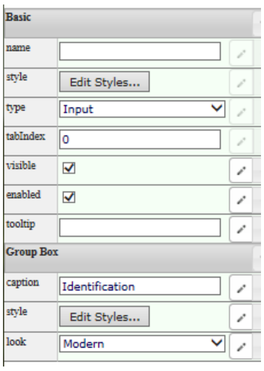
Department ADM Business Unit AM Employee ID A000090 *Maintain Employee Information*

Identification

Employee Title	Mr
Employee Surname	Allen
Given Names	Tobias
Employee Date of Birth . . .	Friday 6 January 1984
Employee Gender	○ Sample Radio Button
Street	120-1451 Ultrices Rd
City	Pittsfield
State	
Postal Code	4361
Country	
Employee Telephone	
Job Title	Asset Manager
Employee Annual Salary . . .	45,000.00
Employee Start Date	Sunday 26 April 2009
Start Action	
Termination Date	Monday 1 January 2001

XHRRPGTRN

Look at the Group Box property sheet:



name	
type	Input
tabIndex	0
visible	☑
enabled	☑
tooltip	
caption	Identification
look	Modern

Change the value of the **caption** property to **Identification**.

Change the value of the **look** property from Classic to **Modern**.

Save your changes.

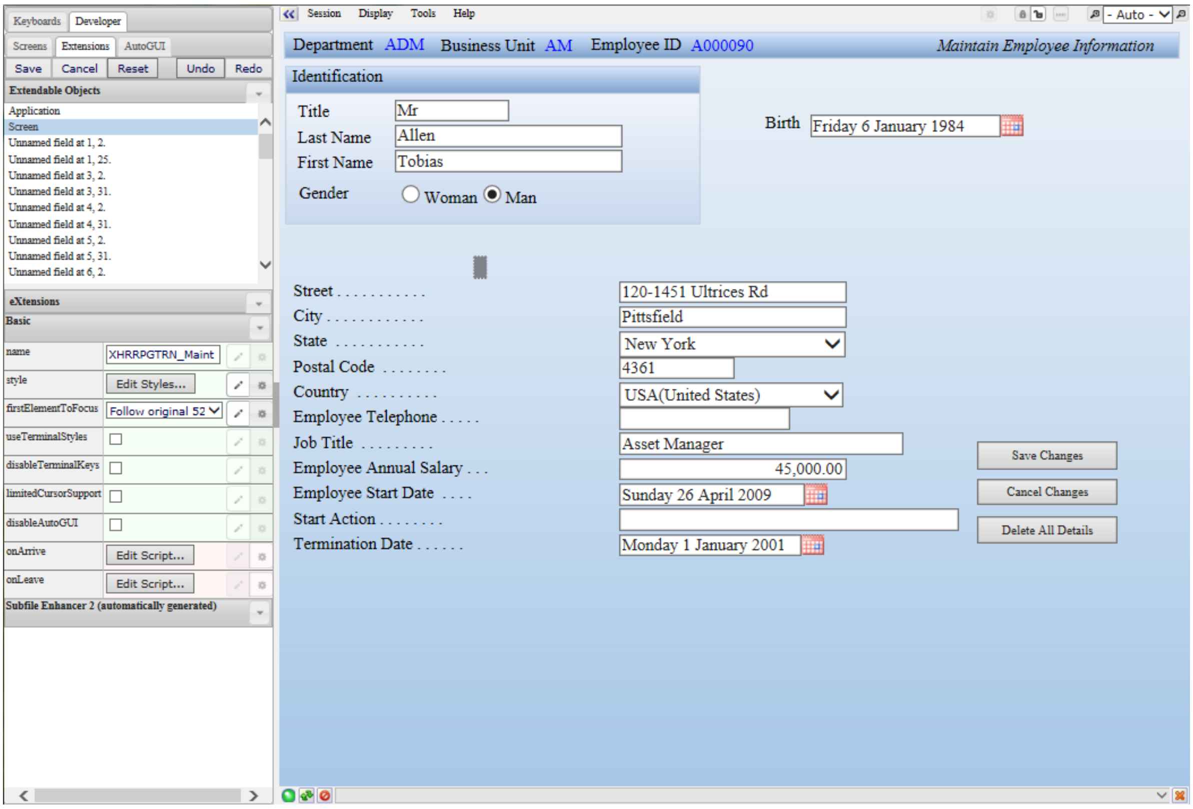
Tip: we can tell the group box to always use the Modern look by modifying the group box eXtension template. This way we don't need to adjust the look property for each group box we add.

> Please refer to FAQ – Customizing eXtensions for more information on how to customize an eXtension template.

Move the group box under the title bar, leaving a small gap, and approximately size it according to the about half the width of the screen plan.

Save your changes.

Now, move the employee identification information according to the plan. The employee identification information is title, last name, first name and gender. After you finish your screen should look something like this:



name: XHRRPGTRN_Maint
style: Edit Styles...
firstElementToFocus: Follow original 52
useTerminalStyles: □
disableTerminalKeys: □
limitedCursorSupport: □
disableAutoGUI: □
onArrive: Edit Script...
onLeave: Edit Script...

Department ADM Business Unit AM Employee ID A000090 *Maintain Employee Information*

Identification

Title: Mr
Last Name: Allen
First Name: Tobias
Gender: ○ Woman ● Man

Birth: Friday 6 January 1984

Street . . . . 120-1451 Ultrices Rd
City . . . . Pittsfield
State . . . . New York
Postal Code 4361
Country . . . . USA(United States)
Employee Telephone . . . . .
Job Title . . . . Asset Manager
Employee Annual Salary . . . 45,000.00
Employee Start Date . . . . Sunday 26 April 2009
Start Action . . . .
Termination Date . . . . . . Monday 1 January 2001

Save your changes.

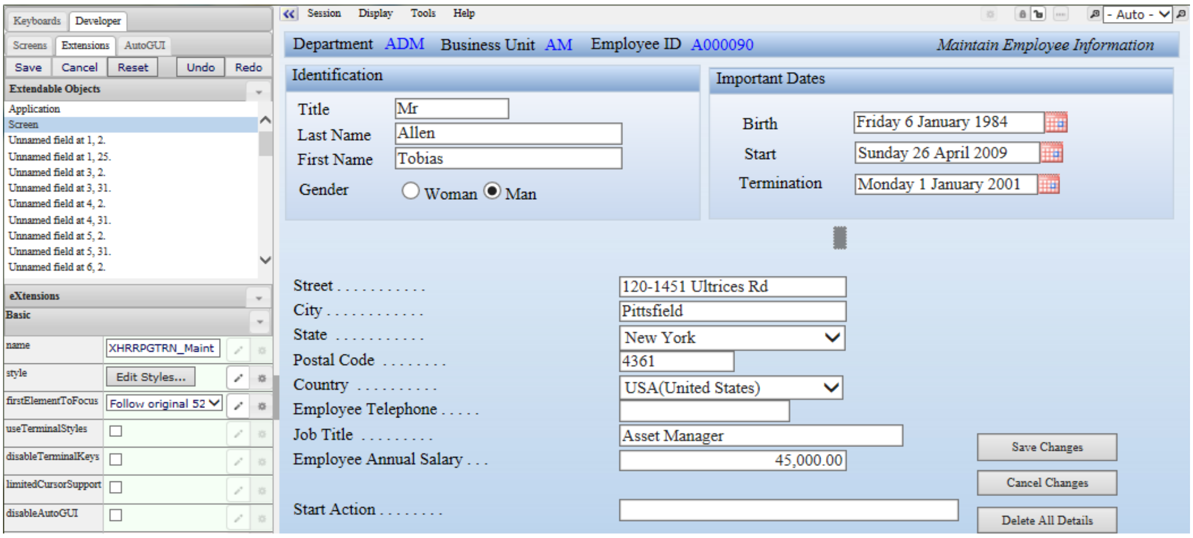
Add another group box and set its caption as **Important Dates**, and the look property to **Modern**, and move it to the right-hand side of the Identification group box.

Change the **Employee Date of Birth** label to **Birth**.

Change the **Employee Start Date** label to **Start**.

Change the **Termination Date** label to **Termination**.

Then arrange the date information according to the overall screen plan, something along these lines:



name: XHRRPGTRN_Maint
style: Edit Styles...
firstElementToFocus: Follow original 52
useTerminalStyles: □
disableTerminalKeys: □
limitedCursorSupport: □
disableAutoGUI: □

Department ADM Business Unit AM Employee ID A000090 *Maintain Employee Information*

Identification

Title: Mr
Last Name: Allen
First Name: Tobias
Gender: ○ Woman ● Man

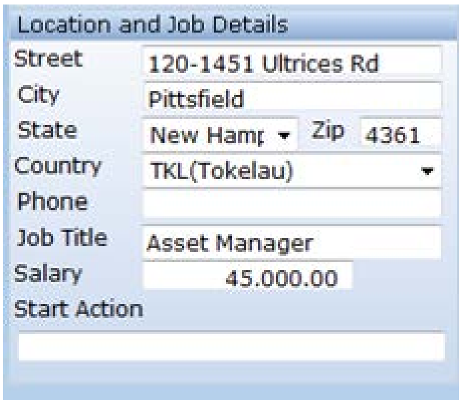
Important Dates

Birth: Friday 6 January 1984
Start: Sunday 26 April 2009
Termination: Monday 1 January 2001

Street . . . . 120-1451 Ultrices Rd
City . . . . Pittsfield
State . . . . New York
Postal Code 4361
Country . . . . USA(United States)
Employee Telephone . . . . .
Job Title . . . . Asset Manager
Employee Annual Salary . . . 45,000.00
Start Action . . . .

Save your changes.

Add another group box labelled **Location and Job Details**, and the look property to **Modern**, and position it on the left-hand side of the screen below the Identification group box.



Location and Job Details

Street: 120-1451 Ultrices Rd
City: Pittsfield
State: New Hamp Zip 4361
Country: TKL(Tokelau)
Phone:
Job Title: Asset Manager
Salary: 45.000.00
Start Action:

After you finish the group box will look like this.

Move these fields into the group box:

- Street
- City
- State
- Zip (Postal Code)
- Country
- Phone (Employee Telephone)
- Job Title
- Salary (Employee Annual Salary)
- Start Action

Do not obsess over the exact layout. You may notice that this layout is slightly different to the original.

Save your changes.

Images

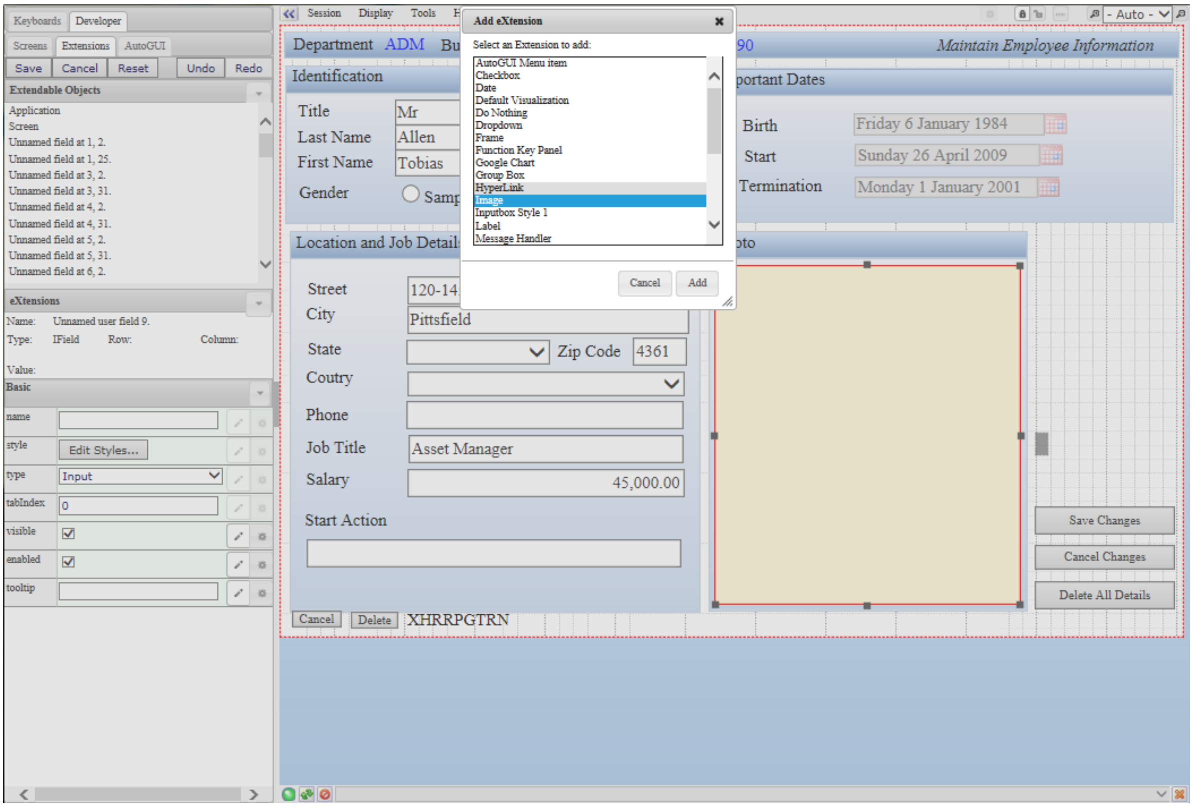
This task will add an image to the 5250 screen. You may notice that the image is not part of the original 5250 screen.

With the aXes Developer window and the XHRRPGTRN_Maint screen displayed, click the Extendable Objects action button, then choose Add User Field from the Add eXtension window.

When aXes displays the list of eXtensions, choose **Group Box** and then click the Add button.

Change the group box caption to **Photo**, and the look property to **Modern**, and move it to the right-hand side of the Location and Job Details group box.

Add another user field and choose the **Image** eXtension, and then click the Add button.



Name: Unnamed user field 9.
Type: IField Row: Column:
Value:
type: Input
tabIndex: 0
visible: ☑
enabled: ☑

Department ADM Bu ... 90
Title Mr
Last Name Allen
First Name Tobias
Gender ○ Samp
Birth Friday 6 January 1984
Start Sunday 26 April 2009
Termination Monday 1 January 2001
Street 120-14
City Pittsfield
State
Zip Code 4361
Coutry
Phone
Job Title Asset Manager
Salary 45,000.00
Start Action
XHRRPGTRN

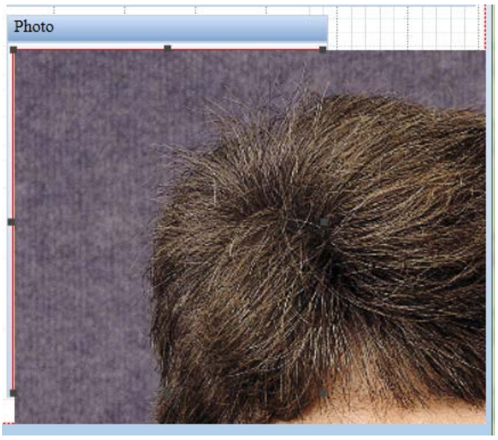
Select the image eXtension.

Change the **imagePath** property to **/ts/skins/images**.

Change the **imageName** property to **examplephoto.gif**.

Save your changes.

The image will appear in the Photo group box.

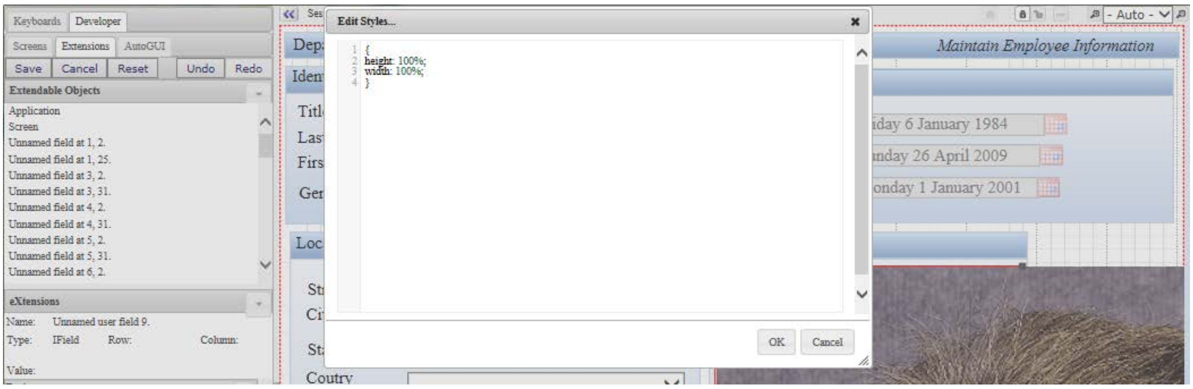


The image may appear to be very large.

This is because the image appears as its full size.

To shrink the image to the desired size, we need to change the height and width properties so that the browser will render the image to fit inside its container.

Click the **Edit Style** button in the **style** property of the image extension.



Name: Unnamed user field 9.
Type: IField Row: Column:
Value:

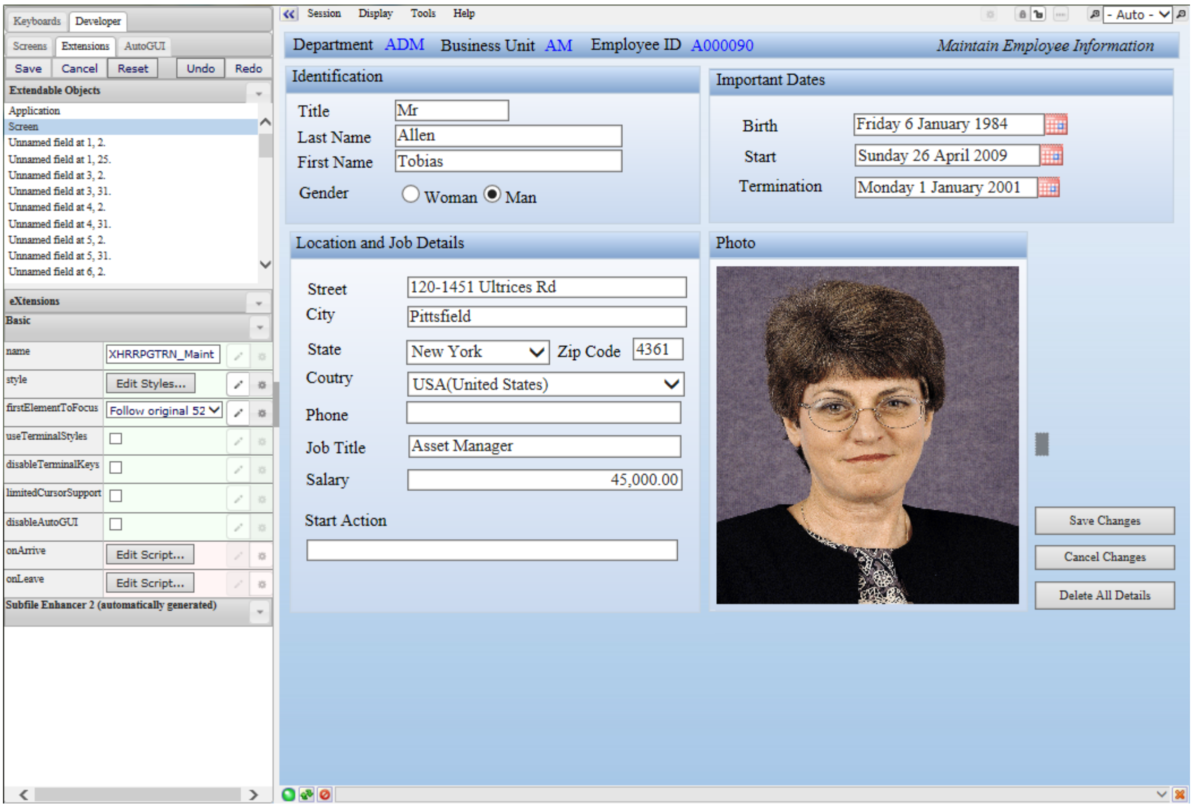
```
{
height: 100%;
width: 100%;
}
```

Copy this code into the Edit Styles window, then click OK.

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

This will ensure the image fits inside the Photo group box and not extend beyond the group box borders.

The completed screen will look similar to this:



Extendable Objects

- Application
- Screen
- Unnamed field at 1, 2.
- Unnamed field at 1, 25.
- Unnamed field at 3, 2.
- Unnamed field at 3, 31.
- Unnamed field at 4, 2.
- Unnamed field at 4, 31.
- Unnamed field at 5, 2.
- Unnamed field at 5, 31.
- Unnamed field at 6, 2.

eXtensions

Basic

name	XHRRPGTRN_Maint
style	Edit Styles...
firstElementToFocus	Follow original 52
useTerminalStyles	□
disableTerminalKeys	□
limitedCursorSupport	□
disableAutoGUI	□
onArrive	Edit Script...
onLeave	Edit Script...

Subfile Enhancer 2 (automatically generated)

Department ADM Business Unit AM Employee ID A000090 *Maintain Employee Information*

Identification

Title	Mr
Last Name	Allen
First Name	Tobias
Gender	○ Woman ● Man

Important Dates

Birth	Friday 6 January 1984
Start	Sunday 26 April 2009
Termination	Monday 1 January 2001

Location and Job Details

Street	120-1451 Ultrices Rd
City	Pittsfield
State	New York
Zip Code	4361
Coutry	USA(United States)
Phone	
Job Title	Asset Manager
Salary	45,000.00
Start Action	

Photo

The ability to script the image properties is useful. Often the image name needs to be dynamically determined by using information present on the screen, such as a product number or an employee number. You may even have to execute an SQL request to look up a database table to find the name of the image.

Hyperlinks

This task will add hyperlinks to the 5250 screen.

With the aXes Developer window and the XHRRPGTRN_Maint screen displayed, click the Extendable Objects action button, then choose Add User Field from the Add eXtension window.

When aXes displays the list of eXtensions, choose **Group Box** and then click the Add button.

Change the group box caption to **Online Documents**, and move it to the right-hand side of the Photo group box.

Add another user field and choose **HyperLink**, then change the caption to **Employment Contract**, and move it into the Online Documents group box.

There is a small problem here. Since we don't want the text to wrap around on two lines the font size needs to be reduced. You can do this in two ways:

Option 1:	Change the style font-size of the hyperlink to 7pt or 8pt
Option 2:	Define a new application-wide style named SmallHyperLink (say) that defines and standardizes the visual characteristics of a small hyperlink.

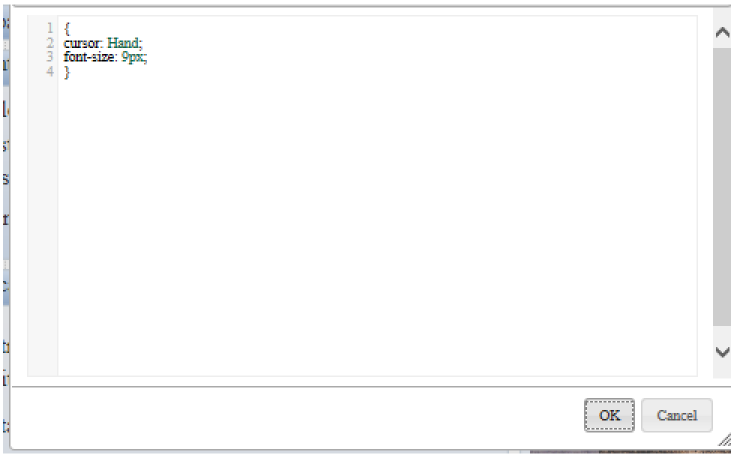
If you choose option 1, you will have to change the font-size for every small hyperlink.

If you choose option 2, you need to edit your application properties and define a new style with properties, something like this:

name	SmallHyperLink
styleFor	Application
style	font-size: 7pt;

Refer to the earlier section in this tutorial describing how to create styles.

Edit the style property of the **Employment Contract** hyperlink.



```
{
cursor: Hand;
font-size: 9px;
}
```

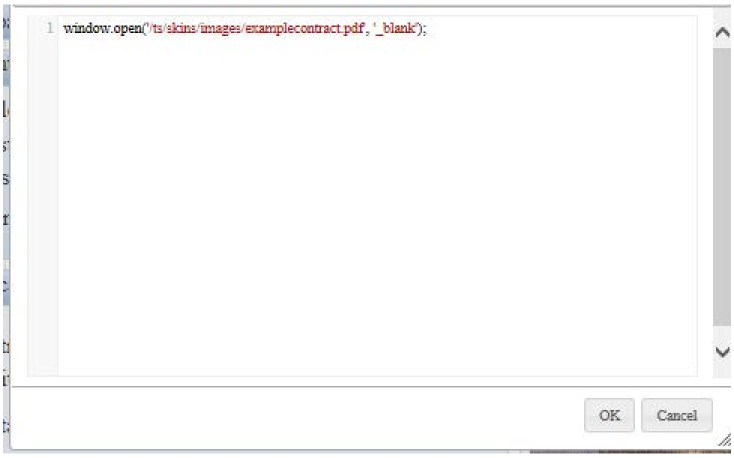
Copy this code into the Edit Styles window, then click OK:

```
{
cursor: Hand;
font-size: 9pt;
}
```

Next, you need to specify what happens when a user clicks the hyperlink.

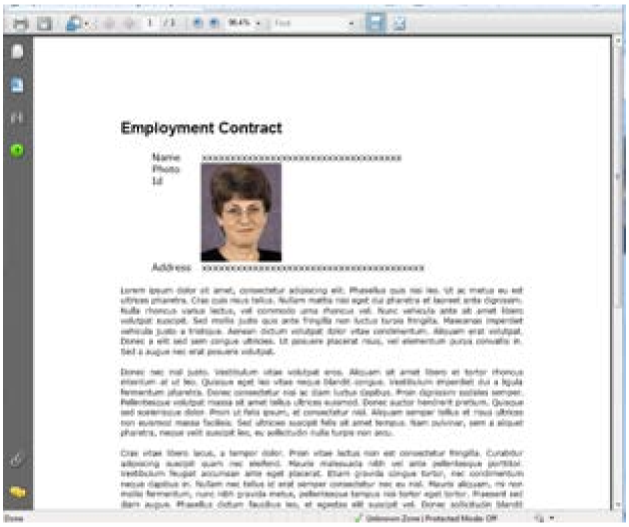
Edit the script for the onClick property of the **Employment Contract** hyperlink, and copy the this code into the Edit Script window, then click OK.

```
{
window.open('/ts/skins/images/examplecontract.pdf','_blank');
}
```



```
window.open('/ts/skins/images/examplecontract.pdf', '_blank');
```

When clicked, the **Employment Contract** hyperlink will open a PDF document.



Employment Contract

Name xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
Photo
Id

Address xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus quis nisi leo. Ut ac metus eu est ultrices pharetra. Cras quis risus tellus. Nullam mattis nisi eget dui pharetra et laoreet ante dignissim. Nulla rhoncus varius lectus, vel commodo urna rhoncus vel. Nunc vehicula ante sit amet libero volutpat suscipit. Sed mollis justo quis ante fringilla non luctus turpis fringilla. Maecenas imperdiet vehicula justo a tristique. Aenean dictum volutpat dolor vitae condimentum. Aliquam erat volutpat. Donec a elit sed sem congue ultrices. Ut posuere placerat risus, vel elementum purus convallis in. Sed a augue nec erat posuere volutpat.

Donec nec nisl justo. Vestibulum vitae volutpat eros. Aliquam sit amet libero et tortor rhoncus interdum at ut leo. Quisque eget leo vitae neque blandit congue. Vestibulum imperdiet dui a ligula fermentum pharetra. Donec consectetur nisi ac diam luctus dapibus. Proin dignissim sodales semper. Pellentesque volutpat massa sit amet tellus ultrices euismod. Donec auctor hendrerit pretium. Quisque sed scelerisque dolor. Proin ut felis ipsum, et consectetur nisl. Aliquam semper tellus et risus ultrices non euismod massa facilisis. Sed ultrices suscipit felis sit amet tempus. Nam pulvinar, sem a aliquet pharetra, neque velit suscipit leo, eu sollicitudin nulla turpis non arcu.

Cras vitae libero lacus, a tempor dolor. Proin vitae lectus non est consectetur fringilla. Curabitur adipiscing suscipit quam nec eleifend. Mauris malesuada nibh vel ante pellentesque porttitor. Vestibulum feugiat accumsan ante eget placerat. Etiam gravida congue tortor, nec condimentum neque dapibus in. Nullam nec tellus id erat semper consectetur nec eu nisl. Mauris aliquam, mi non mollis fermentum, nunc nibh gravida metus, pellentesque tempus nisi tortor eget tortor. Praesent sed diam augue. Phasellus dictum faucibus leo, et egestas elit suscipit vel. Donec sollicitudin blandit

Add another user field, choose HyperLink, and then click the Add button.

Move the hyperlink into the Online Documents group box and change its caption to **Latest Time Sheet**.

Edit the style property of the **Latest Time Sheet** hyperlink and copy this code into the Edit Styles window, then click OK.

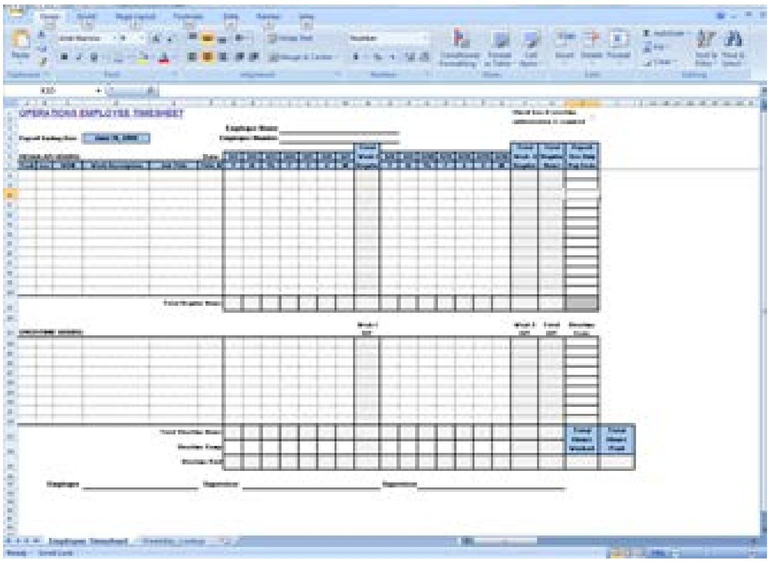
```
{
cursor: hand;
font-size: 9pt;
}
```

Edit the script for the onClick property of the **Latest Time Sheet** hyperlink, and copy the this code into the Edit Script window, then click OK.

```
{
window.open('/ts/skins/images/exampletimesheet.xls','_blank');
}
```

Save your changes.

When clicked, the **Latest Time Sheet** hyperlink will open an Excel workbook.



OPERATIONS EMPLOYEE TIMESHEET

Employee Name:

Employee Number:

Finish the screen customization

This task will add one more group box to the 5250.

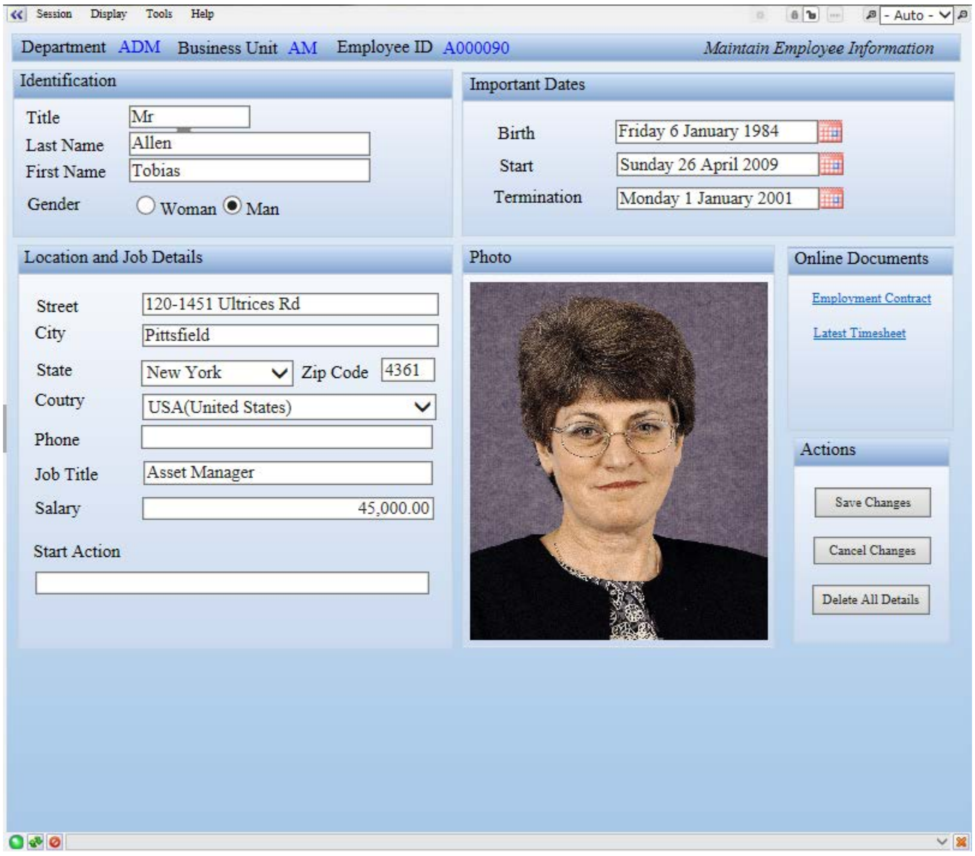
With the aXes Developer window and the XHRRPGTRN_Maint screen displayed, click the Extendable Objects action button, then choose Add User Field from the Add eXtension window.

When aXes displays the list of eXtensions, choose **Group Box** and then click the Add button.

Change the value of the **caption** property to **Actions**.

Change the value of the **look** property to **Modern**.

The finished screen will look something like this:



Department ADM Business Unit AM Employee ID A000090 *Maintain Employee Information*

Identification

Title Mr
Last Name Allen
First Name Tobias
Gender ○ Woman ● Man

Important Dates

Birth Friday 6 January 1984
Start Sunday 26 April 2009
Termination Monday 1 January 2001

Location and Job Details

Street 120-1451 Ultrices Rd
City Pittsfield
State New York Zip Code 4361
Coutry USA(United States)
Phone
Job Title Asset Manager
Salary 45,000.00
Start Action

Photo

Online Documents

Employment Contract
Latest Timesheet

Actions

This screen is not exactly the same as the original plan as some minor improvements have been made along the way.

Testing customizations

Always test your customizations as a user.

You can test your screen customizations while logged on to aXes as a developer, but you must always also test your application while logged on as a user.

To test as a user, open the Projects Home Page.

Select your project, and then use the **Work as a User (aXes-TS2)** option to start aXes.

Add library AXESDEMO to your library list and call program XHRRPGTRN using the buttons you created in the previous tutorial.

Select one of the displayed employees with an X and press enter.

Verify that the customizations display correctly, try out the buttons and hyperlinks.

aXes Documentation Library

eXtensions Tutorial 4 - Tracing and Debugging Techniques

You must complete Tutorial 3 first

Tutorial 3 is assumed knowledge for this tutorial.
If you have not completed tutorial 3 please do so before attempting this tutorial.

Note

When testing your scripts, they may frequently crash. If this happens, you should not just start a new session in the browser. You should close the browser window(s), and start the browser again. This allows the browser to free up any resources left in use at the time of the failure.

Using alert()

When your scripts don't work as expected, the simplest form of debugging is to add an alert to them.

Suppose you wanted to check a value being used in your onClick logic for a button:

Open the *Projects Home Page*. Select your project and then use the *Work as Developer* option.

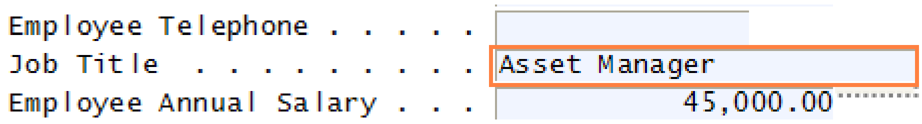
Sign on to aXes and then to a 5250 session.

Display the System i Main Menu.

Start the axes demo, and go to any employee's details.

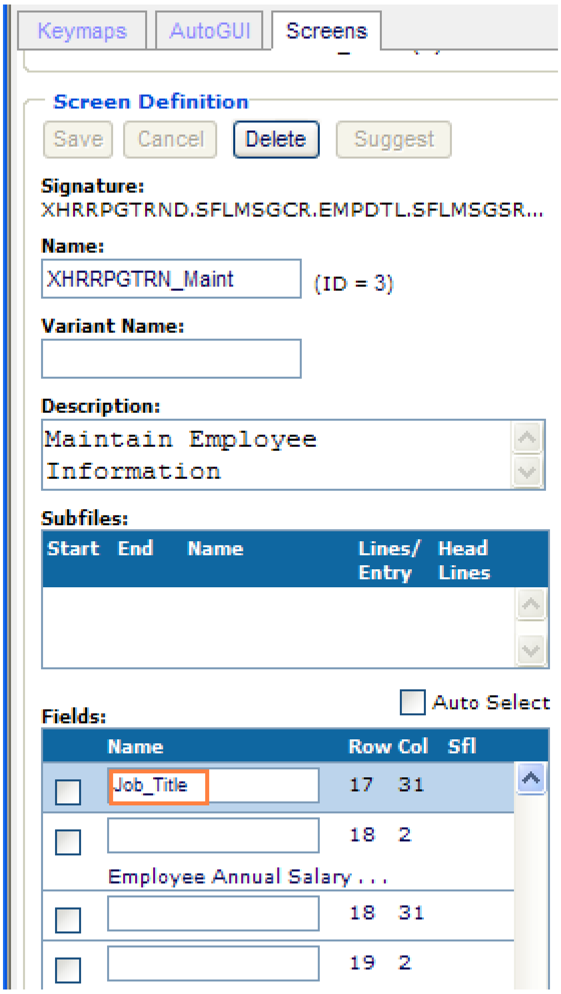
Next you need to identify a field on the screen so that you can debug its value. To do this, name the Job Title field *Job_Title* in the Screens tab like this:

Click on the job title field:



```
Employee Telephone . . . . . .
Job Title  . . . . . . . . . .  Asset Manager
Employee Annual Salary . . .            45,000.00
```

Display the Screens tab and note that the field is automatically selected. Give it the name *Job_Title* and click the Save button:



Screen Definition

Signature:
XHRRPGTRND.SFLMSGCR.EMPDTL.SFLMSGSR...

Name:
XHRRPGTRN_Maint (ID = 3)

Variant Name:

Description:
Maintain Employee Information

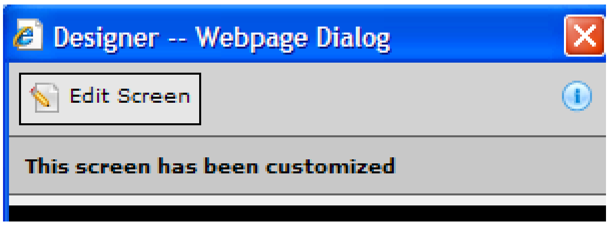
Subfiles:

Start	End	Name	Lines/ Entry	Head Lines

Fields: □ Auto Select

	Name	Row	Col	Sfl
□	Job_Title	17	31	
□		18	2	
	Employee Annual Salary . . .			
□		18	31	
□		19	2	

Now edit the screen in the aXes Designer window:



This screen has been customized

Add a new element.

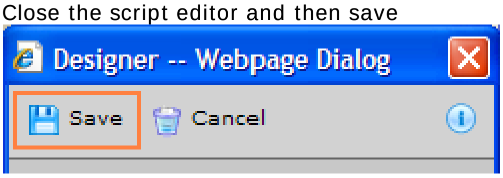
Check the Push Button eXtension.

Give the button the caption *Debug with an alert*

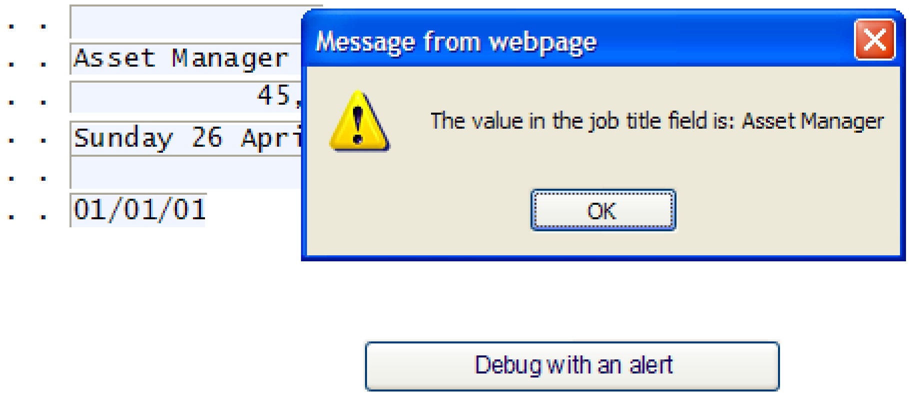
Now edit the onClick property. Replace the code with this:

```
var F = FIELDS("Job_Title");
var value = F.getValue();
alert("The value in the job title field is: " + value );
```

Close the script editor and then save



Now, if you click on the button, you can see the value being processed, in the alert window.



The value in the job title field is: Asset Manager

Debug Basics

Instead of using alert, you can use a DEBUG statement. This produces a similar result, but it puts the output from all DEBUG statements into the same window.

Add a new push button to the employee maintenance screen, with the caption *Output to Debug*

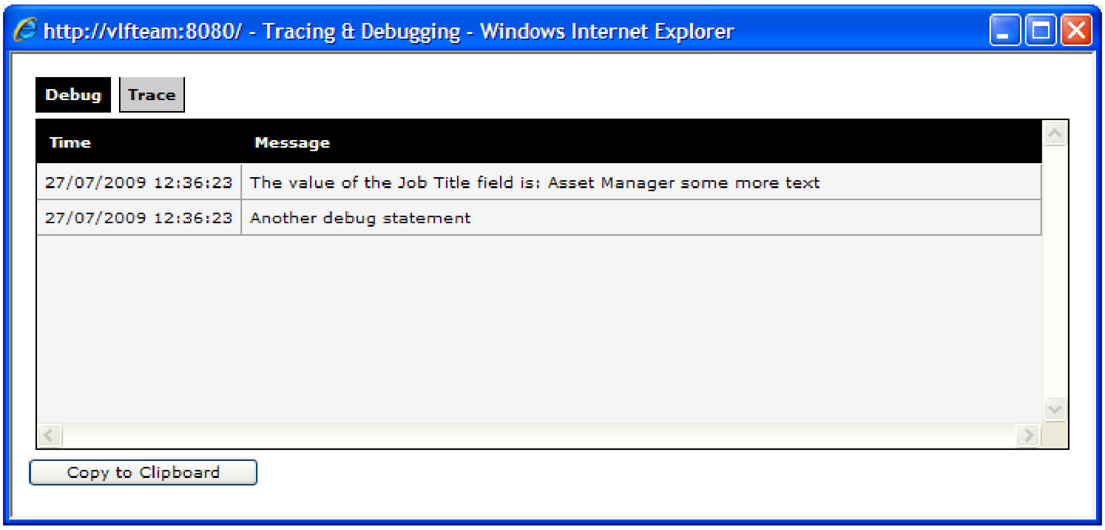
Edit the onClick property and add this code

```
var myField = FIELDS("Job_Title");
var value = myField.getValue();
DEBUG("The value of the Job Title field is:", value, "some more text" );
DEBUG("Another debug statement");
```

Save your changes

Note that the format for DEBUG is slightly different. The values to be concatenated are separated by commas, and you can have as many values as you want.

When you click on the button, a debug window is opened , showing all debug statements.


http://vlfteam:8080/ - Tracing & Debugging - Windows Internet Explorer
Debug
Trace
Copy to Clipboard

Time	Message
27/07/2009 12:36:23	The value of the Job Title field is: Asset Manager some more text
27/07/2009 12:36:23	Another debug statement

Tracing Basics

Sometimes, an alert interrupts the flow of processing too much. Tracing allows you to watch what happens with less impact.

Add a new push button to the employee maintenance screen, with the caption *Output to Trace*

Edit the onClick property and add this code

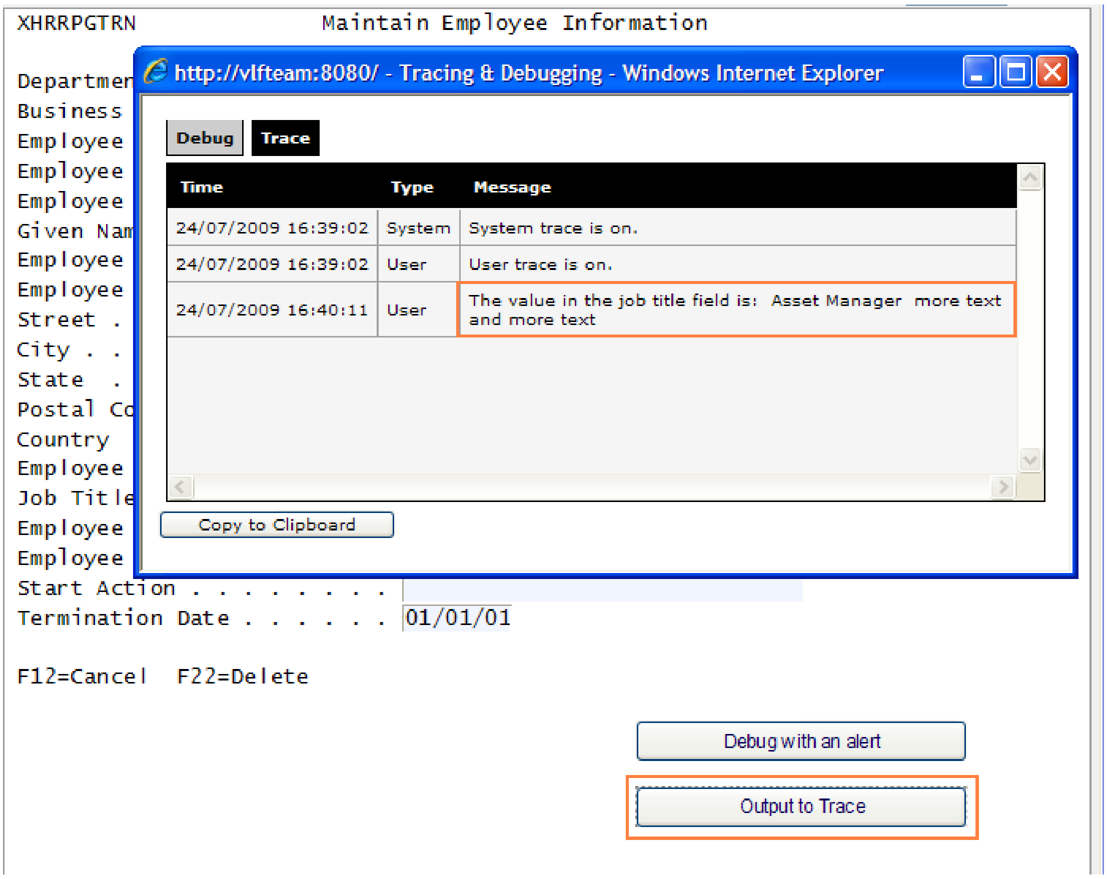
```
var F = FIELDS("Job_Title");
var value = F.getValue();
TRACE("The value in the job title field is: ", value, " more text ", " and more text"   );
```

Note the use of the TRACE command. You can specify as many parameters as you want. The TRACE will concatenate them and send them to the trace window.

To see the trace output, save your changes and exit aXes.

Now open the *Projects Home Page*. Select your project and then use the *Work as Developer (Tracing)* option.

Navigate back to the employee maintenance screen and press the *Output to Trace* button. The trace information appears in the trace window



```
XHRRPGTRN                Maintain Employee Information

Departmen
Business
Employee
Employee
Employee
Given Nam
Employee
Employee
Street .
City . .
State  .
Postal Co
Country
Employee
Job Title
Employee
Employee
Start Action . . . . . . . .
Termination Date . . . . . .  01/01/01

F12=Cancel   F22=Delete
```

http://vlfteam:8080/ - Tracing & Debugging - Windows Internet Explorer

Time	Type	Message
24/07/2009 16:39:02	System	System trace is on.
24/07/2009 16:39:02	User	User trace is on.
24/07/2009 16:40:11	User	The value in the job title field is: Asset Manager more text and more text

TRACE statements can be left in your code. They will only have an impact if aXes is running in trace mode.

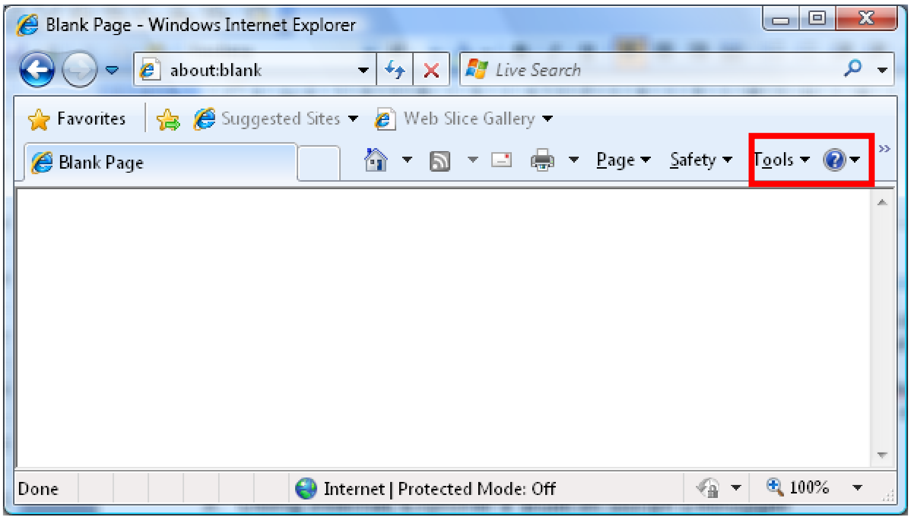
Debugging Tools and Options

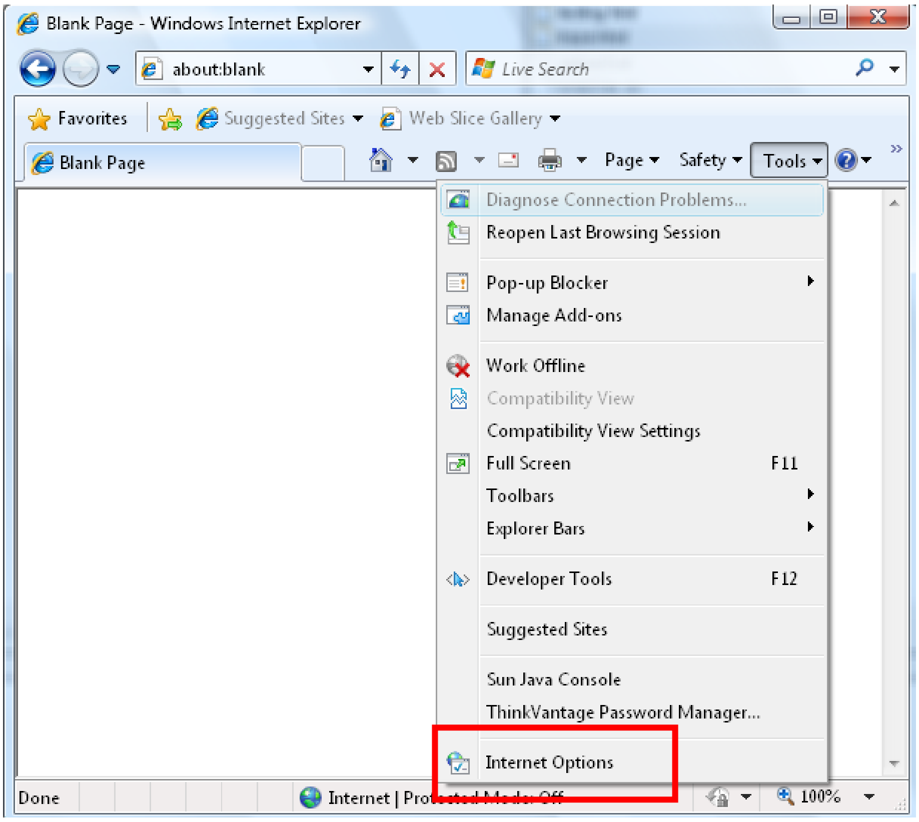
It is also possible to use the Microsoft Internet Explorer 8 Developer Tools to debug your scripts.

Internet Explorer 8 (IE8) comes with a built-in script debugger (part of IE8 developers tools). IE8 can be downloaded from http://www.microsoft.com/windows/internet-explorer/worldwide-sites.aspx

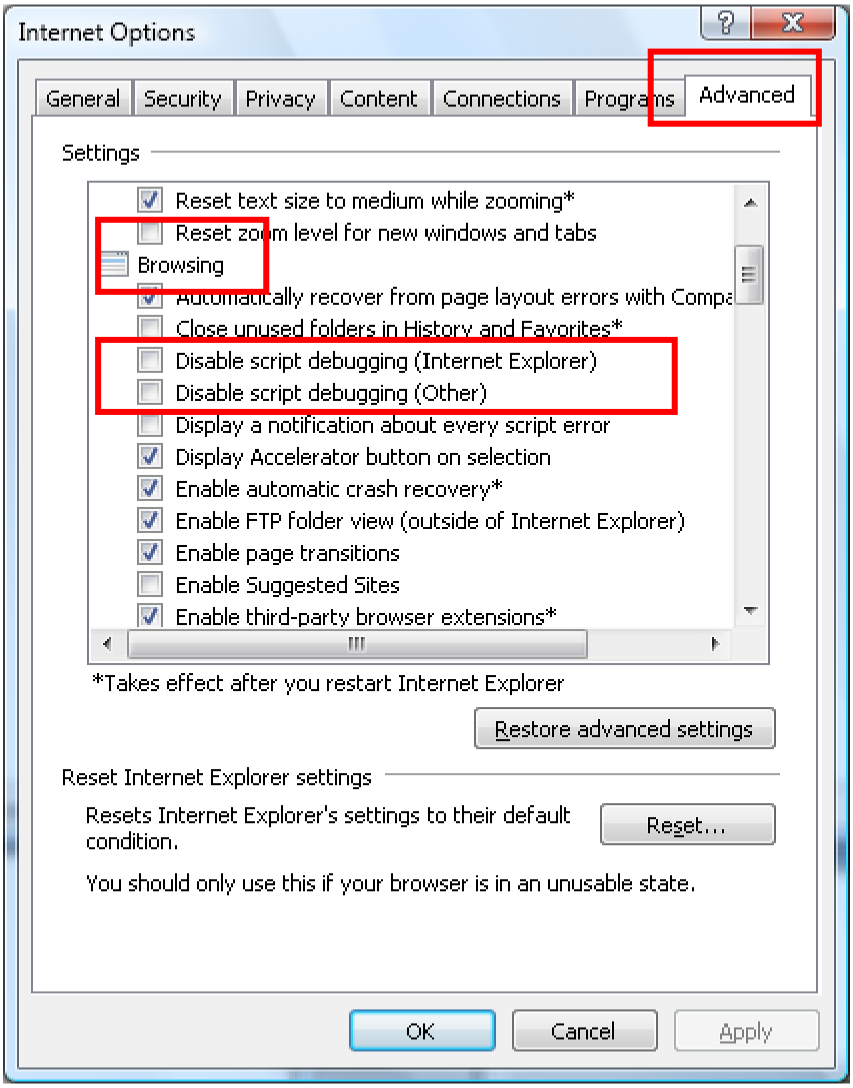
Several things that you need to check or configure before you start debugging your scripts

- Select the “Internet Options” option from the “Tools” menu.





- Go to the “Advanced” tab (last tab), then scroll down until you see the “Browsing” section. Make sure that “Disable script debugging (Internet Explorer)” and “Disable script debugging (Other)” checkboxes are **unchecked**.



Settings

- ☑ Reset text size to medium while zooming*
- □ Reset zoom level for new windows and tabs
- Browsing
- ☑ Automatically recover from page layout errors with Compa
- □ Close unused folders in History and Favorites*
- □ Disable script debugging (Internet Explorer)
- □ Disable script debugging (Other)
- □ Display a notification about every script error
- ☑ Display Accelerator button on selection
- ☑ Enable automatic crash recovery*
- ☑ Enable FTP folder view (outside of Internet Explorer)
- ☑ Enable page transitions
- □ Enable Suggested Sites
- ☑ Enable third-party browser extensions*

*Takes effect after you restart Internet Explorer

Reset Internet Explorer settings

Resets Internet Explorer's settings to their default condition.

You should only use this if your browser is in an unusable state.

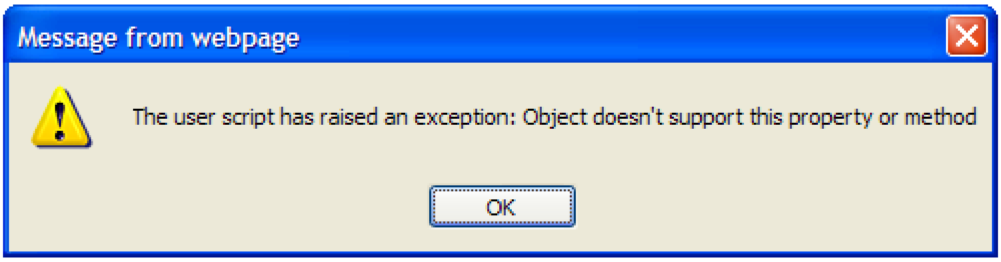
How to force IE8 to enter debug mode so you can step through your script line by line?

Suppose that you have the following fragment of code in your button event handler, which has a deliberate error in it (F.setvalue instead of F.setValue):

Modify the onClick code for the QPRINT Queue button, on the main screen , to contain this code:

```
var F = FIELDS("CommandLine");
F.setvalue("WRKOUTQ QPRINT");
SENDKEY("Enter");
```

You get an error when it runs:

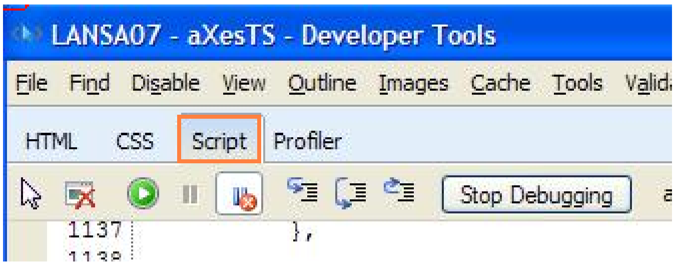


The user script has raised an exception: Object doesn't support this property or method

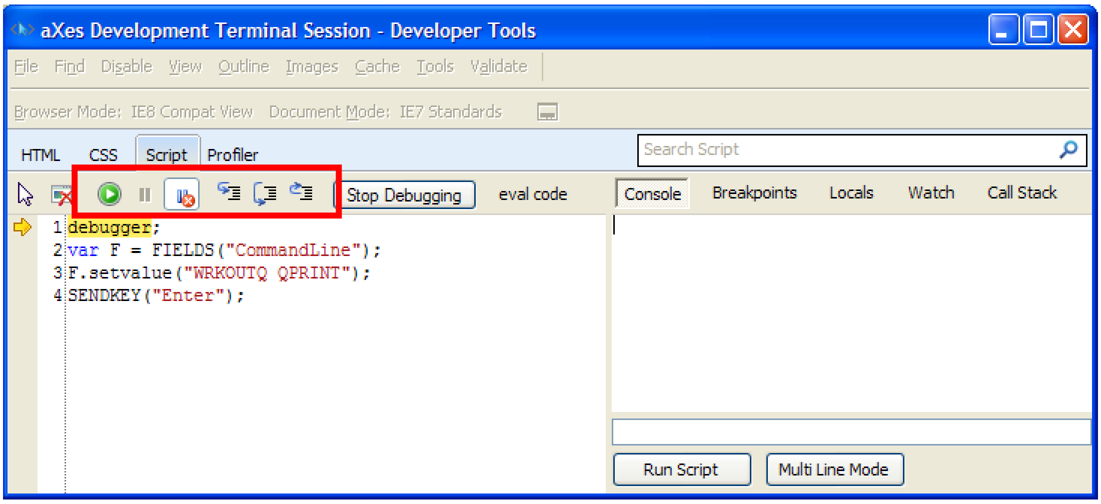
You will want to step through the script, line by line while at the same time watching the changes in the variable values. The easiest way to tell IE8 that you want to debug this piece of code is by putting statement “debugger” at the beginning of this code fragment.

```
debugger;
var F = FIELDS("CommandLine");
F.setvalue("WRKOUTQ QPRINT");
SENDKEY("Enter");
```

In the IE window, select Developer Tools from the Tools menu, or use F12 to start the built in IE debugger. In the Developer Tools window click on the Script tab:


LANSA07 - aXesTS - Developer Tools
File Find Disable View Outline Images Cache Tools Valid
HTML CSS Script Profiler
Stop Debugging
1137
},

When you click on the button that runs the onClick script containing the error, the debug window will come up:


aXes Development Terminal Session - Developer Tools
File Find Disable View Outline Images Cache Tools Validate
Browser Mode: IE8 Compat View Document Mode: IE7 Standards
HTML CSS Script Profiler
Search Script
Stop Debugging
eval code
Console Breakpoints Locals Watch Call Stack
Run Script
Multi Line Mode

```
debugger;
var F = FIELDS("CommandLine");
F.setvalue("WRKOUTQ QPRINT");
SENDKEY("Enter");
```

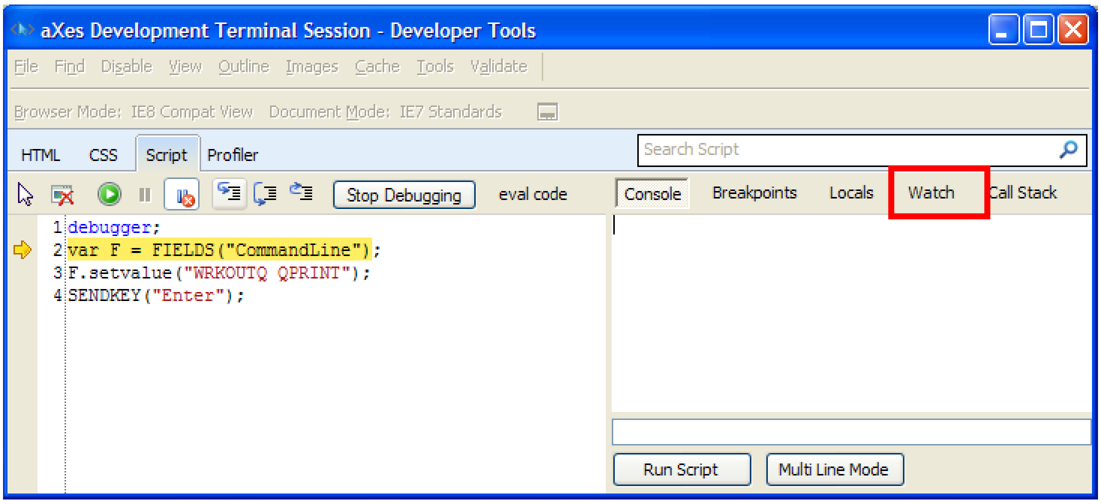
Have a closer look at toolbar buttons. Hover your mouse over those buttons to see the descriptions:

- Continue: exit debug mode and continue normal execution of the script.
- Break All & Break On Error: don't worry about these 2.
- Step Into: if the current statement is calling a function, it will step through the statements inside the function line by line. *You don't want to use this option because you will be stepped through all the internal aXes code and the jQuery code, which is generally not helpful.*
- **Step Over: it will not step through the statements inside the function**. *Use this option to step through the statements in the code you can currently see.*
- Step Out: execute the current function in one ago, then break into debug mode again.

Now click on "Step Over" button. (This steps over the internal code statements, to the next statement in the visible block of code)

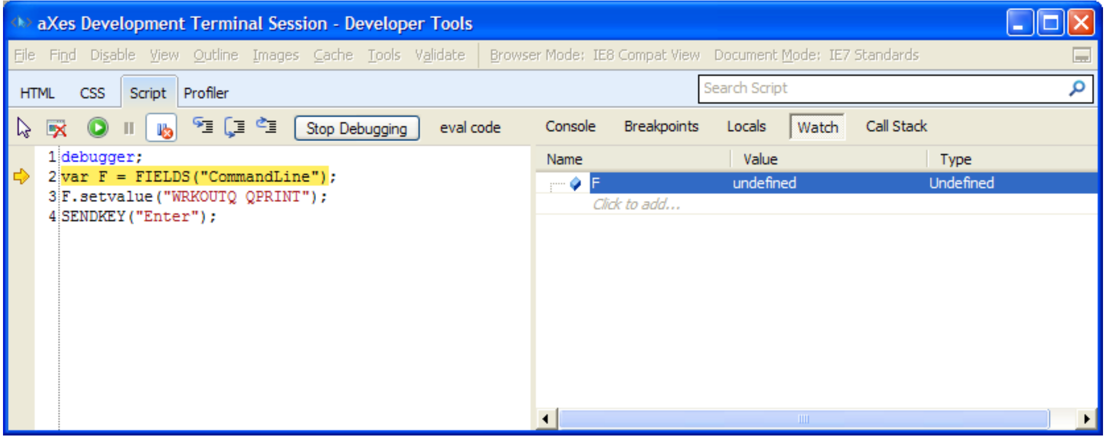
You can see that it's moving to the next statement. Note that the highlighted statement is the **next** statement to be executed – so it hasn't been executed yet.

Now click on the "Watch" tab – we are going to watch the content of variable F.


aXes Development Terminal Session - Developer Tools
File Find Disable View Outline Images Cache Tools Validate
Browser Mode: IE8 Compat View Document Mode: IE7 Standards
HTML CSS Script Profiler
Search Script
Stop Debugging
eval code
Console Breakpoints Locals Watch Call Stack
Run Script
Multi Line Mode

```
debugger;
var F = FIELDS("CommandLine");
F.setvalue("WRKOUTQ QPRINT");
SENDKEY("Enter");
```

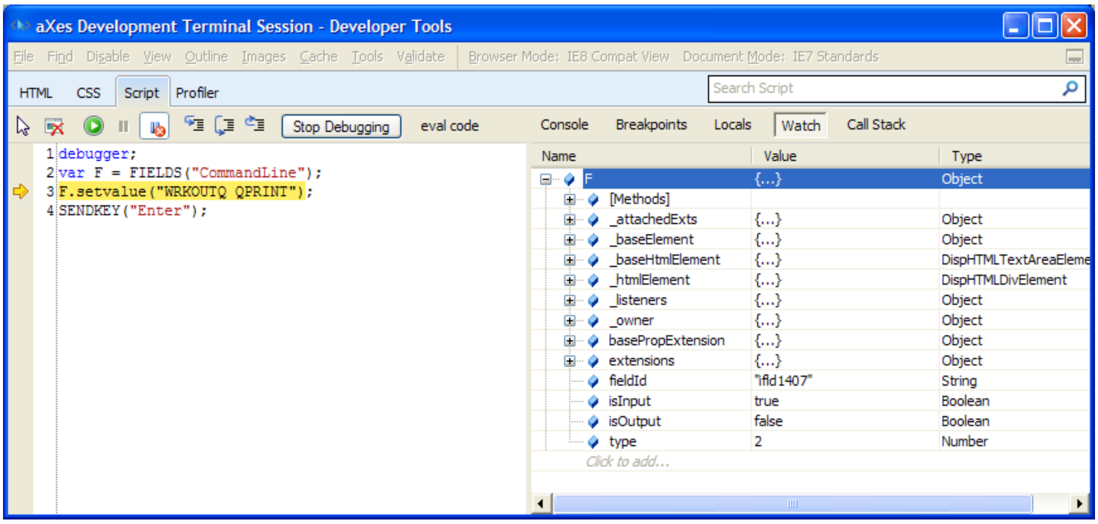
Click on the text “Click to add…” and type in F and press Enter. It will say undefined as the statement hasn’t been executed yet.
Click again the “Step Over” button.


aXes Development Terminal Session - Developer Tools
File Find Disable View Outline Images Cache Tools Validate Browser Mode: IE8 Compat View Document Mode: IE7 Standards
HTML CSS Script Profiler
Search Script
Stop Debugging
eval code
Console Breakpoints Locals Watch Call Stack

```
debugger;
var F = FIELDS("CommandLine");
F.setvalue("WRKOUTQ QPRINT");
SENDKEY("Enter");
```

Name	Value	Type
F	undefined	Undefined
Click to add...		

Now by expanding F, in the Watch tab, you can see all its properties.

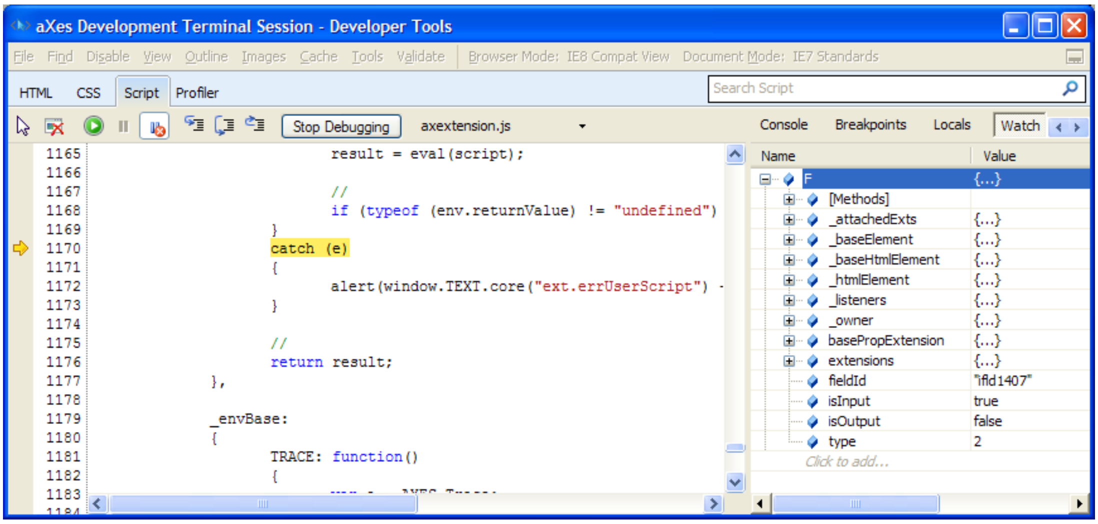

aXes Development Terminal Session - Developer Tools
File Find Disable View Outline Images Cache Tools Validate Browser Mode: IE8 Compat View Document Mode: IE7 Standards
HTML CSS Script Profiler
Search Script
Stop Debugging
eval code
Console Breakpoints Locals Watch Call Stack

```
debugger;
var F = FIELDS("CommandLine");
F.setvalue("WRKOUTQ QPRINT");
SENDKEY("Enter");
```

Name	Value	Type
F	{...}	Object
[Methods]		
_attachedExts	{...}	Object
_baseElement	{...}	Object
_baseHtmlElement	{...}	DispHTMLTextAreaEleme
_htmlElement	{...}	DispHTMLDivElement
_listeners	{...}	Object
_owner	{...}	Object
basePropExtension	{...}	Object
extensions	{...}	Object
fieldId	"ifld1407"	String
isInput	true	Boolean
isOutput	false	Boolean
type	2	Number
Click to add...		

But we have not encountered the error yet.

Try step over, one more time. This time we can see that the previous line was the source of the error. A few more step overs will take us to the actual alert.


aXes Development Terminal Session - Developer Tools
File Find Disable View Outline Images Cache Tools Validate Browser Mode: IE8 Compat View Document Mode: IE7 Standards
HTML CSS Script Profiler
Search Script
Stop Debugging
axextension.js
Console Breakpoints Locals Watch

```
                    result = eval(script);

                    //
                    if (typeof (env.returnValue) != "undefined")
                }
                catch (e)
                {
                    alert(window.TEXT.core("ext.errUserScript") -
                }

                //
                return result;
            },

            _envBase:
            {
                TRACE: function()
                {
```

Name	Value
F	{...}
[Methods]	
_attachedExts	{...}
_baseElement	{...}
_baseHtmlElement	{...}
_htmlElement	{...}
_listeners	{...}
_owner	{...}
basePropExtension	{...}
extensions	{...}
fieldId	"ifld1407"
isInput	true
isOutput	false
type	2
Click to add...	

So at this point, we would know to focus our attention on the line

```
F.setvalue("WRKOUTQ QPRINT");
```

Correct the line back to:

```
F.setValue("WRKOUTQ QPRINT");
```

and save your change.

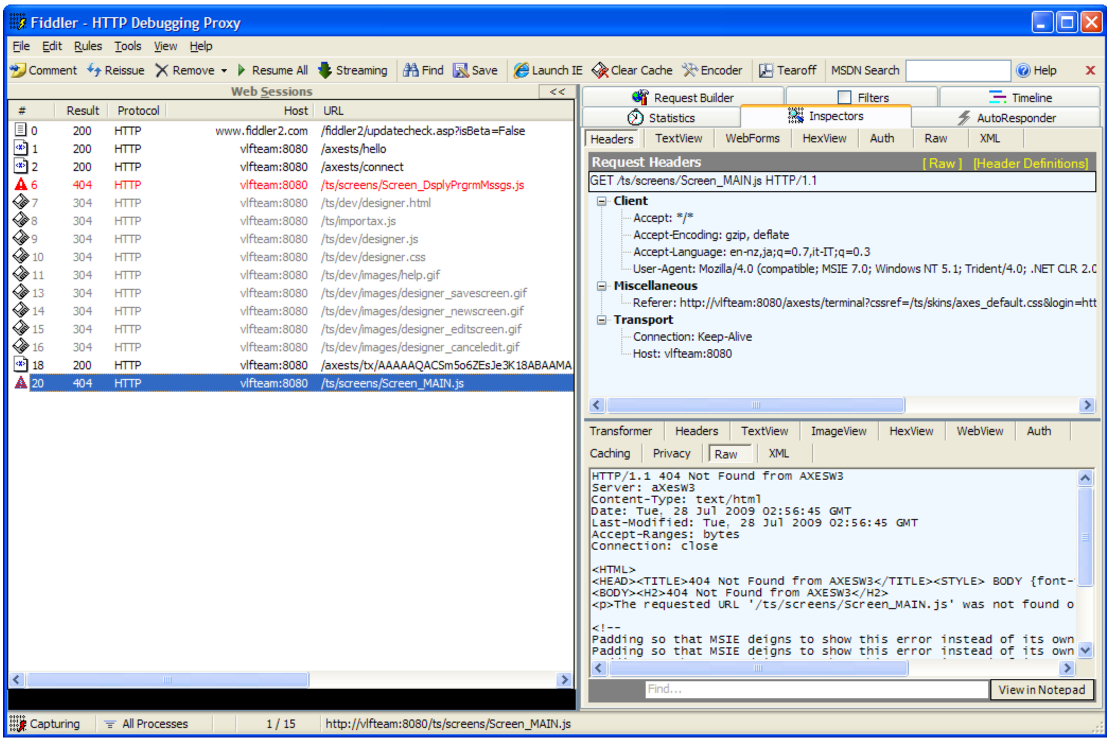
Using Fiddler

Fiddler is a free tool that allows you to watch the traffic sent between the server and the client, as your axes application runs.

You can find out about it and download it from here:
http://www.fiddler2.com/fiddler/help/

Install fiddler, and start it. Then start an axes session.

It produces a screen like this:


Fiddler - HTTP Debugging Proxy
File Edit Rules Tools View Help
Comment Reissue Remove Resume All Streaming Find Save Launch IE Clear Cache Encoder Tearoff MSDN Search Help
Web Sessions
Request Builder Filters Timeline
Statistics Inspectors AutoResponder
Headers TextView WebForms HexView Auth Raw XML
Transformer Headers TextView ImageView HexView WebView Auth
Caching Privacy Raw XML
Find...
View in Notepad
Capturing All Processes 1 / 15 http://vlfteam:8080/ts/screens/Screen_MAIN.js

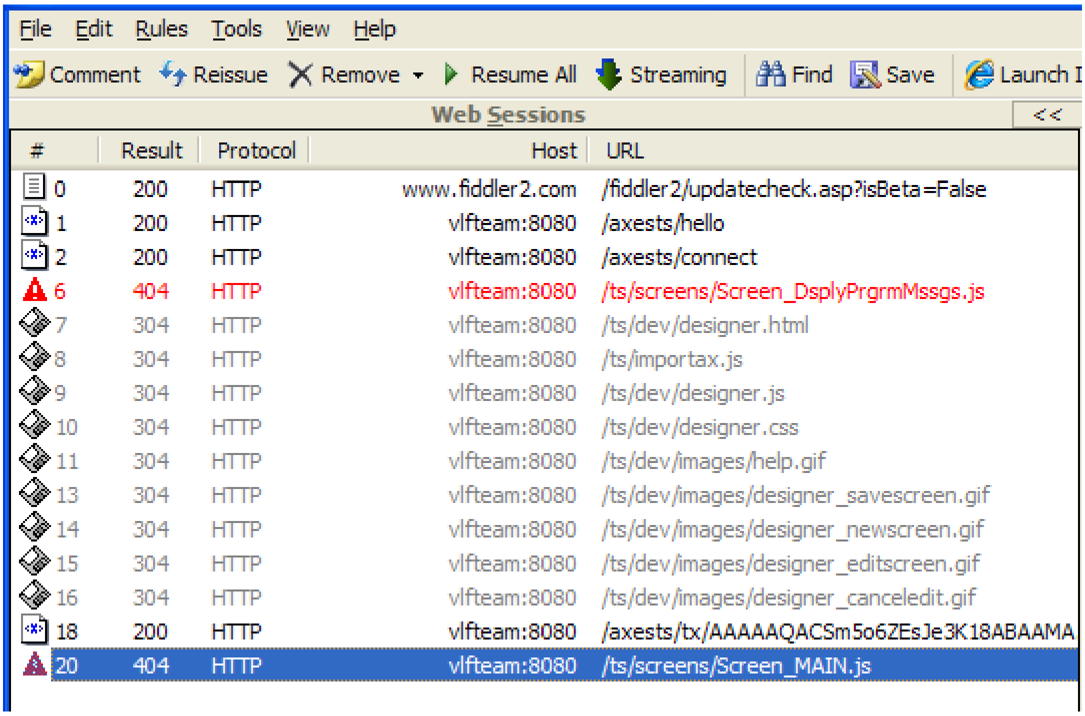
#	Result	Protocol	Host	URL
0	200	HTTP	www.fiddler2.com	/fiddler2/updatecheck.asp?isBeta=False
1	200	HTTP	vlfteam:8080	/axests/hello
2	200	HTTP	vlfteam:8080	/axests/connect
6	404	HTTP	vlfteam:8080	/ts/screens/Screen_DsplyPrgrmMssgs.js
7	304	HTTP	vlfteam:8080	/ts/dev/designer.html
8	304	HTTP	vlfteam:8080	/ts/importax.js
9	304	HTTP	vlfteam:8080	/ts/dev/designer.js
10	304	HTTP	vlfteam:8080	/ts/dev/designer.css
11	304	HTTP	vlfteam:8080	/ts/dev/images/help.gif
13	304	HTTP	vlfteam:8080	/ts/dev/images/designer_savescreen.gif
14	304	HTTP	vlfteam:8080	/ts/dev/images/designer_newscreen.gif
15	304	HTTP	vlfteam:8080	/ts/dev/images/designer_editscreen.gif
16	304	HTTP	vlfteam:8080	/ts/dev/images/designer_canceledit.gif
18	200	HTTP	vlfteam:8080	/axests/tx/AAAAAQACSm5o6ZEsJe3K18ABAAMA
20	404	HTTP	vlfteam:8080	/ts/screens/Screen_MAIN.js

Request Headers [Raw] [Header Definitions]

```
GET /ts/screens/Screen_MAIN.js HTTP/1.1
Client
    Accept: */*
    Accept-Encoding: gzip, deflate
    Accept-Language: en-nz,ja;q=0.7,it-IT;q=0.3
    User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1; Trident/4.0; .NET CLR 2.0
Miscellaneous
    Referer: http://vlfteam:8080/axests/terminal?cssref=/ts/skins/axes_default.css&login=htt
Transport
    Connection: Keep-Alive
    Host: vlfteam:8080
```

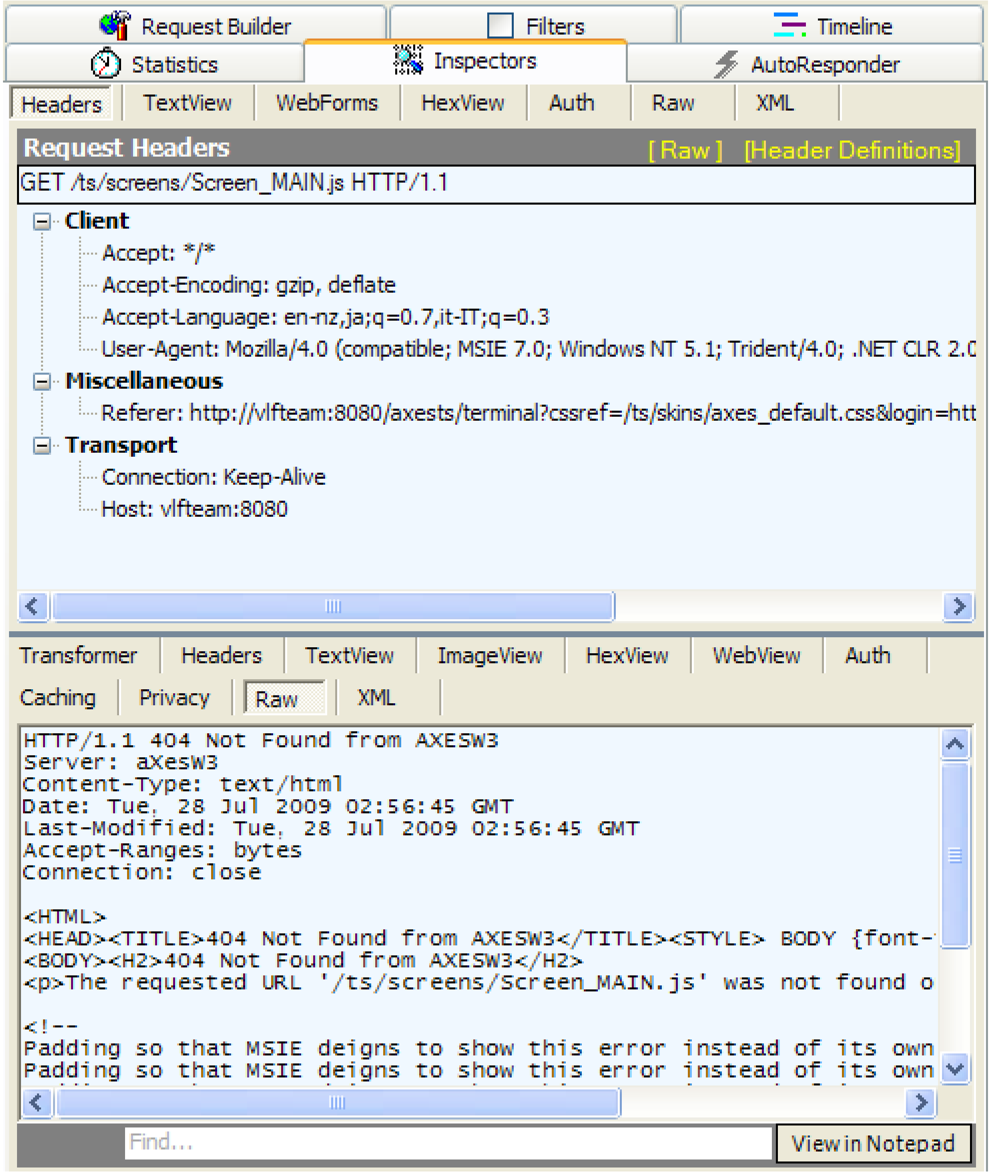
```
HTTP/1.1 404 Not Found from AXESW3
Server: aXesW3
Content-Type: text/html
Date: Tue, 28 Jul 2009 02:56:45 GMT
Last-Modified: Tue, 28 Jul 2009 02:56:45 GMT
Accept-Ranges: bytes
Connection: close

<HTML>
<HEAD><TITLE>404 Not Found from AXESW3</TITLE><STYLE> BODY {font-
<BODY><H2>404 Not Found from AXESW3</H2>
<p>The requested URL '/ts/screens/Screen_MAIN.js' was not found o

<!--
Padding so that MSIE deigns to show this error instead of its own
Padding so that MSIE deigns to show this error instead of its own
```

where each line in the left panel corresponds to a request and response from the web server


File Edit Rules Tools View Help
Comment Reissue Remove Resume All Streaming Find Save Launch I
Web Sessions

#	Result	Protocol	Host	URL
0	200	HTTP	www.fiddler2.com	/fiddler2/updatecheck.asp?isBeta=False
1	200	HTTP	vlfteam:8080	/axests/hello
2	200	HTTP	vlfteam:8080	/axests/connect
6	404	HTTP	vlfteam:8080	/ts/screens/Screen_DsplyPrgrmMssgs.js
7	304	HTTP	vlfteam:8080	/ts/dev/designer.html
8	304	HTTP	vlfteam:8080	/ts/importax.js
9	304	HTTP	vlfteam:8080	/ts/dev/designer.js
10	304	HTTP	vlfteam:8080	/ts/dev/designer.css
11	304	HTTP	vlfteam:8080	/ts/dev/images/help.gif
13	304	HTTP	vlfteam:8080	/ts/dev/images/designer_savescreen.gif
14	304	HTTP	vlfteam:8080	/ts/dev/images/designer_newscreen.gif
15	304	HTTP	vlfteam:8080	/ts/dev/images/designer_editscreen.gif
16	304	HTTP	vlfteam:8080	/ts/dev/images/designer_canceledit.gif
18	200	HTTP	vlfteam:8080	/axests/tx/AAAAAQACSm5o6ZEsJe3K18ABAAMA
20	404	HTTP	vlfteam:8080	/ts/screens/Screen_MAIN.js

On the panel on the right, in the inspectors tab, is the information about the request sent (top panel) and the response received (bottom panel), for the selected line:



GET /ts/screens/Screen_MAIN.js HTTP/1.1

Client
- Accept: */*
- Accept-Encoding: gzip, deflate
- Accept-Language: en-nz,ja;q=0.7,it-IT;q=0.3
- User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1; Trident/4.0; .NET CLR 2.0

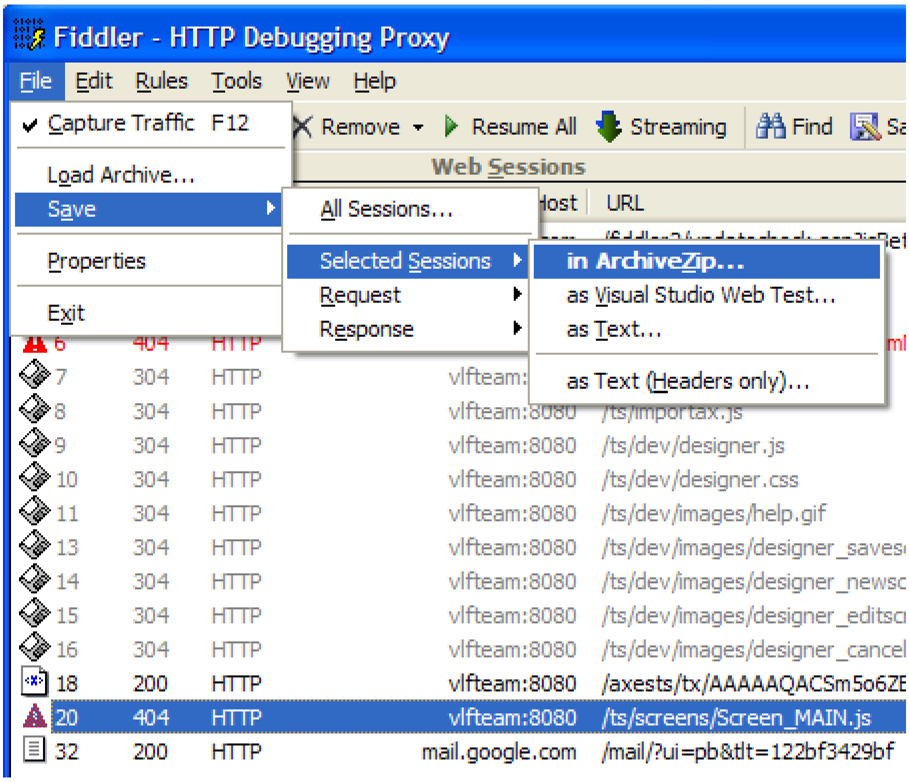
Miscellaneous
- Referer: http://vlfteam:8080/axests/terminal?cssref=/ts/skins/axes_default.css&login=htt

Transport
- Connection: Keep-Alive
- Host: vlfteam:8080

```
HTTP/1.1 404 Not Found from AXESW3
Server: aXesW3
Content-Type: text/html
Date: Tue, 28 Jul 2009 02:56:45 GMT
Last-Modified: Tue, 28 Jul 2009 02:56:45 GMT
Accept-Ranges: bytes
Connection: close

<HTML>
<HEAD><TITLE>404 Not Found from AXESW3</TITLE><STYLE> BODY {font-
<BODY><H2>404 Not Found from AXESW3</H2>
<p>The requested URL '/ts/screens/Screen_MAIN.js' was not found o

<!--
Padding so that MSIE deigns to show this error instead of its own
Padding so that MSIE deigns to show this error instead of its own
```

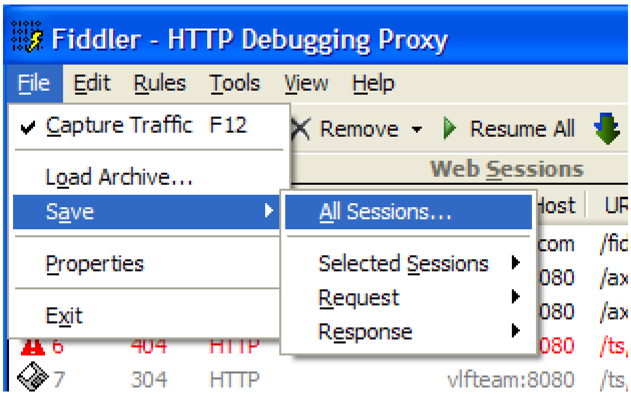
There are many views and features in this product, but one particularly useful one is the ability to save either one or all sessions to file, which can be sent when trying to resolve problems.

To save one session, select it and save as follows:



			Host	URL
6	404	HTTP		
7	304	HTTP	vlfteam:	
8	304	HTTP	vlfteam:8080	/ts/importax.js
9	304	HTTP	vlfteam:8080	/ts/dev/designer.js
10	304	HTTP	vlfteam:8080	/ts/dev/designer.css
11	304	HTTP	vlfteam:8080	/ts/dev/images/help.gif
13	304	HTTP	vlfteam:8080	/ts/dev/images/designer_saves
14	304	HTTP	vlfteam:8080	/ts/dev/images/designer_newsc
15	304	HTTP	vlfteam:8080	/ts/dev/images/designer_editsc
16	304	HTTP	vlfteam:8080	/ts/dev/images/designer_cancel
18	200	HTTP	vlfteam:8080	/axests/tx/AAAAAQACSm5o6ZE
20	404	HTTP	vlfteam:8080	/ts/screens/Screen_MAIN.js
32	200	HTTP	mail.google.com	/mail/?ui=pb&tlt=122bf3429bf

To save all sessions:



eXtensions Tutorial 5 - The USERENV object

What is the USRENV object?

The aXes ENV (Environment) object contains common properties and functions that you use in your eXtension scripting.

The USERENV (User Environment) object is a JavaScript object. It extends the concept of the ENV object by acting as a container for properties and functions that are uniquely part of *your* project.

What is USRENV used for?

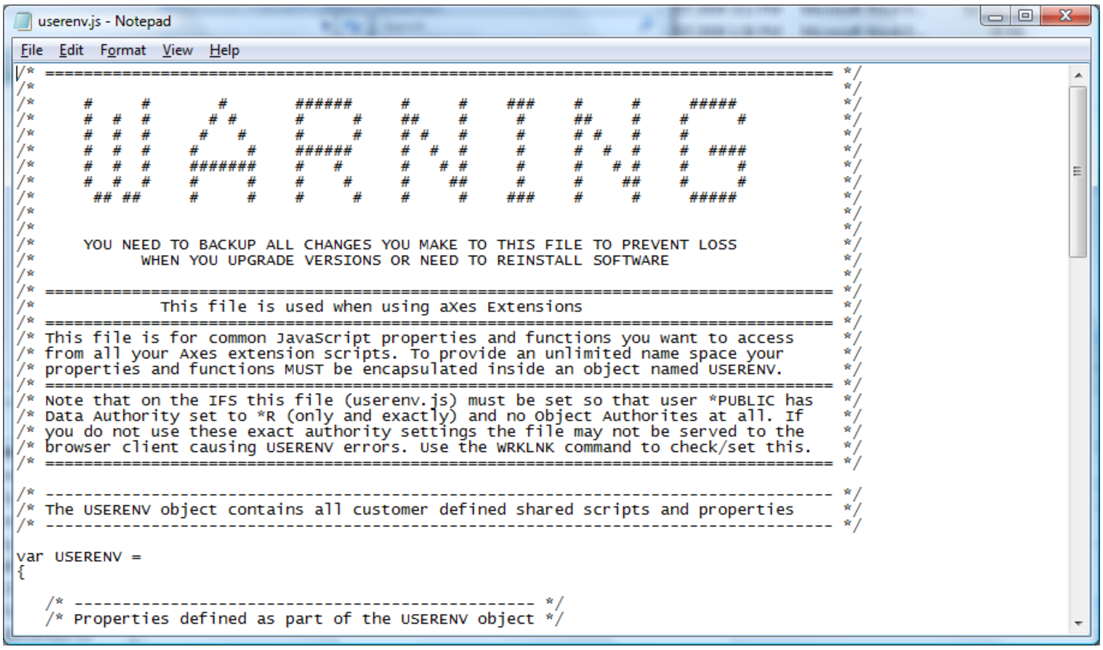
USERENV's main uses are:

- Allows definition and change of common values in just one place.
- Define common logic to encourage reuse and minimize coding.
- As a session persistent container of eXtension state values.
- As a way of exchanging information between eXtension scripts.
- As a way of signalling custom events between eXtension scripts.

How and where is it defined?

The USERENV object is defined in the userenv.js file.
The userenv.js file resides in your definition set / project folder.

Open the *Projects Home Page*. Select your project and then use the *Edit USERENV Object* option. The resulting NOTEPAD display should look something like this:



```
/* ================================================================================== */
/*                                                                                    */
/*     #     #       #       ######     #     #    ###    #     #     #####           */
/*     #  #  #      # #      #     #    ##    #     #     ##    #    #     #          */
/*     #  #  #     #   #     #     #    # #   #     #     # #   #    #                */
/*     #  #  #    #     #    ######     #  #  #     #     #  #  #    #  ####          */
/*     #  #  #    #######    #   #      #   # #     #     #   # #    #     #          */
/*     #  #  #    #     #    #    #     #    ##     #     #    ##    #     #          */
/*      ## ##     #     #    #     #    #     #    ###    #     #     #####           */
/*                                                                                    */
/*                                                                                    */
/*     YOU NEED TO BACKUP ALL CHANGES YOU MAKE TO THIS FILE TO PREVENT LOSS           */
/*           WHEN YOU UPGRADE VERSIONS OR NEED TO REINSTALL SOFTWARE                  */
/*                                                                                    */
/* ================================================================================== */
/*             This file is used when using aXes Extensions                           */
/* ================================================================================== */
/* This file is for common JavaScript properties and functions you want to access     */
/* from all your Axes extension scripts. To provide an unlimited name space your      */
/* properties and functions MUST be encapsulated inside an object named USERENV.      */
/* ================================================================================== */
/* Note that on the IFS this file (userenv.js) must be set so that user *PUBLIC has   */
/* Data Authority set to *R (only and exactly) and no Object Authorites at all. If    */
/* you do not use these exact authority settings the file may not be served to the    */
/* browser client causing USERENV errors. Use the WRKLNK command to check/set this.   */
/* ================================================================================== */

/* ---------------------------------------------------------------------------------- */
/* The USERENV object contains all customer defined shared scripts and properties     */
/* ---------------------------------------------------------------------------------- */

var USERENV =
{

   /* ------------------------------------------------ */
   /* Properties defined as part of the USERENV object */
```

The warning is very important.

When customizing this file, make sure you keep versions to avoid loss of your work when upgrading aXes or if you need to re-install.

Looking at the userenv.js file content you will see this non-comment JavaScript:

```
var USERENV =
{

   /* ------------------------------------------------ */
   /* Properties defined as part of the USERENV object */
   /* ------------------------------------------------ */

   staticTablesFile    : "tables_static.txt",  /* Default name for file …
   dynamicTablesFile   : "tables_dynamic.txt", /* Default name for file …
```

This code declares a JavaScript object called USERENV.
It contains properties named staticTablesFile and dynamicTablesFile.

This means that you can refer to USERENV.staticTablesFile and USERENV.dynamicTablesFile anywhere in JavaScript code that you add to an eXtension.

Adding a property to the USERENV object

Try adding two properties of your own to USERENV.

Open the *Projects Home Page*. Select your project and then use the *Edit USERENV Object* option to open file userenv.js in your project folder in NOTEPAD.

Add the two highlighted property declarations to

```
var USERENV =
{
   /* ------------------------------------------------ */
   /* Properties defined as part of the USERENV object */
   /* ------------------------------------------------ */

   companyName : "My Test Company",
   messageLine : 22,


   staticTablesFile    :  …… etc, etc
```

Now save the updated userenv.js file.

In one of the extensions you created in a preceding tutorial, add code that does this:

```
    alert(USERENV.companyName);
    alert(USERENV.messageLine.toString());
    USERENV.messageLine = USERENV.messageLine + 27;
    alert(USERENV.messageLine.toString());
```

By causing the code to execute in your eXtension you should be able to see that these new properties that you have created can be read from and written to the USERENV object.

> *Hint*: If you change userenv.js, you need to stop and restart your aXes 5250 session to pick up the new version. Userenv.js is read just once as the 5250 session is initialized.

Adding a function to the USERENV object

Now, add a new function to the USERENV object.

Open the *Projects Home Page*. Select your project and then use the *Edit USERENV Object* option to open file userenv.js in your project folder.

In the body of the USERENV object declare function addNumbers like this:

```
   addNumbers : function(a,b,c)
   {
      var result = a + b + c;
      return(result);
   }, /* ← Note the comma */
```

Now save the updated userenv.js file. In one of the extensions you created in a preceding tutorial, add scripting code that does this:

```
var x = USERENV.addNumbers(4,5,6);
alert("Result = " + x.toString());
```

By causing the code to execute in your eXtension you should be able to see that this new function can be used anywhere by eXtension logic.

Extending and Organizing the USERENV namespace

The style of JavaScript coding used to define the USERENV object is primarily used to reserve the USERENV namespace.

This means you can confidently create a new USERENV function called myNewFunction() knowing that there will be no such function already in the aXes namespace, nor will there ever be one in the future - because it will be called USERENV.myNewFunction().

You can extend this approach to subdivide the USERENV name space itself.

For example, if this code is added right at the end of the userenv.js file (after the closure of the USERENV object definition):

```
USERENV.Print =
{
   testProperty1 : "Hello",
   testProperty2 :  300,

   test : function(a)
   {
   } /* ← Note no comma as this is the last member of USERENV.Print */
};
```

Now the object and namespace USERENV.Print has been created, presumably to contain things related to printing.

You can now reference properties USERENV.Print.testProperty1 or USERENV.Print.testProperty2 and function USERENV.Print.test() in any eXtension scripting.

> *Hint*: You can actually imbed the USERENV.Print object declaration inside the USERENV base definition, but sometimes the syntax gets a bit cryptic and hard to follow.

Using a common USERENV and SHARED object (RAMP-TS)

If you are using the RAMP-TS product from LANSA, you may know that its scripting support contains a SHARED object namespace. Its intended purpose is virtually identical to the USERENV namespace.

In RAMP-TS applications you can share a single common object by executing this code:

```
var SHARED = { };
```

in the RAMP-TS UF_SY420_RTS.js file.
This declares the SHARED object as an empty object.

Then in your RAMP-TS logon script, execute the JavaScript code:

```
       SHARED = USERENV;
```

making the RAMP-TS SHARED object and the eXtensions USERENV object one and the same thing.

aXes Documentation Library

eXtensions Tutorial 6 - Tables and XML Documents

Static Tables that load from static data

Static tables are JavaScript objects defined in the file called **tables_static.txt**.

To edit the tables, open the *Projects Home Page*. Select your project and then use the *Edit Static Tables* option.

Each individual table is defined within a scripting block like this:

```
DefineObjectInstance {

};
```

The properties defined in each of the objects are:

className	must be "StaticTable" for tables defined in this file
name	A name that uniquely identifies a table within an aXes session. **This is the name to specify in the tableName property of the eXtension.**
source	"inline" means the table content is defined as an array of rows in the object. "sql" means the table content is a result of executing a Select SQL command.

validateSession	For table operations that use SQL indicates that the execution of the SQL operation may only be initiated by a signed on 5250 session. The default is false, the other allowable value is true. For example:
	DefineObjectInstance {
	className = "DynamicTable",
	validateSession = true,
	name = "TEST_SQL_1",

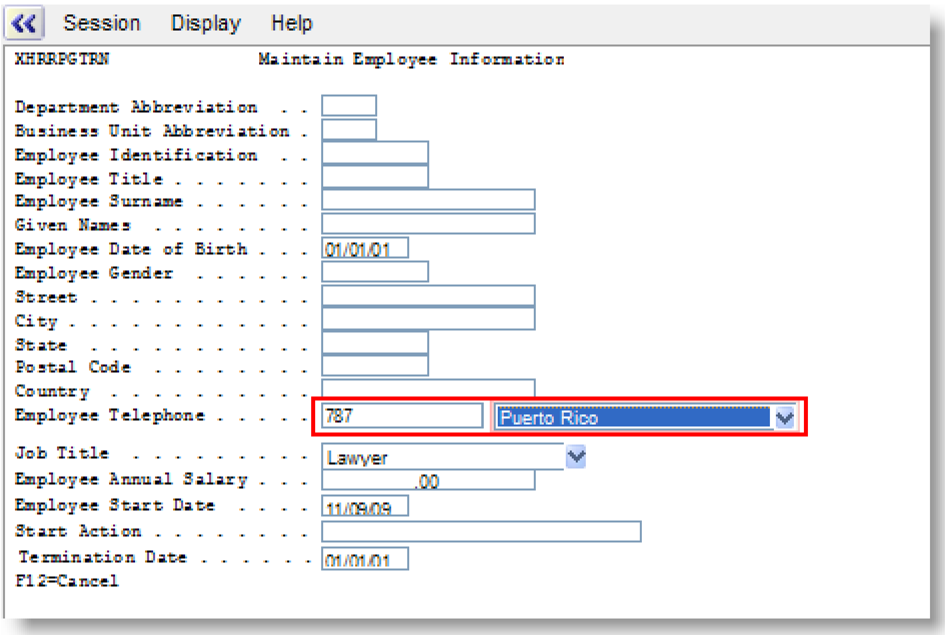
selectSQLcommand	When the source is "sql" this is the command to execute
resultColumnNames	When the source is "sql", the names given to the columns in the table. **These are the names to refer to in the eXtension – e.g. ROW.<colname>**
resultColumnCaptions	When being output as CSV data, the captions to be used for each column in the table.
rows	When the source is "inline", the array of columns that defines the table and its content. Here, the column names are implicitly the left operands and will be the names to refer to in the eXtension. For example if you look at the ISOCountry table and want to refer to the value column you would say ROW.value

In this exercise you will modify the Maintain Employee screen for adding a new employee. The screen was identified in Tutorial 1 as **XHRRPGTRN_Maint.**

You will add a drop-down next to the Employee Telephone field which will display US regions corresponding to telephone area codes. The values in the drop-down are sourced from a static table you will create.

(Whether the screen on your system looks like the following screen shot depends on which other tutorials you have completed.)

When the end-user selects a region, the area code is added to the Employee Telephone field.



```
XHRRPGTRN              Maintain Employee Information

Department Abbreviation  . .
Business Unit Abbreviation .
Employee Identification  . .
Employee Title . . . . . . .
Employee Surname . . . . . .
Given Names  . . . . . . . .
Employee Date of Birth . . .  01/01/01
Employee Gender  . . . . . .
Street . . . . . . . . . . .
City . . . . . . . . . . . .
State  . . . . . . . . . . .
Postal Code  . . . . . . . .
Country  . . . . . . . . . .
Employee Telephone . . . . .  787          Puerto Rico
Job Title  . . . . . . . . .  Lawyer
Employee Annual Salary . . .           .00
Employee Start Date  . . . .  11/09/09
Start Action . . . . . . . .
 Termination Date . . . . . .  01/01/01
F12=Cancel
```

Step 1.

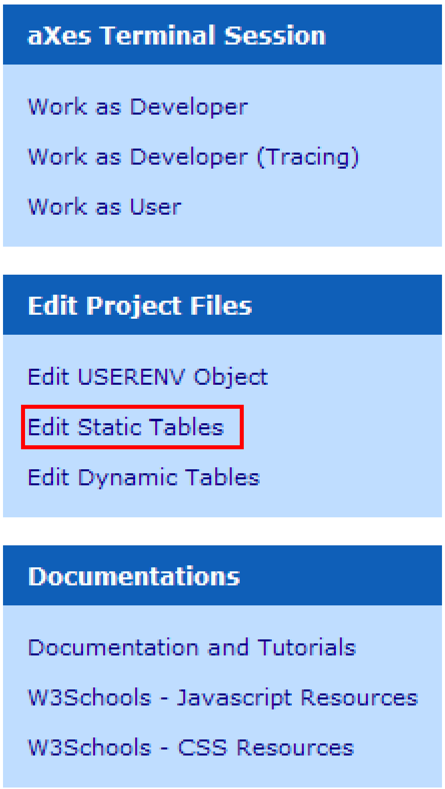
Go to the screen identified in Tutorial 1 as **XHRRPGTRN_Maint**
In **XHRRPGTRN_Maint** identify this field:

Employee Telephone as **Employee_Telephone**

Don't forget to use the Save button, otherwise the Employee_Telephone identification will not be saved.

Step 2.

In the Projects Home Page choose the option Edit Static Tables:



aXes Terminal Session

Work as Developer

Work as Developer (Tracing)

Work as User

Edit Project Files

Edit USERENV Object

Edit Static Tables

Edit Dynamic Tables

Documentations

Documentation and Tutorials

W3Schools - Javascript Resources

W3Schools - CSS Resources

Insert a new line and copy/paste this code and then save the file:

```
-- ============================
-- Some US Telephone area codes
-- ============================

    DefineObjectInstance {

      className = "StaticTable",
      name    = "USAreaCode",
      source  = "inline",
      rows = {
                {areacode="202",location="Washington DC"},
                {areacode="208",location="All parts of Idaho"},
                {areacode="209",location="Fresno and Stockton"},
                {areacode="212",location="New York City (Manhattan only)"},
                {areacode="213",location="Los Angeles, California"},
                {areacode="217",location="Springfield, Champaign-Urbana"},
                {areacode="218",location="Duluth, (Northern) Minnesota"},
                {areacode="228",location="Southern Mississippi"},
                {areacode="307",location="All parts of Wyoming"},
                {areacode="615",location="Chattanooga and Nashville"},
                {areacode="630",location="Chicago Metro area"},
                {areacode="650",location="San Francisco area, CA"},
                {areacode="715",location="Eau Claire and Wausau"},
                {areacode="740",location="South Eastern Ohio"},
                {areacode="787",location="Puerto Rico"},
                {areacode="903",location="Texarkana and Paris"},
                {areacode="904",location="Jacksonville and Pensacola"},
                {areacode="909",location="Riverside and San Bernardino"},
                {areacode="918",location="Muskogee and Tulsa"}

             },

      };
```

Step 3.

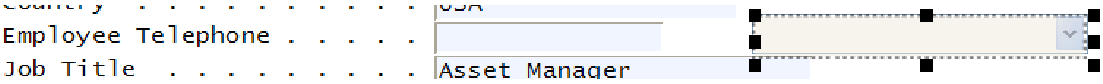
Start an aXes development session and navigate to the **XHRRPGTRN_Maint** screen.

Step 4.

Start customising the screen.

Step 5.

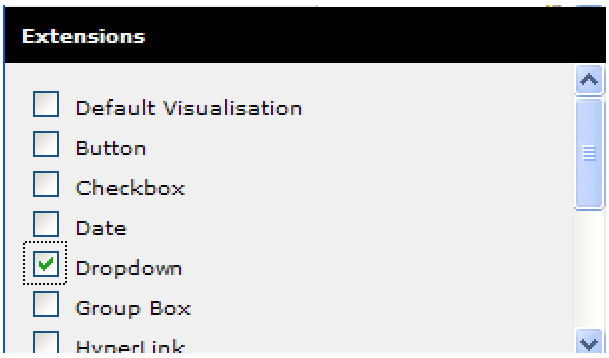
Add a new element and drag it next to the Employee Telephone input field.



Country . . . . USA
Employee Telephone . . . . .
Job Title . . . . Asset Manager

Step 6.

Check the Drop Down eXtension.



Extensions

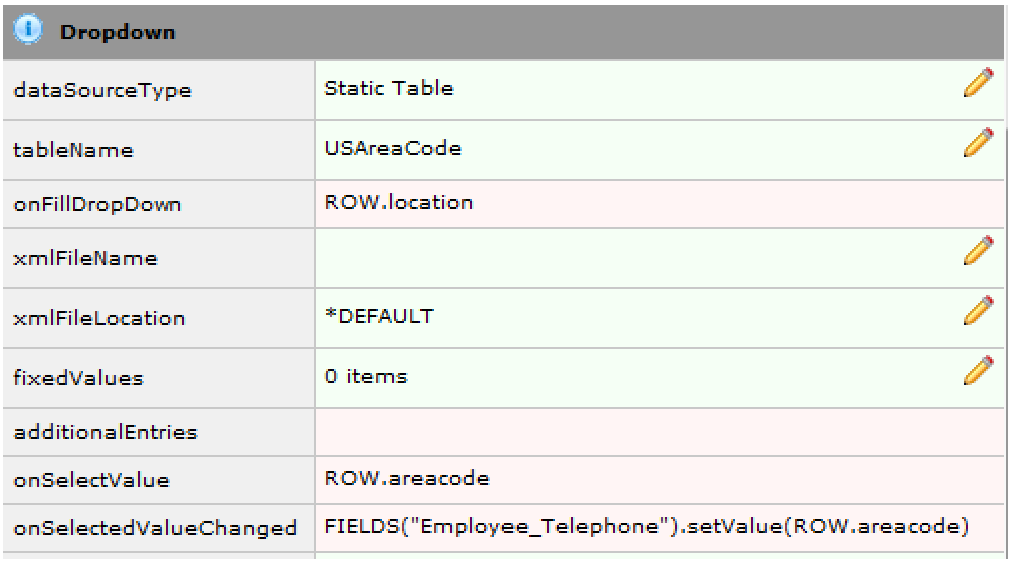
☐ Default Visualisation
☐ Button
☐ Checkbox
☐ Date
☑ Dropdown
☐ Group Box
☐ HyperLink

Step 7.

Set these Drop Down properties:

```
dataSourceType: Static Table
tableName: USAreaCode
onFillDropDown: ROW.location
```

```
onSelectValue: ROW.areacode
onSelectedValueChanged: FIELDS("Employee_Telephone").setValue(ROW.areacode)
```



Dropdown	
dataSourceType	Static Table
tableName	USAreaCode
onFillDropDown	ROW.location
xmlFileName	
xmlFileLocation	*DEFAULT
fixedValues	0 items
additionalEntries	
onSelectValue	ROW.areacode
onSelectedValueChanged	FIELDS("Employee_Telephone").setValue(ROW.areacode)

Step 8.

Save the extension, then exit from the screen to the XHRRPGTRN_Select screen and click the Add button to add a new employee.

In the XHRRPGTRN_Maintain screen use the Region drop-down to set the area code in the Employee Telephone field.

Step 9.

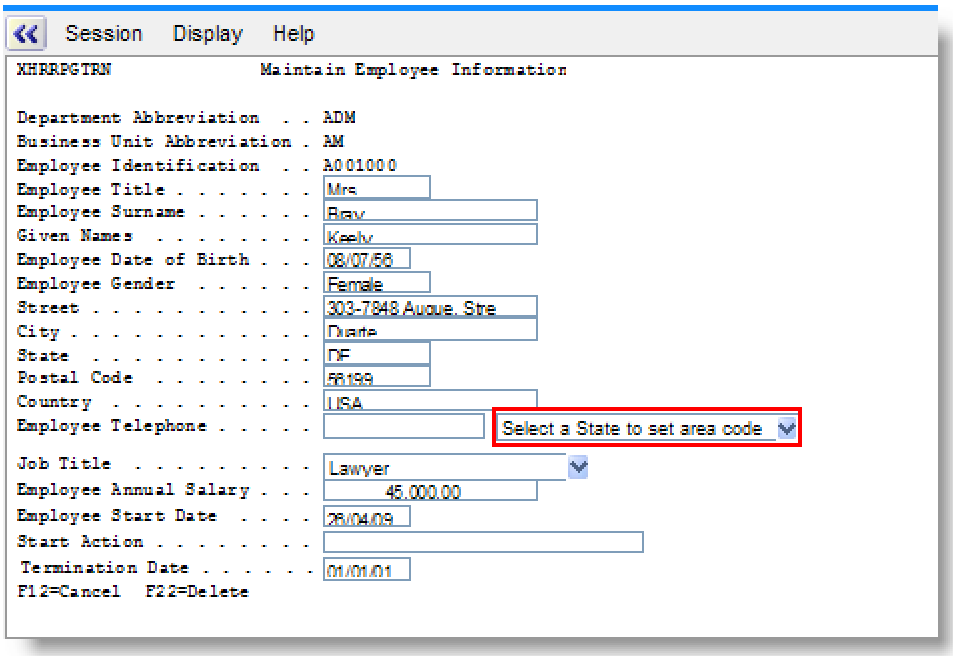
Notice that when you display the details for a new employee, the Region drop-down always shows the first entry in the USAreaCode table, Washington DC.

You can use the Additional Entries property to add an entry to appear at the top of the drop down that says something like "Select a State to set area code". Edit the property and add this code:

```
var oROW = { areacode: "", location: "Select a State to set area code" };

this.addTableROW (oROW, true);
```

Restart aXes.

Now when employee details are displayed, the drop-down will show the additional entry:



```
XHRRPGTRN              Maintain Employee Information

Department Abbreviation  . .  ADM
Business Unit Abbreviation .  AM
Employee Identification  . .  A001000
Employee Title . . . . . . .  Mrs
Employee Surname . . . . . .  Bray
Given Names  . . . . . . . .  Keely
Employee Date of Birth . . .  08/07/56
Employee Gender  . . . . . .  Female
Street . . . . . . . . . . .  303-7848 Augue, Stre
City . . . . . . . . . . . .  Duarte
State  . . . . . . . . . . .  DE
Postal Code  . . . . . . . .  56199
Country  . . . . . . . . . .  USA
Employee Telephone . . . . .             Select a State to set area code
Job Title  . . . . . . . . .  Lawyer
Employee Annual Salary . . .      45,000.00
Employee Start Date  . . . .  26/04/09
Start Action . . . . . . . .
Termination Date . . . . . .  01/01/01
F12=Cancel  F22=Delete
```

Static Tables that load from a database file

Static tables can load from a database file. The data is loaded the first time the table is used in an aXes session, and remembered thereafter.

In this example we will add a dropdown that is populated from a table that already exists, called XHRDepartment.

To view the table:

Step 1.

In the Projects Home Page choose the option Edit Static Tables.

The definition of the XHRDepartment table should look like this:

```
-- ===============================================================================
-- Departments Table - Select from the shipped demonstration table AXESDEMO.XHRDEPT
-- ===============================================================================

DefineObjectInstance {

     className          = "StaticTable",
     name               = "XHRDepartment",
     source             = "sql",
     selectSQLcommand   = "XHRDEPCDE,XHRDEPNME from AXESDEMO.XHRDEPT",
     resultColumnNames  = {"value","text"},
  };
```

Note how the source for the table is sql. This means that the data for the table is read from the iSeries database file, using the command in selectSQLcommand

that is: read fields XHRDEPCDE and XHRDEPNME from database file XHRDEPT in library AXESDEMO.

The data read is mapped into the table columns named in resultColumnNames.

So the data in field XHRDEPCDE is mapped into the table column named "value", and the data in field XHRDEPNME is mapped into the table column called "text"

To associate this table with a drop down:

Step 1.

Go to the screen identified in Tutorial 1 as **XHRRPGTRN_Select**
Press F6 to Add an Employee.

The Add screen should be recognised as the **XHRRPGTRN_Maint** screen.

Step 2.

Click on the department field and change the visualization to drop down.

Change the properties of the drop down as follows

```
dataSourceType: Static Table
tableName      : XHRDepartment
onFillDropDown : ROW.text
onSelectValue  : ROW.value
```

Step 3.

Also add an "unselected" entry to the drop down, by adding the following script to the additionalEntries property

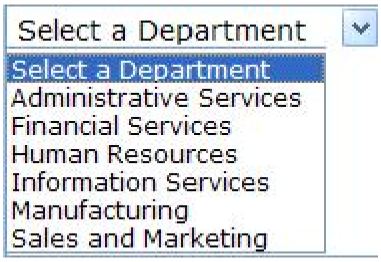
```
var oROW = { value: "", text: "Select a Department" };
this.addTableROW (oROW, true);
```

Step 4.

Save your changes, (Edit and resize the field if necessary).

Restart aXes.

You now have a drop down that allows the user to select any of the departments in the XHRDEPT database file.



Static Tables and SQL Variables

Static tables can also be filled by executing SQL commands.

Every static table is filled the first time any static table is referenced by a script. They are normally filled only once and they remain constant until the 5250 session ends.

The SQL commands used to fill static table can also contain SQLVariableXXXXX substitution names, just like Dynamic Tables.

Imagine you have a set of static tables whose content relates to the company number that user selects as they log on.
This means you need to pass SQLVariableCompanyNumber (say) to the routine that loads all the static tables.
You also need to do this before any drop down or other scripting tries to use the static table contents.

To do this you need to do something like this in your scripting before any drop down or scripting attempts to use any static table:

```
// Create an object that defines all the SQLVariables required

   Var SQLVariables = { SQLVariableCompanyNumber : sCompany ,
                        SQLVariableLibrary : USERENV.dftSQLDataLibrary  };

// Ask the table manager to load the static tables and give it the SQLVariables
// that it needs to execute any SQL commands defined in the static table file

  TABLEMANAGER.loadStaticTables(USERENV.staticTablesFile, SQLVariables, false);
```

This requests that all the static tables defined in the server file defined in the file named in USERENV.staticTablesFile (eg: "tables_static.txt") should be loaded now.
Where the static tables are being loaded from SQL commands the variables SQLVariableCompanyNumber and SQLVariableLibrary should be substituted.

The last parameter indicates that a (re)load of the static tables should be forced even if they have already been loaded. Here it is passed as false, but using true might be appropriate if the user had just changed the company they were using in the 5250 application.

Using Several Static Table Files

You don’t have to define all the Static Tables in one server file.

By default the server's static tables definition file is defined by USERENV.staticTablesFile which is shipped containing "tables_static.txt".

When you manually load tables you could do this to aggregate 3 sets of static tables:

```
   TABLEMANAGER.loadStaticTables("Static_Set_1.txt", null, false);
   TABLEMANAGER.loadStaticTables("Static_Set_2.txt", null, true);
   TABLEMANAGER.loadStaticTables("Static_Set_3.txt", null, true);
```

Or even do this to selectively load the static tables from different data sources:

```
If (CompanyNumber == "01")
   TABLEMANAGER.loadStaticTables("Static_Company_001.txt", null, false);
else
   TABLEMANAGER.loadStaticTables("Static_Company_002.txt", null, false);
```

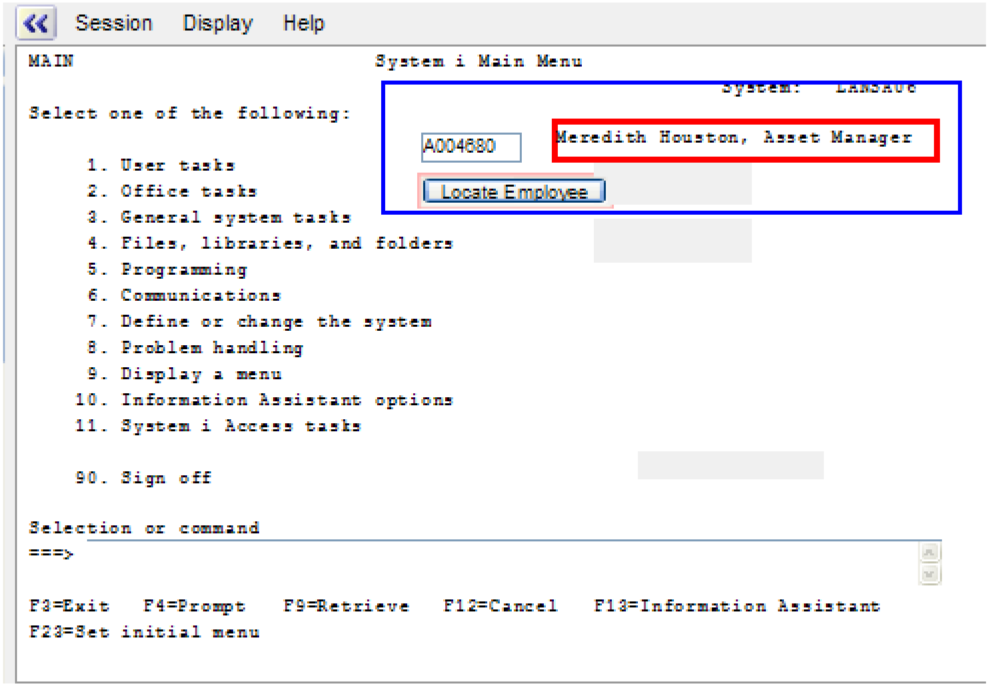
The key thing to remember is that you need to do this in your scripting before any drop down requests that the static tables are loaded.

Dynamic Tables

Like static tables, dynamic tables can be used to do a lot more than just load data into combo boxes.

The only difference between a dynamic and a static table is that the data sourced from a static table persists for the entire aXes session. Dynamic table data however is refreshed each time there is a screen interaction where the screen contains an eXtension using the dynamic table.

In this tutorial step you are going to add an employee enquiry to the IBM i Main menu that retrieves employee details based on the employee number:



```
MAIN                          System i Main Menu
                                                     System:   LANSA07
Select one of the following:
                                  A004680   Meredith Houston, Asset Manager
     1. User tasks
     2. Office tasks
     3. General system tasks
     4. Files, libraries, and folders
     5. Programming
     6. Communications
     7. Define or change the system
     8. Problem handling
     9. Display a menu
    10. Information Assistant options
    11. System i Access tasks

    90. Sign off

Selection or command
===>

F3=Exit   F4=Prompt   F9=Retrieve   F12=Cancel   F13=Information Assistant
F23=Set initial menu
```

First, open the *Projects Home Page*. Select your project and then use the *Edit Dynamic Tables* option to open **tables_dynamic.txt**. Add this dynamic table definition to it:

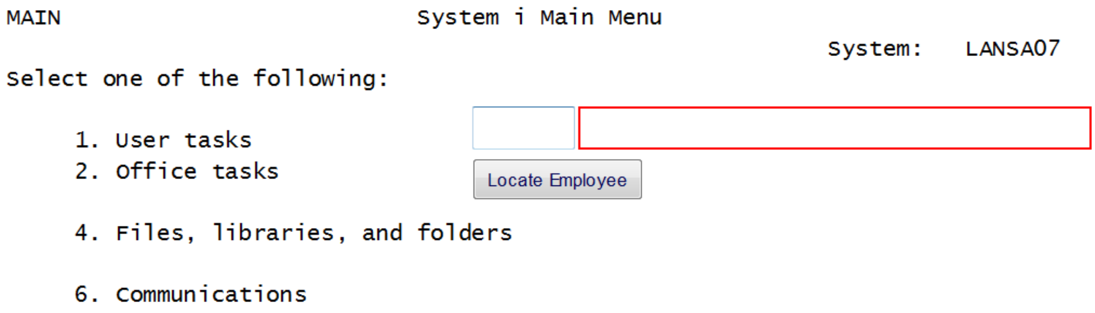
```
-- ===========================================================================
-- Look up the details of an Employee
-- ===========================================================================

    DefineObjectInstance {
      className          = "DynamicTable",
      name               = "FetchBasicEmployeeInfo",
      source             = "sql",
      selectSQLcommand   = "XHRSURNME,XHRGIVNME, XHRJOBTLE from AXESDEMO.XHREMPTN where
XHREMPID = ':SQLVariable_RequestedNumber' ",
      resultColumnNames   = { "lastName", "firstName", "jobTitle" },
    };
```

This dynamic table is named FetchBasicEmployeeInfo.

It requires the caller to provide an employee number in a variable named SQLVariable_RequestedNumber.

It reads the name and title details from the employee master file shipped in the AXESDEMO library, returning them with the values lastName, firstName and jobTitle respectively.

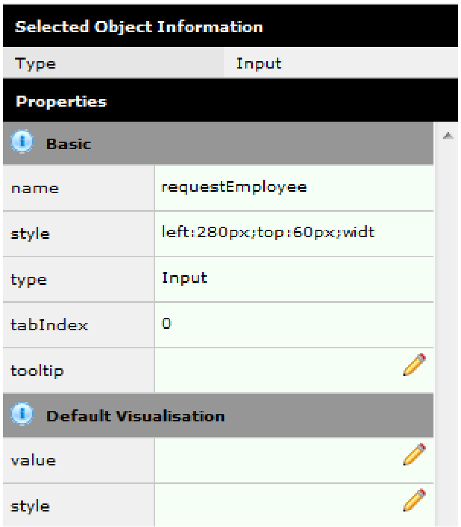
Now, alter the System i Main menu so that it can support employee enquiries, for example like this:



```
MAIN                          System i Main Menu
                                                     System:   LANSA07
Select one of the following:

     1. User tasks
     2. Office tasks

     4. Files, libraries, and folders

     6. Communications
```

To do this add 3 new elements to the 5250 screen.

The first is a Default Visualization input field named **requestEmployee**:



Selected Object Information	
Type	Input

Properties

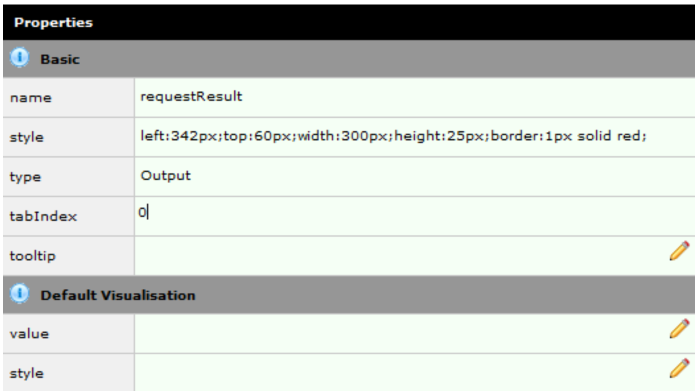
Basic

name	requestEmployee
style	left:280px;top:60px;widt
type	Input
tabIndex	0
tooltip	✎

Default Visualisation

value	✎
style	✎

The second is an output result field named **requestResult**, with a 1px solid red border:



Properties

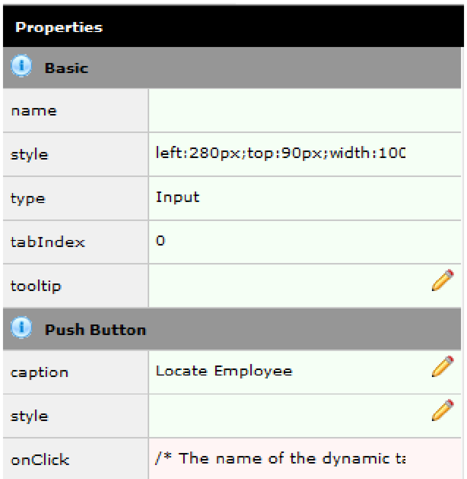
Basic

name	requestResult
style	left:342px;top:60px;width:300px;height:25px;border:1px solid red;
type	Output
tabIndex	0
tooltip	✎

Default Visualisation

value	✎
style	✎

And the last is a push button with caption *Locate Employee*:



Properties

Basic

name	
style	left:280px;top:90px;width:10C
type	Input
tabIndex	0
tooltip	✎

Push Button

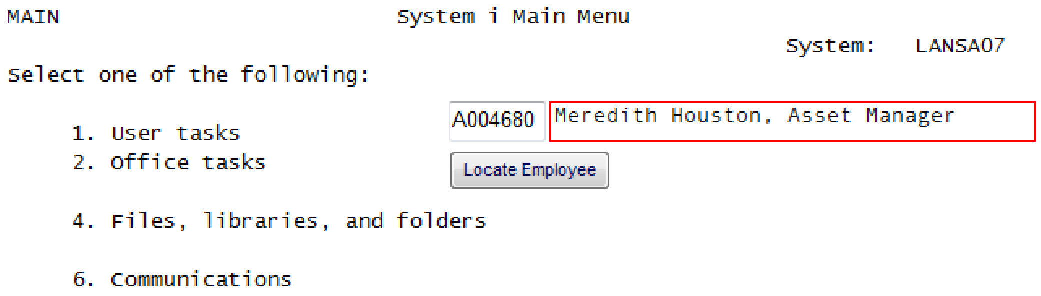
caption	Locate Employee ✎
style	✎
onClick	/* The name of the dynamic ta

The important part is this: add the following code to the **onClick** script:

```
/* The name of the dynamic table being used */

var sDynamicTable = "FetchBasicEmployeeInfo";

/* Get the employee number input and convert to upper case */

var sRequestedNumber = FIELDS("requestEmployee").getValue();
sRequestedNumber = sRequestedNumber.toUpperCase();

/* As the manager to load the dynamic table, passing the requested employee number */

TABLEMANAGER.loadDynamicTable(sDynamicTable,USERENV.dynamicTablesFile,{SQLVariable_RequestedN
umber : sRequestedNumber}, false );

/* Get row 0 from the result produced in the dynamic table */

var oRow = TABLEMANAGER.getTable(sDynamicTable).child(0);

/* If no resulting row was found then show error message in result field */

if (oRow == null)
FIELDS("requestResult").setValue("** NOT FOUND **");
else

   /* Format up the name and job title into the result field */

FIELDS("requestResult").setValue(oRow.firstName + " " + oRow.lastName + ", " +
oRow.jobTitle);
```

What this script does is to get the uppercased value of the employee number entered in field **requestEmployee**. It then asks the table manager to create the dynamic table named FetchBasicEmployee, passing in the employee number as an SQL variable. Then row 0 of the result is checked, and either an error message or the employee details are output onto the screen in the **requestResult** field.

Save the extensions and enter employee numbers like A002450, A004680 or A008550 in the requestEmployee field and click the button. You will see this:



```
MAIN                              System i Main Menu
                                                            System:    LANSA07
Select one of the following:
                                           A004680  Meredith Houston, Asset Manager
     1. User tasks
     2. Office tasks                       Locate Employee

     4. Files, libraries, and folders

     6. Communications
```

You have now added a new and quite foreign capability to the System i Main menu!

At one level this example is a bit silly, but at another level it demonstrates a very powerful facility.

Imagine the screen was not the System i Main Menu, but an *Order Details* screen, and the button said *DHL Delivery Status*.

When the button is clicked an order/DHL cross reference table is accessed on the server to provide an associated DHL consignment number. The DHL consignment number is then used to open a web browser window (provided by DHL) to display the order's delivery status.

Frequently Asked Questions about this Example

Q: Does the library name have to be hard coded?
No. You just can use "... from **XHREMPTN** where ..." if you want.

Q: What happens if the library name is not specified?
The axes server's current library list is used to locate the table.

Q: How can I avoid using a library name?
Leave it out of the SQL commands and put the required library(s) into the aXes server job's library list.

Q: I sign on to an aXes 5250 session as user FRED - is user FRED's library list used to find the file?
No. The library list used is the library list of the aXes server job.

Q: Can I dynamically change the aXes server's library list
This would not be advisable. Server requests execute in a multi-threaded, multi-user stateless environment. Even if you changed the server jobs library list, another thread might change it 2 milliseconds later – even before you could get to perform your SQL request.

Q: So is the library list solution suitable for all situations?
No. In situations where you need different library lists for different users (say) or different companies (say) to support access to different databases you generally need to specify the exact library name for SQL requests.

Q: How can I make the library name soft?
By making it an **SQLVariable** like the employee number in this example.

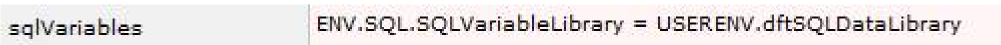
The preceding example could be coded as:

```
DefineObjectInstance {
  className         = "DynamicTable",
  name              = "FetchBasicEmployeeInfo",
  source            = "sql",
  selectSQLcommand  = "XHRSURNME,XHRGIVNME, XHRJOBTLE from
                       :SQLVariableLibrary.XHREMPTN where
                       XHREMPID = ':SQLVariable_RequestedNumber' ",
  resultColumnNames  = { "lastName", "firstName", "jobTitle" },
};
```

Now an SQLVarible named **SQLVariableLibrary** will be substituted into the SQL command before it is executed.
This is no different to substituting the employee number or any other value.

Q: Where does SQLVariableLibrary come from?
It needs to be supplied by the eXtension script that causes the SQL command to be executed.

Q How is the value of SQLVariable set?
By normal eXtension scripting. If you look at the drop down eXtension's *sqlVariables* property you will see this default value:



sqlVariables	ENV.SQL.SQLVariableLibrary = USERENV.dftSQLDataLibrary

So when a drop down is to be filled from an SQL command, by default, the SQL variable named SQLVariableLibrary is passed to the server from the USERENV's dftSQLDataLibrary property. If you look at the USERENV object definition you will see this line:

```
/* ------------------------------------------------ */
/* Properties defined as part of the USERENV object */
/* ------------------------------------------------ */

staticTablesFile    : "tables_static.txt",
dynamicTablesFile   : "tables_dynamic.txt",
dftSQLDataLibrary   : "QGPL",
```

So by default, any SQL command used to fill a drop down has a variable named SQLVariableLibrary available to it for substitution.
Also, by default, it will contain QGPL as the library name.

You could do this in any individual drop down's sqlVariables property:

```
ENV.SQL.SQLVariableLibrary = "MYDATALIB";
```

or your could do this in the USERENV object:

```
dftSQLDataLibrary : "MYOTHERLIB",
```

to change the SQLVariableLibrary value being sent to the server.

Q: Does the variable have to be named SQLVariableLibrary?
No. You can use any SQLVariablexxxxx name you like.

You might also have multiple names like SQLVariableGlobalLibrary and SQLVariableCompanyLibrary to separate shared global databases from individual company databases.

Q: How can I work out different library names for different situations?
However you like.

Imagine your 5250 application made the user select a company as they logged on. In the screen onLeave script you could code something like this:

```
var CompanyNumber = FIELDS("CompanyNumber").getValue()

switch (CompanyNumber)
{
   case "01": USERENV.dftSQLDataLibrary = "DLCOMP_01"; break;
   case "02": USERENV.dftSQLDataLibrary = "DLCOMP02";   break;
   case "03": USERENV.dftSQLDataLibrary = "DLCOMP_3R"; break;
   case "04": USERENV.dftSQLDataLibrary = "DLCOMP4AG"; break;
}
```

This sets USERENV.dftSQLDataLibrary to different library names based on the selected company number for use by all subsequent SQL commands.

More sophisticated and more generic examples of this can themselves execute an SQL command to look up a table by company number (say) or user id (say) to determine the names of the libraries to be used. This is similar to logic commonly used in many IBM i application 5250 logon programs. See Some Usage Ideas.

A tip for setting up SQL commands in the dynamic tables file

Before you create a dynamic table definition, it's a good idea to first try out your SQL command in an SQL command line session to get rid of any mistakes.

In a 5250 session use the STRSQL command to start a SQL command line session. Try out your intended SQL command, for example:

```
       SELECT XHREMPID, XHRGIVNME,XHRSTREET,XHRCITY FROM AXESDEMO/XHREMPTN
       WHERE UPPER(XHRGIVNME) LIKE '%IN%'
```

which is a list of all employees whose name (in uppercase) contains the letters "IN". Test it until you get it right, then copy/paste the command into your aXes dynamic tables file and generalize it as required - for example:

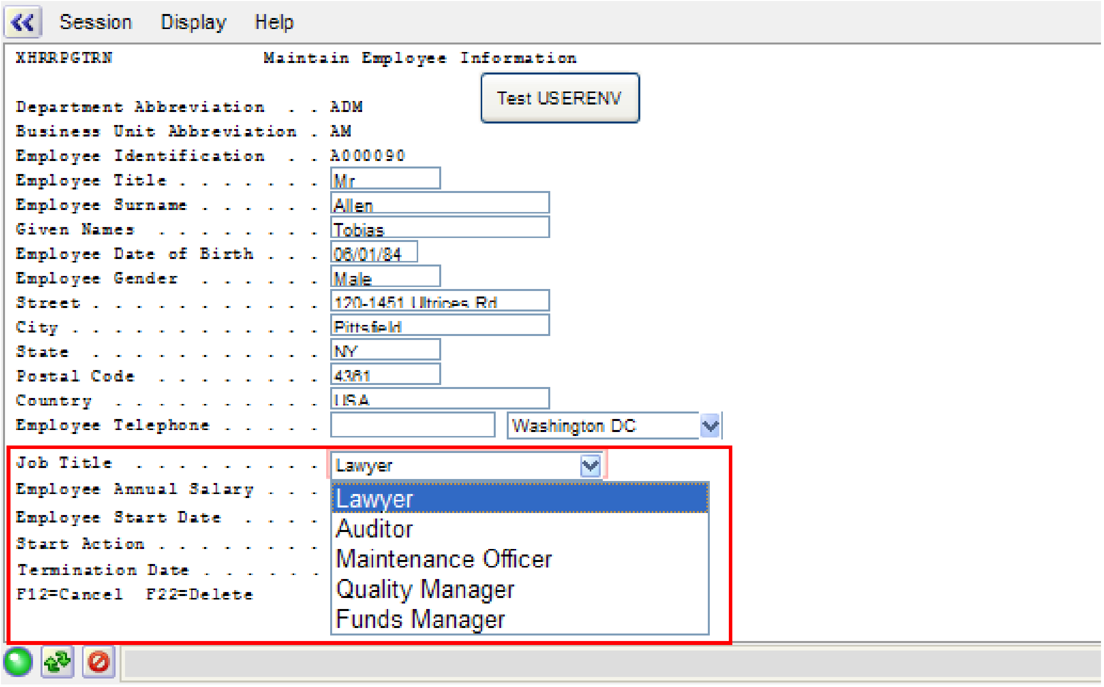
```
    DefineObjectInstance
    {
      className          = "DynamicTable",
      name               = "AnExample",
      source             = "sql",
      selectSQLcommand   = "XHREMPID, XHRGIVNME,XHRSTREET,XHRCITY FROM
                           :SQLVariable_DataLibrary.XHREMPTN
                           WHERE UPPER(XHRGIVNME) LIKE '%:SQLVariable_PartialName%' ",
    };
```

Working this way might save you several cycles of dynamic table updating to get your SQL command right.

BTW: You have to define the SQL commands in a server based file because this makes it difficult for a browser client to change what columns are selected. For security reasons only SELECT (read) SQL commands should ever be allowed. Developers need to be careful not to get too inventive with SQL variables and accidentally allow simple SQL injection attacks.

XML Documents

In this step you will use data from an XML document to populate a drop down with job titles.



```
XHRRPGTRN                    Maintain Employee Information

Department Abbreviation  . . ADM
Business Unit Abbreviation . AM
Employee Identification  . . A000090
Employee Title . . . . . . . Mr
Employee Surname . . . . . . Allen
Given Names  . . . . . . . . Tobias
Employee Date of Birth . . . 06/01/84
Employee Gender  . . . . . . Male
Street . . . . . . . . . . . 120-1451 Ultrices Rd
City . . . . . . . . . . . . Pittsfield
State  . . . . . . . . . . . NY
Postal Code  . . . . . . . . 4361
Country  . . . . . . . . . . USA
Employee Telephone . . . . .            Washington DC
Job Title  . . . . . . . . . Lawyer
Employee Annual Salary . . .
Employee Start Date  . . . .
Start Action . . . . . . . .
Termination Date . . . . . .
F12=Cancel  F22=Delete
```

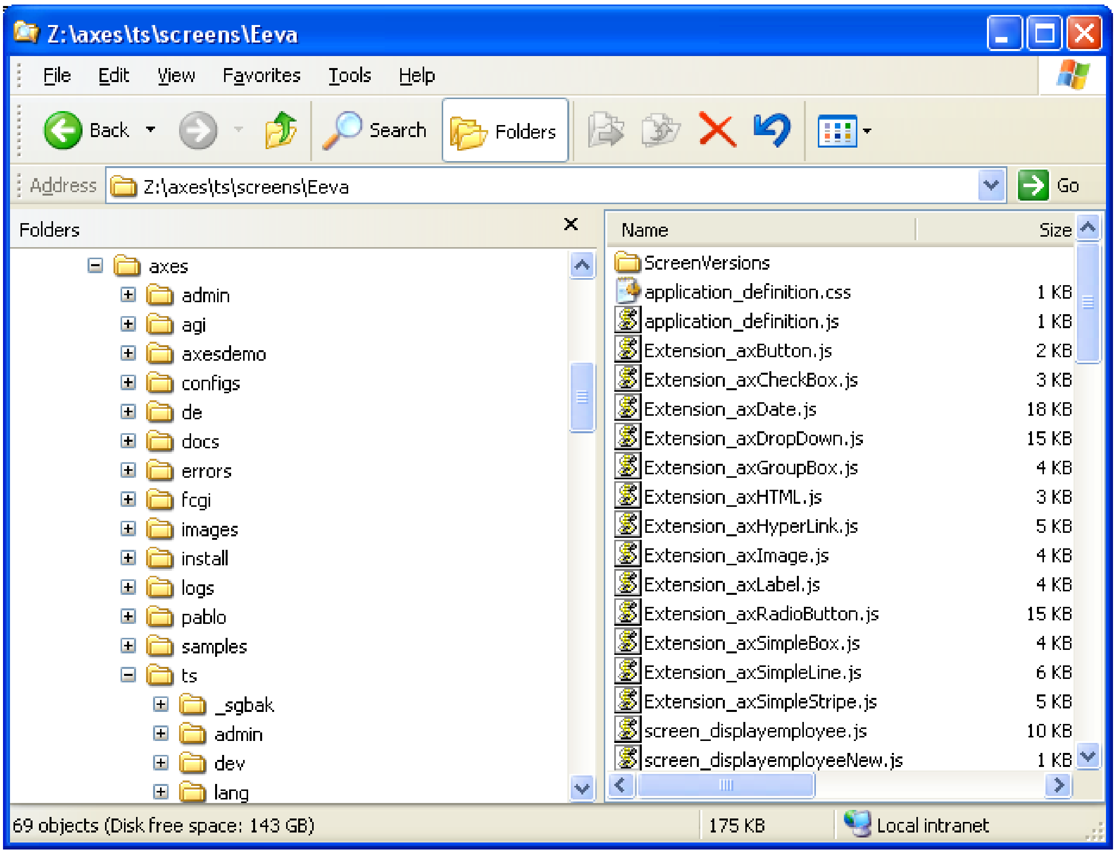
Step 1.

Use a text editor like notepad to copy/paste this XML:

```xml
<?xml version="1.0" encoding="iso-8859-1"?>

<jobtitles>
   <title>Lawyer</title>
   <title>Auditor</title>
   <title>Maintenance Officer</title>
   <title>Quality Manager</title>
   <title>Funds Manager</title>
</jobtitles>
```

Note: you can add more `<title>` nodes if you wish.

Save this file as **JobTitles_en.xml** in the axes\ts\screens folder or your private definition set folder, for example:
\axes\ts\screens\eeva (where eeva is the name of your private definition set folder).



Z:\axes\ts\screens\Eeva

- axes
  - admin
  - agi
  - axesdemo
  - configs
  - de
  - docs
  - errors
  - fcgi
  - images
  - install
  - logs
  - pablo
  - samples
  - ts
    - _sgbak
    - admin
    - dev
    - lang

Name	Size
ScreenVersions	
application_definition.css	1 KB
application_definition.js	1 KB
Extension_axButton.js	2 KB
Extension_axCheckBox.js	3 KB
Extension_axDate.js	18 KB
Extension_axDropDown.js	15 KB
Extension_axGroupBox.js	4 KB
Extension_axHTML.js	3 KB
Extension_axHyperLink.js	5 KB
Extension_axImage.js	4 KB
Extension_axLabel.js	4 KB
Extension_axRadioButton.js	15 KB
Extension_axSimpleBox.js	4 KB
Extension_axSimpleLine.js	6 KB
Extension_axSimpleStripe.js	5 KB
screen_displayemployee.js	10 KB
screen_displayemployeeNew.js	1 KB

69 objects (Disk free space: 143 GB) 175 KB Local intranet

Step 2.

Go to the screen identified in Tutorial 1 as **XHRRPGTRN_Maint**
In **XHRRPGTRN_Maint** identify the field Job Title as **`Job_Title`**

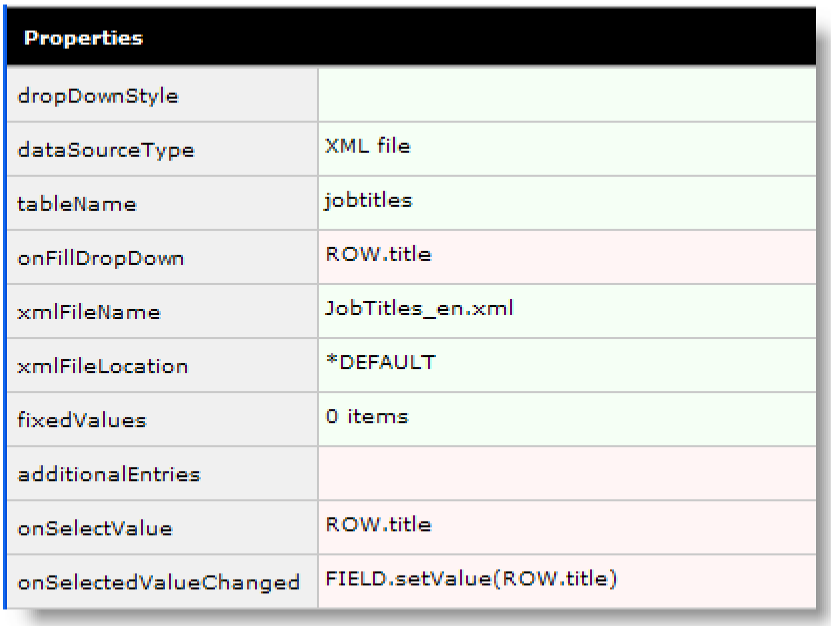
Step 3.

Click on the **`Job_Title`** field to select it and make it a Drop Down by checking the Drop Down eXtension.

Step 4.

Set these Drop Down properties:

```
dataSourceType: XML file
tableName: JobTitles
onFillDropDown: ROW.title
xmlFileName: JobTitles_en.xml
onSelectValue: ROW.title
onSelectedValueChanged: FIELD.setValue(ROW.title)
```



Properties	
dropDownStyle	
dataSourceType	XML file
tableName	jobtitles
onFillDropDown	ROW.title
xmlFileName	JobTitles_en.xml
xmlFileLocation	*DEFAULT
fixedValues	0 items
additionalEntries	
onSelectValue	ROW.title
onSelectedValueChanged	FIELD.setValue(ROW.title)

Note that the value *DEFAULT for xmlFileLocation means either /ts/screens/<definition set> if there is a definition set specified in the URL, Otherwise it means /ts/screens/.

Step 5.

Save the screen and test the drop-down.

Step 6.

Note that using the LANGUAGE predefined variable you can fill the drop down with entries sourced from language specific data.

Copy the file JobTitles_**en**.xml to a file called JobTitles_**es**.xml.

Step 7.

Replace the contents with this:

```
<?xml version="1.0" encoding="iso-8859-1"?>

<jobtitles>
   <title>Abogado</title>
   <title>Auditor</title>
   <title>Oficial de Mantenimiento</title>
   <title>Gerente de Control de Calidad</title>
   <title>Gerente de Cartera</title>
</jobtitles>
```

Step 7.

Edit the screen and change the xmlFileName property like this:

```
ENV.returnValue = "JobTitles_" + LANGUAGE + ".xml";
```

The result of this statement will be the JobTitles_<languagecode>.xml.

Step 8.

Add *lang=es* at the end of the URL and you will notice how the drop down will now have the Spanish language entries from the Spanish JobTitles_es.xml.

For example http://myhost/ts/skins/ts.html?lang=es

Performance Considerations

Once the data has been stored in a table it will be reused for the duration of the session *except when a dynamic table* is used.

Usage of dynamic tables should be carefully considered because they will most definitely have a performance impact.

Each time a screen arrives, for each extension whose data is sourced from a Dynamic Table, the query request will run to get the data from the server and store the data in the table.

The performance of a dynamic table can be improved if the same request is commonly issued repeatedly, by using the keepLastKey option. This causes Axes to store the key of the last request, and if another request is made for the same key, the results of the last request are reused (without any access to the server).

keepLastKey can be specified on the extensions that can use a dynamic table (The drop down and radio buttons extensions), or it can be specified in a script as the 4th parameter of the Axes.TableManager.loadDynamicTable function)

Some Usage Ideas

You can use tables for a lot more than just filling combo boxes and setting up radio button sets.

For example, a common system design requirement is a set of information that defines the characteristics of a system. If you were writing an RPG server based application you would typically put such information into a data area or a data base table.

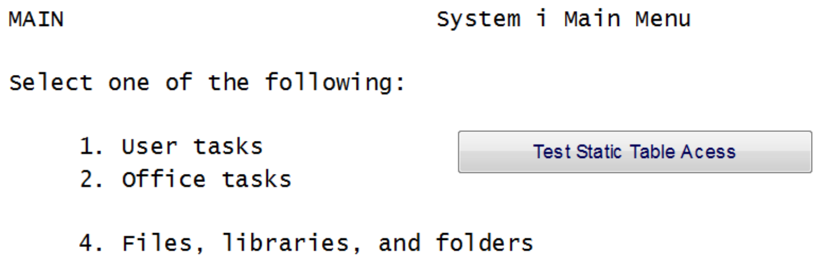
Here's an example of doing this with a static table.

Open the *Projects Home Page*. Select your project and then use the *Edit Static Tables* option.

Add this new static table into your application:

```
-- ===================================================================================
-- MySystemInfo - Single row static table containing information about my system
-- ===================================================================================

   DefineObjectInstance {
      className = "StaticTable",
      name      = "MySystemInfo",
      source    = "inline",
      rows      = {
                    {
                      companyName          = "Acme and Acme",
                      companyWebSite       = "www.mycompany.com",
                      productVersion       = "1.0",
                      defaultLanguage      = "English",
                      defaultOS400Library  = "MYLIBRARY"
                    }
                 }
    };
```

Now add a push button like this to the IBM i Main menu:



```
MAIN                              System i Main Menu

Select one of the following:

     1. User tasks
     2. Office tasks

     4. Files, libraries, and folders
```

For the **onClick** property use this scripting:

```
/* Load all static tables, in case they are not already been loaded */
/* If they are already loaded this request will just be ignored     */

TABLEMANAGER.loadStaticTables(USERENV.staticTablesFile);
```

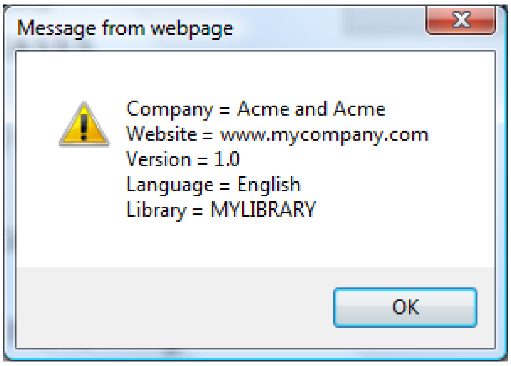
```
/* Now get a reference to the "MySystemInfo" static table defined on the server */
/* Then put a reference to row/child 0 directly into the USERENV object         */

USERENV.systemInfo = TABLEMANAGER.getTable("MySystemInfo").child(0);

if (USERENV.systemInfo == null)
{
  window.alert("MySystemInfo row could not be read");
}
else
{
  var sMessage = "";
  sMessage += "Company = " + USERENV.systemInfo.companyName + "\r";
  sMessage += "Website = " + USERENV.systemInfo.companyWebSite + "\r";
  sMessage += "Version = " + USERENV.systemInfo.productVersion + "\r";
  sMessage += "Language = " + USERENV.systemInfo.defaultLanguage + "\r";
  sMessage += "Library = " + USERENV.systemInfo.defaultOS400Library + "\r";
  window.alert(sMessage);
}

/* USERENV.systemInfo is now a real script object, so other scripts can */
/* now just access USERENV.systemInfo.companyWebSite (say) directly     */
```

Save your screen changes.

You may have already loaded the static tables, so to be sure, close and restart your aXes development session.

Redisplay the IBM i Main Menu and click the *Test Static Table Access* button.

You should see a message box like this:



Message from webpage

Company = Acme and Acme
Website = www.mycompany.com
Version = 1.0
Language = English
Library = MYLIBRARY

This example demonstrates how a static table may be used to provide useful control information scripts.

Controlling Axes Using a System Definition Table

A common requirement in aXes implementations is to set up soft coded values that can be used to generically control your eXtension scripts and aXes application.

Such soft coded values are variously referred to as system values, settings, definitions or properties.

> Note: Every application ever created has such settings to change how the application behaves at each deployment site or even for each user. Typically the settings are permanently stored in DB2/400 data base tables or in IBM i data areas.

The following material describes a simple and extensible technique for setting up soft coded system definitions for your aXes applications. The definitions are easily accessed by eXtension scripts, almost infinitely extensible and easy to deploy.

If you are interested in IT technologies this technique is a practical example of using **JSON** in your aXes applications.

Step 1 – Set up your system definition data base table

In this example a data base table (an IBM i physical file) named MYSYSDEF is created in library QGPL. The SQL command to create this table could be like this - but you could just easily create the table using traditional DDS ...

```
        CREATE TABLE QGPL/MYSYSDEF (SYSNAME CHAR (10) NOT NULL, JSONDATA CHAR (500) NOT NULL)
```

This table is logically keyed by a char(10) system name.

The char(500) JSONDATA field will store the system definition values - referred to as properties from now on.

The example system definition properties to be stored in table MYSYSDEF are:

⇨ *The company name.*

⇨ *The URL to be used for web search requests.*

⇨ *A flag indicating whether users are allowed to do web searches.*

Step 2 – Define a dynamic query to read data base table MYSYSDEF

Your eXtension scripting will need to be able to read the MYSYSDEF table - so you need to define an SQL query into your project's ***Dynamic Tables*** definition file:

```
DefineObjectInstance {
   className           = "DynamicTable",
   name                = "MYSYSDEF",
   source              = "sql",
   selectSQLcommand    = "JSONDATA from QGPL.MYSYSDEF where SYSNAME = ':SQLVariable_System' ",
   resultColumnNames   = { "JSONDATA" },
   };
```

Save your changes and restart any developer sessions.

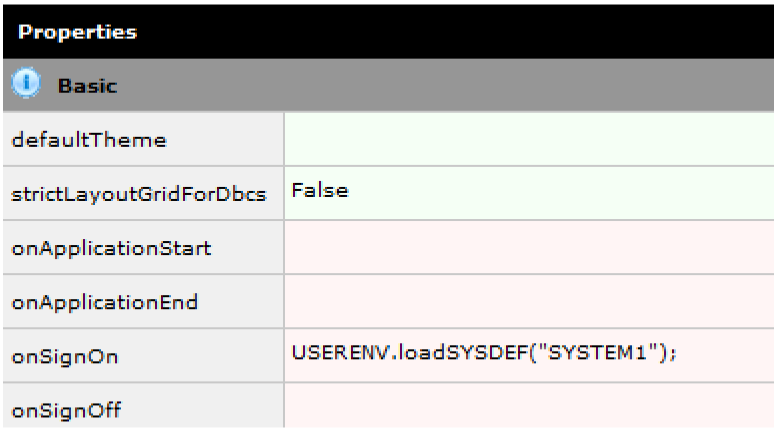
Step 3 – Read data base table MYSYSDEF when the user logs on

Put this function into your project's USERENV.JS file - as part of your USERENV object:

```
   /* ---------------------------------------------------------------------------- */
   /* Load the system definition from MYSYSDEF file for the system name specified */
   /* ---------------------------------------------------------------------------- */

   loadSYSDEF : function(sysname)
   {
      /* Load the SQL table build by reading table MYSYSDEF for the specified system name.
The  */

AXES.TableManager.loadDynamicTable("MYSYSDEF",USERENV.dynamicTablesFile,{SQLVariable_System:s
ysname}, false);
      var oTable = AXES.TableManager.getTable("MYSYSDEF");
      var oChild = oTable.child(0);

      /* Handle not found */

      if (oChild == null)
      {
         USERENV.SYSDEF = {}; /* Create default empty USERENV.SYSDEF object */
         window.alert("USRENV.loadSYSDEF: MYSYSDEF load failed. No data available for system
named " + sysname );
      }

      /* Attempt to convert JSON data in USERENV.SYSDEF java script object */

      else
      {
          try
          {
             USERENV.SYSDEF = eval("({ " + oChild.JSONDATA + " })");
          }
          catch (oe)
          {
             USERENV.SYSDEF = {};  /* Create default empty USERENV.SYSDEF object */
             window.alert("USRENV.loadSYSDEF: Error " + oe.description + " detected when
loading JSONDATA from MYSYSDEF.");
          }
      }
      /* Insert the correct default values for all missing USERENV.SYSDEF properties   */
      /* This saves all later scripts from having to check whether the property exists */

      {
         var SYSDEF = USERENV.SYSDEF;
         if (SYSDEF.companyName   == null) SYSDEF.companyName  = "NOT AVAILABLE";
         if (SYSDEF.allowSearch   == null) SYSDEF.allowSearch  = false;
         if (SYSDEF.searchEngine  == null) SYSDEF.searchEngine = "http://www.google.com";
      }

      /* Finished */

      return;

   }, /* <-- remember the trailing comma */
```

Save your changes and restart any currently open developer sessions.

Now edit your project so that at application sign on it executes the new USERENV.loadSYSDEF("SYSTEM1"); function in the onSignOn event, like this:



Properties

Basic

defaultTheme	
strictLayoutGridForDbcs	False
onApplicationStart	
onApplicationEnd	
onSignOn	USERENV.loadSYSDEF("SYSTEM1");
onSignOff	

Save your changes.

Step 4 – Define your system values in the MYSYSDEF table

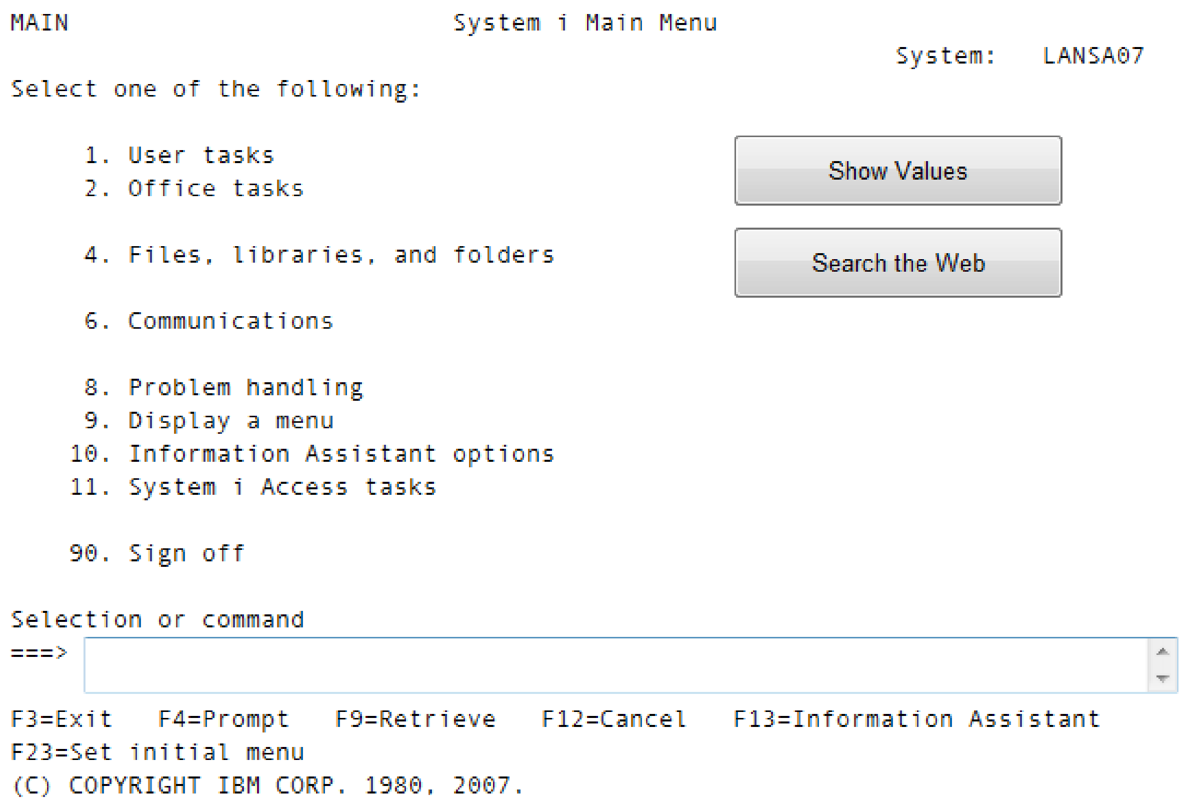
Using the IBM i command UPDDTA FILE(MYSYSDEF) or a similar application insert a row (record) in to the MYSYSDEF table as follows:

SYSNAME *Value*	***JSONDATA*** *Value- exact case is required*
SYSTEM1	companyName : "ACME", allowSearch : true

<u>Note</u>: Double check the entry for exact case, quotes, colons and commas.

Step 5 – Check that it all works okay

Name the IBM i main system menu and add two push button eXtensions.



```
MAIN                          System i Main Menu
                                                             System:   LANSA07
Select one of the following:

     1. User tasks
     2. Office tasks

     4. Files, libraries, and folders

     6. Communications

     8. Problem handling
     9. Display a menu
    10. Information Assistant options
    11. System i Access tasks

    90. Sign off

Selection or command
===>

F3=Exit   F4=Prompt   F9=Retrieve   F12=Cancel   F13=Information Assistant
F23=Set initial menu
(C) COPYRIGHT IBM CORP. 1980, 2007.
```

The first, titled ***Show Values*** should do this in its onClick event:

```
        window.alert("companyname=" + USERENV.SYSDEF.companyName);
        window.alert("allowSearch=" + USERENV.SYSDEF.allowSearch.toString());
        window.alert("searchEngine=" + USERENV.SYSDEF.searchEngine);
```

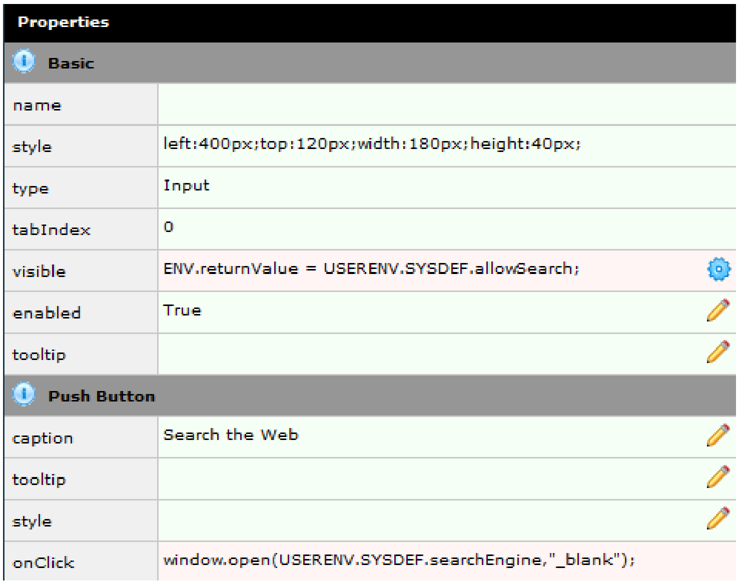
The second, titled ***Search the Web*** should do this in its onClick event:

```
        window.open(USERENV.SYSDEF.searchEngine,"_blank");
```

Additionally, the ***Search the Web*** button should have its visibility conditioned by altering the visible property to execute this script:

```
  ENV.returnValue = USERENV.SYSDEF.allowSearch;
```

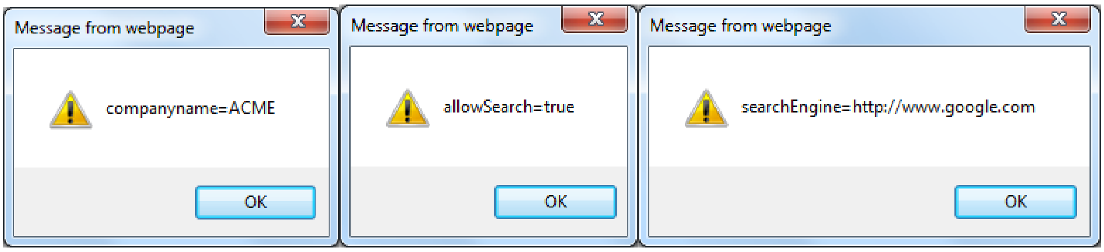
Like this :



Properties	
Basic	
name	
style	left:400px;top:120px;width:180px;height:40px;
type	Input
tabIndex	0
visible	ENV.returnValue = USERENV.SYSDEF.allowSearch;
enabled	True
tooltip	
Push Button	
caption	Search the Web
tooltip	
style	
onClick	window.open(USERENV.SYSDEF.searchEngine,"_blank");

Save your changes and start a ***new*** user 5250 session.

Click the ***Show Values*** button. You should see 3 message boxes like this:



companyname=ACME

allowSearch=true

searchEngine=http://www.google.com

Click the ***Search the Web*** button.

You should see a new web page open up with the Google search engine displayed.

Now - update the SYSTEM1 row in the MYSYSDEF data base table so that the field JSONDATA contains this data:

```
companyName : "Wigets", allowSearch : true, searchEngine : "http://www.bing.com"
```

being careful with case, colons, quotes and commas.

Start a ***new*** user session so that the updated MYSYSDEF data is loaded at log on.

When you click the ***Search Values*** button you should see the company name Widgets and the search engine URL as Microsoft's Bing.

When you click the ***Search the Web*** button you should see the Microsoft Bing search engine instead of the Google engine displayed.

Finally, update the JSONDATA field in the MYSYSDEF data base table to be like this:

```
companyName : "Wigets", allowSearch : false, searchEngine : "http://www.bing.com"
```

Start a ***new*** user session to pick up the modified data the MYSYSDEF table.

The ***Search the Web*** button should not appear because its visibility has been conditioned by USERENV.SYSINFO.allowSearch property – which is now false.

The key to all of this is that you have changed the behaviour of your aXes application without changing anything in a an eXtension script – you have only changed a row in the MYSYSDEF data base table.

Step 6 – Basic Concepts

The field or column JSONDATA in your MYSYSDEF table contains a JSON (**J**ava**S**cript **O**bject **N**otation) string.

When it is read from the MYSYSDEF table it can be converted directly to a JavaScript object by the JavaScript eval() function. This happens in this line of code in USERENV.loadSYSDEF():

```
USERENV.SYSDEF = eval("({ " + oChild.JSONDATA + " })");
```

The JSON string approach is very powerful because:

- It converts directly into a JavaScript object named USERENV.SYSDEF.

- It is extremely extensible because it allows you to invent new system definition properties at any time.

  For example if you change the JSONDATA field/column to:

  ```
  shoeSize :2, hatSize: 14, companyName : "Wigets", allowSearch : false, searchEngine :
  "http://www.bing.com"
  ```

then you have just invented two new USERENV.SYSDEF properties that you can immediately reference in your eXtension scripts as USERENV.SYSDEF.**shoeSize** and USERENV.SYSDEF.**hatSize**.

1. You can also define arrays and even JavaScript functions in JSONDATA - see the following optional advanced capabilities steps.

Step 7 – Infinite Extensibility , JSON Formatting and Default Values

Adding new properties is easy and almost infinitely extensible – you just add them to the JSONDATA field/column in your MYSYSDEF table and then you can immediately reference them in your eXtension scripts using the name format USERENV.SYSDEF.propertyname.

You can add numbers as name : number and strings as name : "string".

Strictly speaking the JSON format is:

"name" : numeric value or "name" : "string value"

Generally you can use either format - but the property names are always case sensitive.

You need to be careful with using the special ":" and "," separation characters.

If you get the syntax wrong then this line in USERENC.loadSYSDEF() will pop up:

```
window.alert("USRENV.loadSYSDEF: Error " + oe.description + " detected when loading
JSONDATA from MYSYSDEF.");
```

When adding new properties you need to consider aXes applications that you may have already deployed. For example, you could invent a new property called myNewProperty, add it to your MYSYSDEF table and then immediately start using it in your eXtension scripts.

However, when you deploy your application you might find that in the deployed application environment there is not a myNewproperty value defined in the MYSYSDEF table – so your application will get an error when it tries to reference the property.

This is very easily solved – when you add a new property ***always add a default value for it to the USERENV.loadSYSDEF()*** function. That is why this code exists in USERENV.loadSYSDEF and it is also why the first example displayed the search engine as Google …

```
/* Insert the correct default values for all missing USERENV.SYSDEF properties   */
/* This saves all later scripts from having to check whether the property exists */

{
   var SYSDEF = USERENV.SYSDEF;
   if (SYSDEF.companyName  == null) SYSDEF.companyName  = "NOT AVAILABLE";
   if (SYSDEF.allowSearch  == null) SYSDEF.allowSearch  = false;
   if (SYSDEF.searchEngine == null) SYSDEF.searchEngine = "http://www.google.com";
}
```

So you would add a line like this:

```
if (SYSDEF.myNewProperty == null) SYSDEF.myNewProperty = 500; /* Say */
```

or:

```
if (SYSDEF.myNewProperty == null) SYSDEF.myNewProperty = "YYYYYXXX"; /* Say */
```

so that USERENV.SYSDEF.myNewProperty always exists – even it is not defined by the JSONDATA string read from the MYSYSDEF data base table.

Note: This code is also a good place to document all the properties that exist in your USERENV.SYSDEF object.

Step 8 – Advanced Capabilities – Arrays of System Properties

You can easily define arrays in a JSON string.

Try adding this to the JSONDATA field/column in your MYSYSDEF data base table:

validTypes : ["A", "B", "C"]

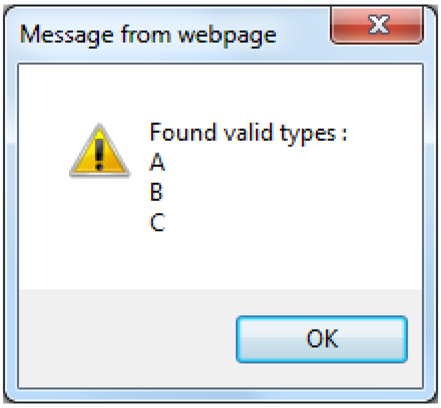
Following the rule of always setting up a default value for a new property you should add this to your USERENV.loadSYSDEF() function:

```
if (SYSDEF.validTypes == null) SYSDEF.validTypes = [ ]; /* empty array */
```

In an eXtension script activated by a push button execute this script:

```
var SYSDEF = USERENV.SYSDEF;
var message = "Found valid types :";

for (var index in SYSDEF.validTypes)
{
  message += "\r" + SYSDEF.validTypes[index]; /* Insert value preceded by a CR */
}

window.alert(message);
```

When the script is executed you should see:



Message from webpage

Found valid types :
A
B
C

You can also define arrays of number and even arrays of objects. For example:

```
Sizes : [ {x:1, y:2}, {x:3, y:4} , {x:5, y:6} ]
```

defines an array of objects. The array is named Sizes and each entry in the array is an object containing the properties x and y – which could be processed in a JavaScript loop like this:

```
var SYSDEF = USERENV.SYSDEF;

for (var index in SYSDEF.Sizes)
{
   var o = SYSDEF.Sizes[index];

   o.x and o.y are now accessible to the code.
}
```

Step 9 – Very Advanced Capabilities - Functions and "Soft" Logic

As a very advanced capability you can also define JavaScript functions in a JSON string.

First imagine a new function named showSearchEngine in your USERENV.SYSDEF object.

This function will contain the ***logic*** that is executed when a user requests a web search, rather than just the state that properties contain.

To try this out put a default value for the function into your USERENV.loadSYSDEF() function like this:

```
if (SYSDEF.showSearchEngine == null) SYSDEF.showSearchEngine = function(){};
```

So by default the showSearchEngine() function does nothing – it just starts and ends.

Save your changes.

Now change the "Search the Web" button to do this when it is clicked:

```
USERENV.SYSDEF.showSearchEngine();
```

This means when the "Web Search" button is clicked the function showSearchEngine() in the USERENV.SYSDEF object will be executed.

Start a ***new*** user session and verify that your ***Search the Web*** button now does nothing at all when it is clicked.

Now update your SYSTEM1 definition in the MYSYSDEF data base table to include this:

```
    showSearchEngine : function(){ window.open("http://www.google.com"); }
```

Start a ***new*** user session. You should find that now when you click the ***Search the Web*** button that a Google window opens.

Now update your SYSTEM1 definition in the MYSYSDEF data base table to include this:

```
    showSearchEngine : function(){ window.alert("Do not click this button"); }
```

Start a ***new*** user session. You should find that now when you click the ***Search the Web*** button that the message "Do not click this button" appears.

The key point here is that you have changed the ***logic*** in showSearchEngine by changing a record in a data base table on the server – you did not need to change any scripting on the client.

This is an example of how you can not only make USERENV.SYSDEF based properties "soft" – you can also make logic "soft" as well.

It would not be viable or sensible to try to script everything (or even most things) this way – but in situations where site dependent calculations or special variable logic is required this capability may be very powerfully used.

SQL and CCSIDs

When using SQL commands to load static or dynamic tables you may need to know about CCSID handling.

In this example a data base table named PBEMPF is used. It has these columns:

Name	**Type**	**Description**
EMPNO	Alpha(5)	Employee Number
EMNAM	Alpha(14)	Employee Name – Open field - CCSID 937
EMADR	Alpha(62)	Employee Address – Open field - CCSID 937

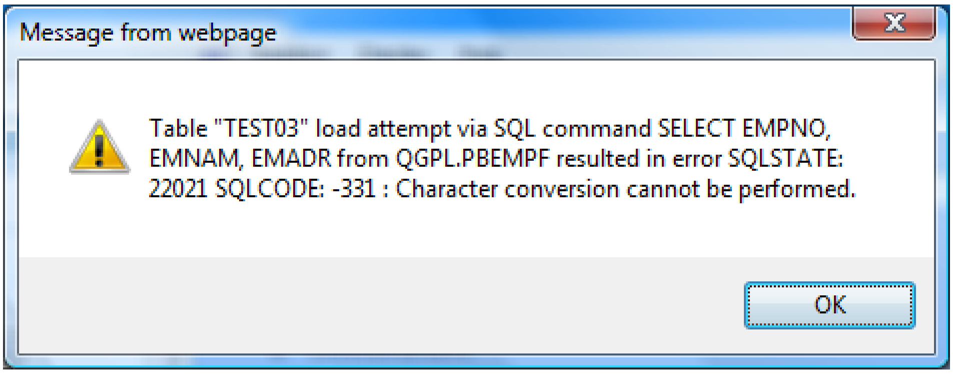
Note: CCSID 937 is Traditional Chinese.

The contents of table PBEMPF are to be loaded as a static table named TEST03.

This is done adding an instruction like this to the static table definition file:

```
DefineObjectInstance {
      className          = "StaticTable",
      name               = "TEST03",
      source             = "sql",
      selectSQLcommand   = "EMPNO, EMNAM, EMADR from QGPL.PBEMPF",
      resultColumnNames  = { "value", "text", "address" },
   };
```

However, when the static tables are being loaded this error is displayed:



Message from webpage

Table "TEST03" load attempt via SQL command SELECT EMPNO, EMNAM, EMADR from QGPL.PBEMPF resulted in error SQLSTATE: 22021 SQLCODE: -331 : Character conversion cannot be performed.

This error is indicating that data in the file connot be converted to the code page being used by the aXes server.

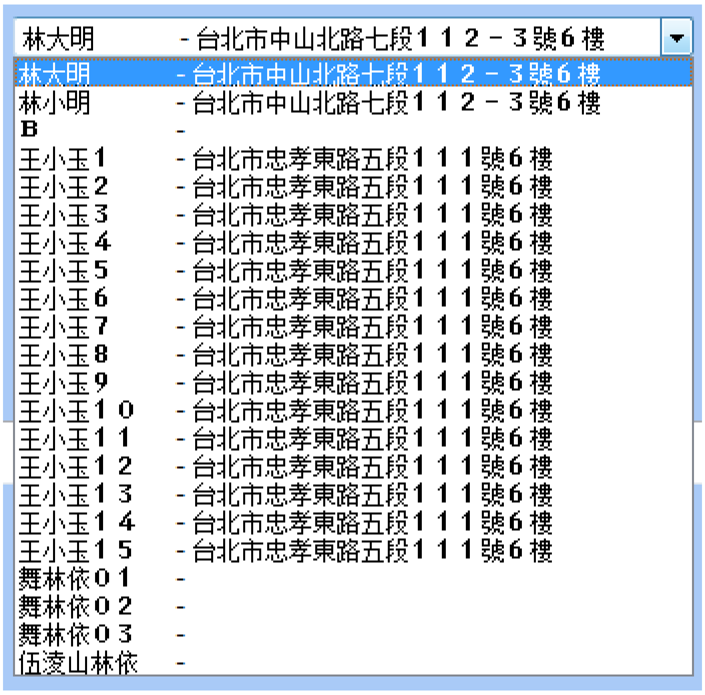
To avoid this issue the static table definition is changed to this:

```
  DefineObjectInstance {
      className           = "StaticTable",
      name                = "TEST03",
      source              = "sql",
      selectSQLcommand    = "EMPNO, CAST(EMNAM AS CHAR(14) CCSID 65535), CAST(EMADR AS
CHAR(62) CCSID 65535) from QGPL.PBEMPF",
      resultColumnNames   = { "value", "text", "address" },
      resultColumnCCSID   = { text=937, address=937 };
   };
```

The CAST operations indicate that ENNAM and EMADR are to be read as binary data.

The ResultColumnCCSID definition then informs aXes that the result columns logically named text and address should be treated as CCSID 937 (Traditional Chinese) data.

The EMPNAM and EMPADR values can now be used in a drop down eXtension, like this example displaying ROW.text + “ – “ + ROW.address:



```
林大明        - 台北市中山北路七段１１２－３號６樓
林大明        - 台北市中山北路七段１１２－３號６樓
林小明        - 台北市中山北路七段１１２－３號６樓
B             -
王小玉１      - 台北市忠孝東路五段１１１號６樓
王小玉２      - 台北市忠孝東路五段１１１號６樓
王小玉３      - 台北市忠孝東路五段１１１號６樓
王小玉４      - 台北市忠孝東路五段１１１號６樓
王小玉５      - 台北市忠孝東路五段１１１號６樓
王小玉６      - 台北市忠孝東路五段１１１號６樓
王小玉７      - 台北市忠孝東路五段１１１號６樓
王小玉８      - 台北市忠孝東路五段１１１號６樓
王小玉９      - 台北市忠孝東路五段１１１號６樓
王小玉１０    - 台北市忠孝東路五段１１１號６樓
王小玉１１    - 台北市忠孝東路五段１１１號６樓
王小玉１２    - 台北市忠孝東路五段１１１號６樓
王小玉１３    - 台北市忠孝東路五段１１１號６樓
王小玉１４    - 台北市忠孝東路五段１１１號６樓
王小玉１５    - 台北市忠孝東路五段１１１號６樓
舞林依０１    -
舞林依０２    -
舞林依０３    -
伍淩山林依    -
```

Tip for handling SQL variables and differing CCSIDs

Taking data from a Unicode based browser and passing it around for use in EBCDIC based SQL commands and databases can sometimes be slightly problematic.

Here's a tip for a way you may be able to get around Unicode and EBCDIC code page conversion situations using an SQL feature called a Unicode Literal.

In an SQL command you can type requests containing things like:
WHERE GREETING LIKE '%HELLO%'

You can also type:
WHERE GREETING LIKE **UX**'002500480045004C004C004F0025'

which is pretty much the same thing except that its meaning is absolutely exact in all code pages. Here 0025=%, 0048=H, 0045=E etc. are in Unicode hex format (for example see http://www.fileformat.info/info/unicode/char/48/index.htm for what a Unicode “H” is). This technique may be most useful when using SQLVariable substitution variables.

In a dynamic table SQL definition you can also code:
CUSTNAME LIKE ':SQLVariable_Name'

and then use the variable in client-side JavaScript like this:

```
        var SearchString = "%" + FIELDS("NAME").getValue() + "%";  /* A %value% scan value */
        var SQLVars = { SQLVariable_Name :  SearchString };
        TABLEMANAGER.loadDynamicTable("Test", etc, SQLVars, etc );
```

In the SQL definition you might also code something like this:
CUSTNAME LIKE **UX**':SQLVariable_Name'

And in the client-side JavaScript use a function that converts the JavaScript strings into UX style SQL literal values:

```
        var SearchString = "%" + FIELDS("NAME").getValue() + "%";
        var SQLVars = { SQLVariable_Name :  USERENV.toSQLUnicode(SearchString) };
        TABLEMANAGER.loadDynamicTable("Test", etc, SQLVars, etc );
```

The function **toSQLUnicode** used above is a simple function coded in the USERENV object:

```
        toSQLUnicode : function(s)
        {
           var r = "";
           if (s != null)
           {
               s = s.toString();
               for (var i = 0; i < s.length; i++)
               {
                  var cc = s.charCodeAt(i);
                  var uc = cc.toString(16).toUpperCase();
                  while (uc.length < 4) uc = "0" + uc;
                  r += uc;
               }
           }
           return(r);
        }, /* <- Remember the comma */
```

You can go further and specify the exact conversion by making the LIKE clause on the server into something like this:

CUSTNAME LIKE **CAST(UX':SQLVariable_Name' AS VARCHAR(50) CCSID 297)**

which takes the Unicode literal and then unequivocally converts it to EBCDIC CCSID 297 - which is then used for the LIKE comparison value - and is presumably the same code page as the field CUSTNAME.

You don’t normally need to do a CAST like this (you could experiment to confirm this), but casting may be significant with DBCS languages that need SO (Shift-Out) and SI (Shift-In) characters in their EBCDIC representations because shift characters are not normally present in Unicode strings.

If you want to try using some SQL UX literals manually using the IBM i's STRSQL command, here is the source for an HTML page that will turn what you type in into SQL UX literals ready for copy/paste into your 5250 STRSQL session:

```
<html>
<head>
<script type="text/javascript">
function  toSQLUnicode(s)
{
   var r = "";
   if (s != null)
   {
       s = s.toString();
       for (var i = 0; i < s.length; i++)
       {
          var cc = s.charCodeAt(i);
          var uc = cc.toString(16).toUpperCase();
          while (uc.length < 4) uc = "0" + uc;
          r += uc;
       }
   }
   return(r);
}
</script>
</head>
<body>
<input type="text" id="InputValue" />
<input type="button" value="Convert=&gt;" onclick="OutputValue.value =
toSQLUnicode(InputValue.value);" />
<input type="text" id="OutputValue" />
</body>
</html>
```

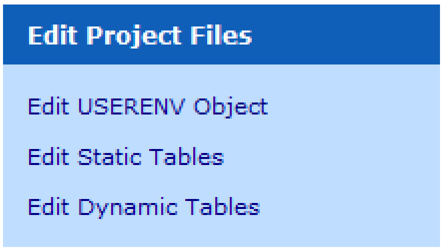
Using Dynamic Tables to Produce Spreadsheets and Reports

You can use Dynamic table definitions to produce spreadsheet data. This offers a convenient way to download data to a user's PC. Once the data is in a spreadsheet the user can format it, save it and print it as they like.

Dynamic tables have to be defined by a professional software developer so only approved and content audited downloads can be executed.

Note: Generic downloading tools may present some security and/or data content misinterpretation risks (eg: end users including deleted orders into quarterly sales revenue reports).

In this tutorial three dynamic SQL requests are added to the dynamic tables definition file. Use the *aXes Projects Home* page *Edit Dynamic Tables* option:

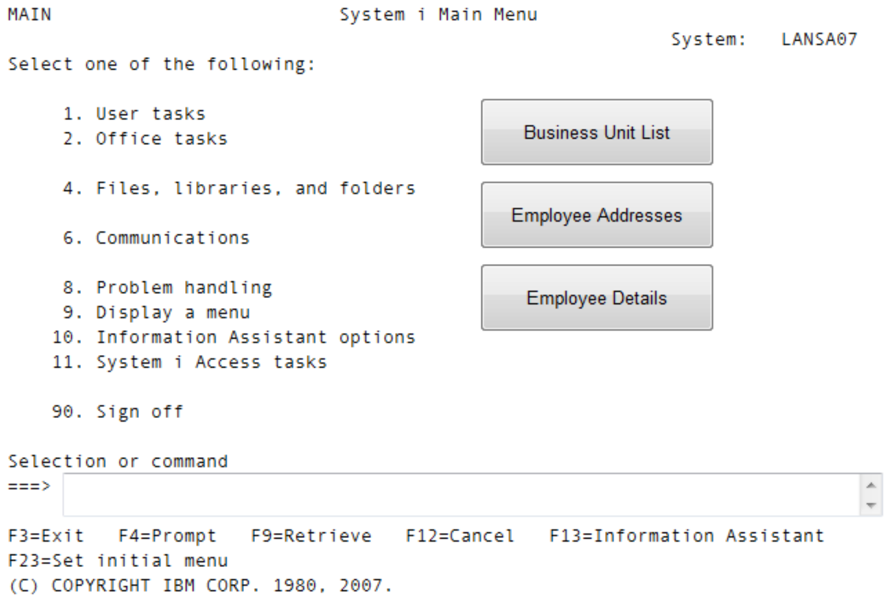


Then add these three definitions named Example1, Example2 and Example3:

```
    DefineObjectInstance {
      className           = "DynamicTable",
      name                = "Example1",
      source              = "sql",
      selectSQLcommand    = "XHRBUABRV, XHRBUSUNT from AXESDEMO.XHRBU",
      resultColumnNames    = { "value", "text" },
      resultColumnCaptions = { "Business Unit Code", "Business Unit Description" },
    };

    DefineObjectInstance {
      className           = "DynamicTable",
      name                = "Example2",
      source              = "sql",
      selectSQLcommand    = " XHREMPID, XHRGIVNME,XHRSTREET,XHRCITY,XHRSTATE from
AXESDEMO.XHREMPTN",
      resultColumnCaptions = { "Id", "First Name", "Street Address", "City", "State" },
    };

    DefineObjectInstance {
      className           = "DynamicTable",
      name                = "Example3",
      source              = "sql",
      selectSQLcommand   = "* from AXESDEMO.XHREMPTN where XHREMPID =
':SQLVariable_EmployeeId' ",
    };
```

Note: Any active development session needs to be closed and restarted to pick up these new dynamic table defintions:

Next three buttons elements are added to the System I Main Menu:



```
MAIN                          System i Main Menu
                                                          System:   LANSA07
Select one of the following:

     1. User tasks
     2. Office tasks

     4. Files, libraries, and folders

     6. Communications

     8. Problem handling
     9. Display a menu
    10. Information Assistant options
    11. System i Access tasks

    90. Sign off

Selection or command
===>

F3=Exit   F4=Prompt   F9=Retrieve   F12=Cancel   F13=Information Assistant
F23=Set initial menu
(C) COPYRIGHT IBM CORP. 1980, 2007.
```

Note: The Selection or command entry field is named *CommandLine* in this example.

The three new buttons are then given this *onClick* scripting:

Business Unit List Button

```
var result = TABLEMANAGER.convertDynamicTable("Example1",USERENV.dynamicTablesFile);
if (result.error == false)
{
   var URL = document.location.protocol + "//" + document.location.host + "/ts/" +
result.outputFileName;
   window.open(URL,"_blank");
}
```

Employee Address Details Button

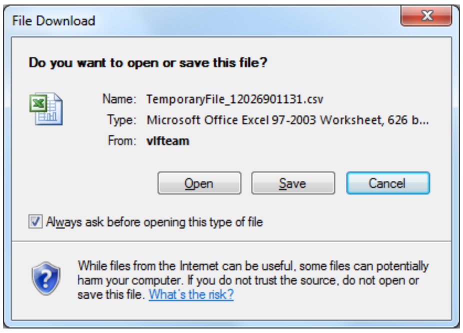
```
var result = TABLEMANAGER.convertDynamicTable("Example2",USERENV.dynamicTablesFile);
if (result.error == false)
{
   var URL = document.location.protocol + "//" + document.location.host + "/ts/" +
result.outputFileName;
   window.open(URL,"_blank");
}
```

Employee Details Button

```
var empid = FIELDS("CommandLine").getValue();
var requestdetails = { orientation:"V", SQLVariable_EmployeeId:empid };
var result =
TABLEMANAGER.convertDynamicTable("Example3",USERENV.dynamicTablesFile,requestdetails);
if (result.error == false)
{
   if (result.outputLineCount == 0)
   {
      window.alert("Employee with number " + empid + " not found. Is the number and the case
of the number correct?");
   }
   else
   {
      var URL = document.location.protocol + "//" + document.location.host + "/ts/" +
result.outputFileName;
      window.open(URL,"_blank");
   }
}
```

When the ***Business Unit List Button*** is clicked the SQL command named *Example1* is executed. The output sent to a CSV file. The name of the file produced is returned in result.outputFileName. The output file is then opened in a new browser window.

So the user should see:



File Download

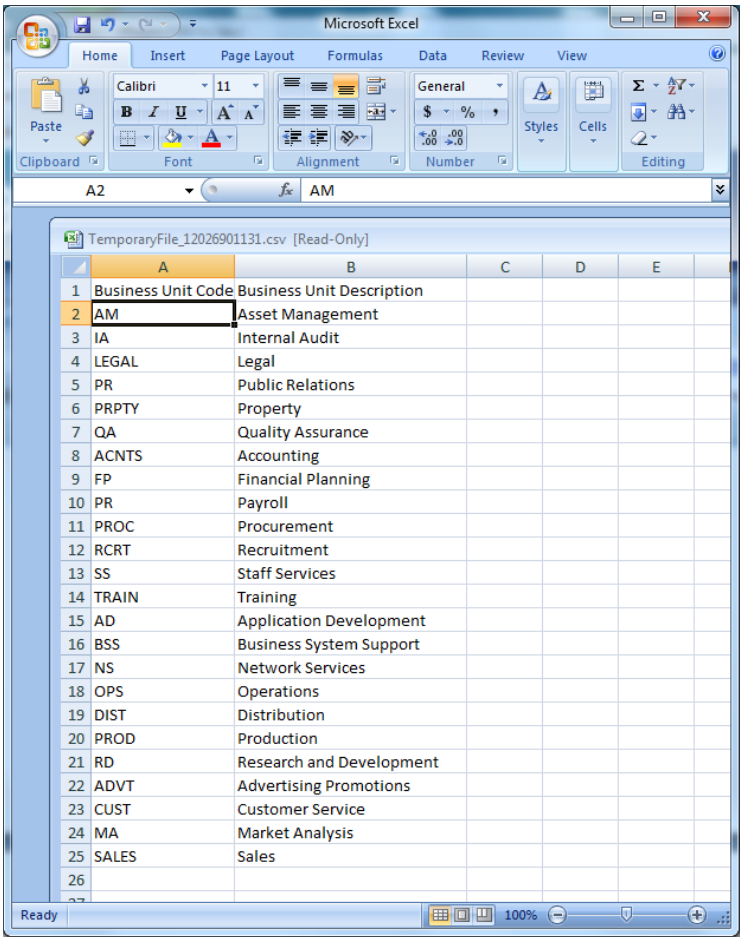
Do you want to open or save this file?

Name: TemporaryFile_12026901131.csv
Type: Microsoft Office Excel 97-2003 Worksheet, 626 b...
From: **vlfteam**

☑ Always ask before opening this type of file

While files from the Internet can be useful, some files can potentially harm your computer. If you do not trust the source, do not open or save this file. What's the risk?

And when they click *Open* button they should see:



A2 AM

TemporaryFile_12026901131.csv [Read-Only]

	A	B	C	D	E
1	Business Unit Code	Business Unit Description			
2	AM	Asset Management			
3	IA	Internal Audit			
4	LEGAL	Legal			
5	PR	Public Relations			
6	PRPTY	Property			
7	QA	Quality Assurance			
8	ACNTS	Accounting			
9	FP	Financial Planning			
10	PR	Payroll			
11	PROC	Procurement			
12	RCRT	Recruitment			
13	SS	Staff Services			
14	TRAIN	Training			
15	AD	Application Development			
16	BSS	Business System Support			
17	NS	Network Services			
18	OPS	Operations			
19	DIST	Distribution			
20	PROD	Production			
21	RD	Research and Development			
22	ADVT	Advertising Promotions			
23	CUST	Customer Service			
24	MA	Market Analysis			
25	SALES	Sales			
26					

From MS-Excel they can save the file to their hard drive, reformat the content, display graphs and pivot tables, print the content, etc, etc.

The ***Employee Address*** button processing is similar. It's processing is designed to demonstrate how quickly more than 900 employee records can be downloaded into a spreadsheet.

When the ***Employee Details*** is button is clicked an employee number is extracted from the command input area on the screen (eg: A002450). This is passed to the SQL command named *Example3* as a SQL variable in the normal manner. If no lines were output to the file a message is displayed indicating that employee specified could not be found.

Significant Points

- You don’t have to invoke TABLEMANAGER.convertDynamicTable() from a button. You can invoke it from anywhere. eg: From a hyper-link or when a drop entry is selected from an drop down containing 50 selectable spreadsheet reports.

- The definition of the SQL command for Example1 demonstrates use of the resultColumnCaptions property. This allows the captions to be defined for the output fields to be defined. The precedence is that a resultColumnCaption will be used if it exists, then a resultColumnName if it exists, and finally an automatically generated caption like Column_n or Field_n will be used.

- You can separate your spreadsheet SQL commands from your other SQL commands by putting them in another file. The file that contains the dynamic SQL definitions is specified as property USERENV.dynamicTablesFile - which defaults to "tables_dynamic.txt". You could could change this to TABLEMANAGER.convertDynamicTable(“Example1”,”spreadsheetTables.txt”), for example, where spreadsheetTables.txt contains a different set of dynamic table definitions. Changing the source file name like this can also be used to logically partition access to the SQL commands by user, department, group, etc.

- The ***Employee Details*** example demonstrates extracting information from the current screen and passing it to the SQL request. For example - if the current screen displayed a product, then the product number could be extracted and passed to an SQL command that listed the sales of the product for the last 12 months.

- The **Employee Details** example also demonstrates adding additional properties to the SQL request in addition to the usual SQL variables. The possibilities include:

Name	**Meaning**
orientation	H or V indicating whether the spreadsheet columns should be arranged horizontally or vertically. The default is H.
outprefix	Output file prefix. Default is TemporaryFile.
outsuffix	Output file suffix. Default is csv.
outfolder	Output file folder. Default is <axes root folder>/ts/
delimiter	String field delimiter. Default is “ (double quote).
seperator	Field separator Default is , (comma)
outputCCSID	Output file CCSID/Code page. Must be valid windows client CCSID such as 950 or 932 Default is 1208 (UTF-8) which is the best choice for most situations. Make sure that the SQL dynamic table is also defined correctly with the source CCSID for columns read.

➔ The result object returned by TABLEMANAGER.convertDynamicTable() contains these properties:

Name	**Meaning**
outputFilePath	The output path (folder) in which the result file was produced. This is an IFS folder name, which typically is not the same as a URL reference to the file.
outputFileName	The output file name.
outputLineCount	The number of lines written to the output file, including any column headings.
error	True/false. Indicates that an error was trapped.
errorDesc	A description of the trapped error, if available. This message si automatically displayed and traced so normally you do not need to do this.

➔ The temporary files are put into the specified server folder so they can be accessed over the Internet. They are not automatically deleted because it is impossible to exactly know when to delete them. Typically they are deleted by generic name (eg: TemporaryFile*.csv) as part of overnight batch processing logic, or in high volume situations, they may be quickly scanned every hour or so and all files that are older than one hour are deleted (say).

More about using Dynamic Tables with SQL

Here are some more examples and tips for using SQL with eXtensions:

SQL Example 1 – Using STRSQL to Test your commands first

Here's a basic SQL example

```
select the department fields (XHRDEPCDE and XHRDEPNME) from the department file (XHRDEPT).
```

Test it on your iSeries in this format, using ***STRSQL***:

```
SELECT XHRDEPCDE, XHRDEPNME FROM AXESDEMO/XHRDEPT
```

Here it is in the format when defined in an Axes Dynamic table - the slash is replaced with a dot:

```
SELECT XHRDEPCDE, XHRDEPNME FROM AXESDEMO.XHRDEPT
```

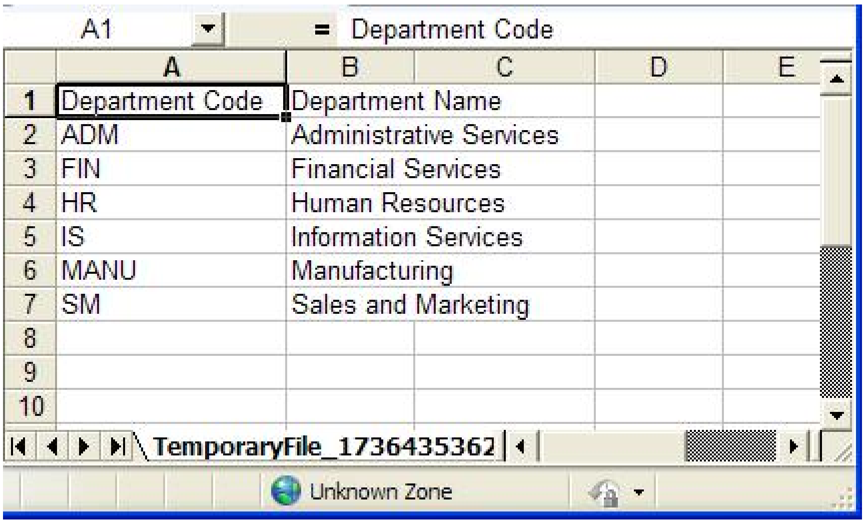
If we want to output the result to a MS excel spreadsheet, we first create a dynamic table containing the SQL statement, like this (remembering to remove the initial "select"):

```
    DefineObjectInstance {
      className           = "DynamicTable",
      name                = "SQLExample01",
      source              = "sql",
      selectSQLcommand    = " XHRDEPCDE, XHRDEPNME FROM AXESDEMO.XHRDEPT",
      resultColumnCaptions = { "Department Code", "Department Name" },
    };
```

Then we create an Axes button that makes use of the dynamic table, with onClick script as follows:

```
var result = TABLEMANAGER.convertDynamicTable("SQLExample01",USERENV.dynamicTablesFile);
if (result.error == false)
{
   var URL = document.location.protocol + "//" + document.location.host + "/ts/" +
result.outputFileName;
   window.open(URL,"_blank");
}
```

When the button is clicked, it should produce a simple list of departments in a spreadsheet, like this:



A1 = Department Code

	A	B	C	D	E
1	Department Code	Department Name			
2	ADM	Administrative Services			
3	FIN	Financial Services			
4	HR	Human Resources			
5	IS	Information Services			
6	MANU	Manufacturing			
7	SM	Sales and Marketing			
8					
9					
10					

SQL Example 2 – Joining Files

Suppose we wanted to do a basic JOIN between the Departments (XHRDEPT) and Employees (XHREMPTN) to show the department name for each employee.

When a field could come from more than one file, its file has to be specified, so in this code the field XHRDEPCDE becomes XHRDEPT.XHRDEPCDE

```
select XHRDEPT.XHRDEPCDE, XHRDEPT.XHRDEPNME, XHRSURNME, XHRGIVNME
from AXESDEMO/XHRDEPT, AXESDEMO/XHREMPTN
where XHRDEPT.XHRDEPCDE = XHREMPTN.XHRDEPCDE
ORDER BY XHRDEPT.XHRDEPCDE, XHRSURNME
```

or, to make it easier to refer to the files, we could give the files short names (de for Department, em for Employee)

```
select de.XHRDEPCDE, de.XHRDEPNME, XHRSURNME, XHRGIVNME
from AXESDEMO/XHRDEPT de, AXESDEMO/XHREMPTN em
where de.XHRDEPCDE = em.XHRDEPCDE
ORDER BY de.XHRDEPCDE, XHRSURNME
```

we create a dynamic table containing the SQL statement, (remembering to change the / to a dot, and removing the select) like this:

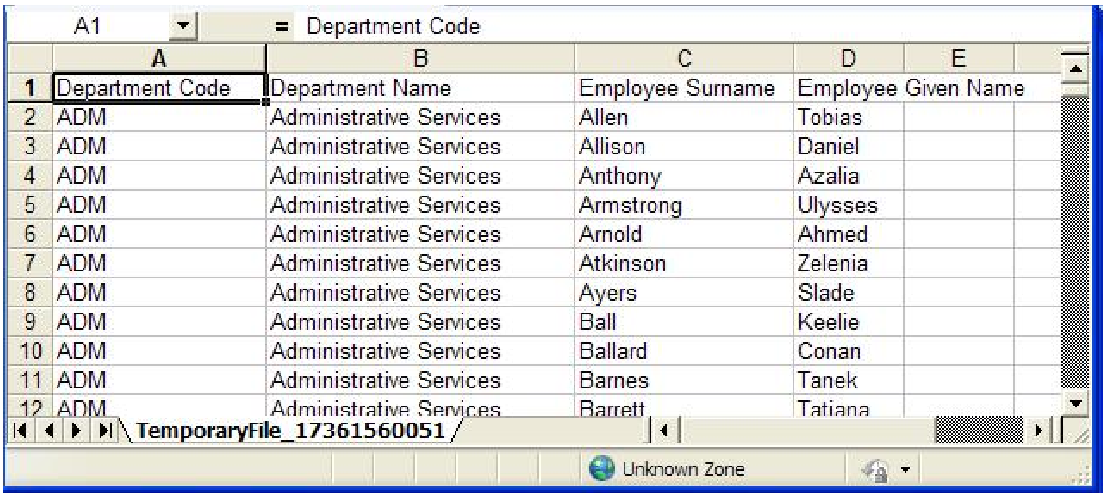
```
    DefineObjectInstance {
      className          = "DynamicTable",
      name               = "SQLExample02",
      source             = "sql",
      selectSQLcommand   =
" de.XHRDEPCDE, de.XHRDEPNME, XHRSURNME, XHRGIVNME from AXESDEMO.XHRDEPT de,
AXESDEMO.XHREMPTN em where de.XHRDEPCDE = em.XHRDEPCDE ORDER BY de.XHRDEPCDE, XHRSURNME ",
      resultColumnCaptions = { "Department Code", "Department Name", "Employee Surname",
"Employee Given Name" },
    };
```

To make it easier to read, we can use square brackets instead of double quotes around the SQL string. This allows us to use returns to break up the SQL statement into its sections.

```
    DefineObjectInstance {
      className          = "DynamicTable",
      name               = "SQLExample02",
      source             = "sql",
      selectSQLcommand   =
      [[
        de.XHRDEPCDE, de.XHRDEPNME, XHRSURNME, XHRGIVNME
        from AXESDEMO.XHRDEPT de, AXESDEMO.XHREMPTN em
        where de.XHRDEPCDE = em.XHRDEPCDE ORDER BY de.XHRDEPCDE, XHRSURNME
      ]],

      resultColumnCaptions = { "Department Code", "Department Name", "Employee  Surname",
"Employee Given Name" },

    };
```

Save your table, and modify the onClick routine of the Axes button to point to "SQLExample02"

If you now click the button you should see



A1 = Department Code

	A	B	C	D	E
1	Department Code	Department Name	Employee Surname	Employee Given Name	
2	ADM	Administrative Services	Allen	Tobias	
3	ADM	Administrative Services	Allison	Daniel	
4	ADM	Administrative Services	Anthony	Azalia	
5	ADM	Administrative Services	Armstrong	Ulysses	
6	ADM	Administrative Services	Arnold	Ahmed	
7	ADM	Administrative Services	Atkinson	Zelenia	
8	ADM	Administrative Services	Ayers	Slade	
9	ADM	Administrative Services	Ball	Keelie	
10	ADM	Administrative Services	Ballard	Conan	
11	ADM	Administrative Services	Barnes	Tanek	
12	ADM	Administrative Services	Barrett	Tatiana	

Note: If you want to do a join where records from the first file always appear, use this syntax:

```
select XHRDEPT.XHRDEPCDE, XHRDEPT.XHRDEPNME, XHRSURNME, XHRGIVNME
from AXESDEMO/XHRDEPT
LEFT OUTER JOIN AXESDEMO/XHREMPTN
ON XHRDEPT.XHRDEPCDE = XHREMPTN.XHRDEPCDE
ORDER BY XHRDEPT.XHRDEPCDE, XHRSURNME
```

Note: If you want to do a join that returns only the records in the first file that don't have a match in the second file, use this syntax

```
select XHRDEPT.XHRDEPCDE, XHRDEPT.XHRDEPNME, XHRSURNME, XHRGIVNME
from AXESDEMO/XHRDEPT
EXCEPTION JOIN AXESDEMO/XHREMPTN
ON XHRDEPT.XHRDEPCDE = XHREMPTN.XHRDEPCDE
ORDER BY XHRDEPT.XHRDEPCDE, XHRSURNME
```

SQL Example 3 – Summarizing Data

Suppose that instead of showing one line for every employee, we wanted a summary result, with one line for each department. We use the GROUP BY parameter

```
select de.XHRDEPCDE
from AXESDEMO/XHRDEPT de, AXESDEMO/XHREMPTN em
where de.XHRDEPCDE = em.XHRDEPCDE
GROUP BY de.XHRDEPCDE
```

When we work with a group of records there are several useful things we can do with each group of detail records. Examples are:

```
COUNT
SUM
AVG
MAX
MIN
```

So, in our summary query we can show the number of employees, the average salary, the maximum salary, and the total salary for each department, as follows

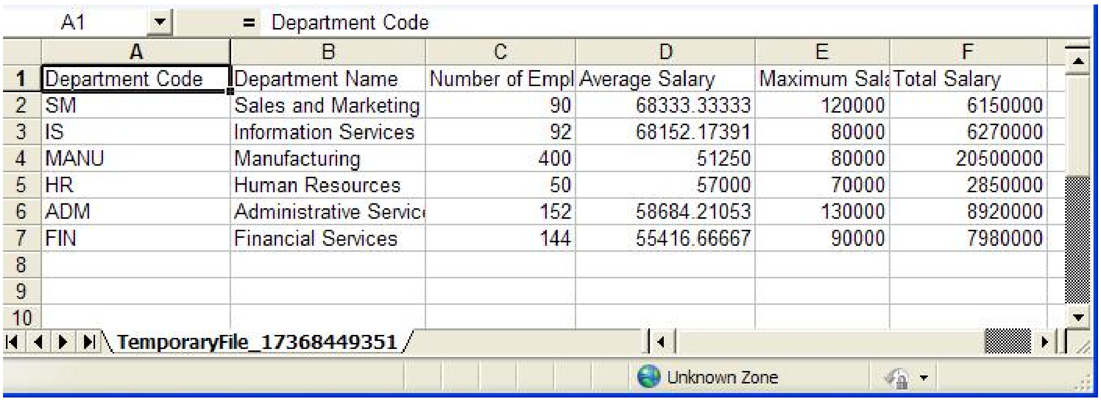
```
select de.XHRDEPCDE, MAX(de.XHRDEPNME), COUNT(*), AVG(XHRSALARY),
MAX(XHRSALARY), SUM(XHRSALARY)
from AXESDEMO/XHRDEPT de, AXESDEMO/XHREMPTN em
where de.XHRDEPCDE = em.XHRDEPCDE
GROUP BY de.XHRDEPCDE
```

we create a dynamic table containing the SQL statement, like this:

```
    DefineObjectInstance {
      className           = "DynamicTable",
      name                = "SQLExample03",
      source              = "sql",
      selectSQLcommand    =
      [[ de.XHRDEPCDE, MAX(XHRDEPNME), COUNT(*), AVG(XHRSALARY), MAX(XHRSALARY),
SUM(XHRSALARY)
      FROM AXESDEMO.XHRDEPT de, AXESDEMO.XHREMPTN em
      where de.XHRDEPCDE = em.XHRDEPCDE
      GROUP BY de.XHRDEPCDE
      ]],
      resultColumnCaptions = { "Department Code", "Department Name", "Number of Employees",
"Average Salary", "Maximum Salary", "Total Salary" },
    };
```

Save your table, and modify the onClick routine of the Axes button to point to "SQLExample03"

If you now click the button you should see:



A1 = Department Code

	A	B	C	D	E	F
1	Department Code	Department Name	Number of Empl	Average Salary	Maximum Sala	Total Salary
2	SM	Sales and Marketing	90	68333.33333	120000	6150000
3	IS	Information Services	92	68152.17391	80000	6270000
4	MANU	Manufacturing	400	51250	80000	20500000
5	HR	Human Resources	50	57000	70000	2850000
6	ADM	Administrative Servic	152	58684.21053	130000	8920000
7	FIN	Financial Services	144	55416.66667	90000	7980000
8						
9						
10						

SQL Example 4 – Controlling Data Extracted

When you want to show only some of the records on the file, there are two ways to select them.

Method 1) When you are selecting records based on field values for individual records, you add extra AND commands to the WHERE

using comparison operators like > < = != <= >= or BETWEEN, LIKE, IN

For example - Get all the employees with a salary outside 30,000 to 50,000, who are not in the SALES or LEGAL departments, with surnames beginning with S.

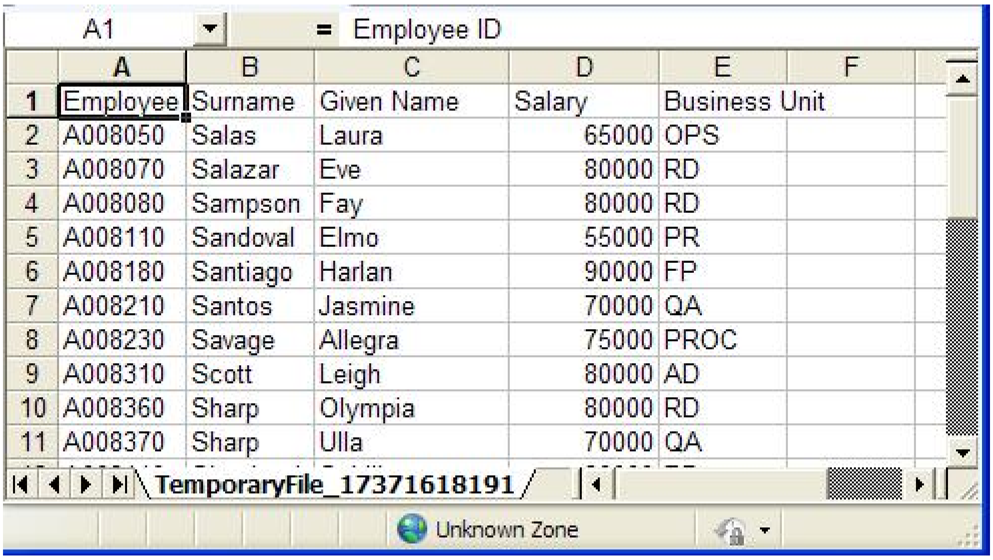
```
SELECT XHREMPID, XHRSURNME, XHRGIVNME, XHRSALARY, XHRBUABRV
FROM AXESDEMO/XHREMPTN
WHERE XHRSALARY NOT BETWEEN 30000 AND 50000
AND XHRBUABRV NOT IN ('SALES', 'LEGAL')
AND XHRSURNME LIKE 'S%'
```

we create a dynamic table containing the SQL statement, like this:

```
    DefineObjectInstance {
      className          = "DynamicTable",
      name               = "SQLExample04",
      source             = "sql",
      selectSQLcommand   =
      [[
      XHREMPID, XHRSURNME, XHRGIVNME, XHRSALARY, XHRBUABRV
      FROM AXESDEMO.XHREMPTN
      WHERE XHRSALARY NOT BETWEEN 30000 AND 50000
      AND XHRBUABRV NOT IN ('SALES', 'LEGAL')
      AND XHRSURNME LIKE 'S%'
      ]],
      resultColumnCaptions = { "Employee ID", "Surname", "Given Name", "Salary", "Business
Unit" },
    };
```

Save your table, and modify the onClick routine of the Axes button to point to "SQLExample04"

If you now click the button you should see something like:



	A	B	C	D	E	F
1	Employee	Surname	Given Name	Salary	Business Unit	
2	A008050	Salas	Laura	65000	OPS	
3	A008070	Salazar	Eve	80000	RD	
4	A008080	Sampson	Fay	80000	RD	
5	A008110	Sandoval	Elmo	55000	PR	
6	A008180	Santiago	Harlan	90000	FP	
7	A008210	Santos	Jasmine	70000	QA	
8	A008230	Savage	Allegra	75000	PROC	
9	A008310	Scott	Leigh	80000	AD	
10	A008360	Sharp	Olympia	80000	RD	
11	A008370	Sharp	Ulla	70000	QA	

Note:
Method 2) If you want to select values based solely on summary values, you use a different keyword - HAVING

Suppose you wanted to select all the departments with an average above a value

```
SELECT XHRDEPCDE, AVG(XHRSALARY)
FROM AXESDEMO/XHREMPTN
GROUP BY XHRDEPCDE
HAVING AVG(XHRSALARY) > 60000
ORDER BY XHRDEPCDE
```

SQL Example 5 – Nested Queries

Suppose you want to do one query and then compare the result of that query with another file. You can nest queries by making the query one of the FROM values:

For example, list all the employees with 50% higher than the average for their department.

One query works out what the average salary is for all the departments.
The employees file is joined with the result of that query, to determine which employees have 50% higher than the average salaries for the department.

There are two FROM entries in this query; one is the employees file (named em), one is a nested query (named mysummary)

```
select em.XHREMPID, em.XHRSURNME, em.XHRGIVNME, em.XHRSALARY,
mysummary.empcount, mysummary.avgsal, em.XHRDEPCDE, em.XHRSTATE
FROM
AXESDEMO/XHREMPTN em,
(
select XHRDEPCDE, COUNT(*) empcount, AVG(XHRSALARY) avgsal
from AXESDEMO/XHREMPTN
GROUP BY XHRDEPCDE
ORDER BY XHRDEPCDE
) mysummary
where
mysummary.XHRDEPCDE = em.XHRDEPCDE AND
em.XHRSALARY > (mysummary.avgsal * 1.5)
```

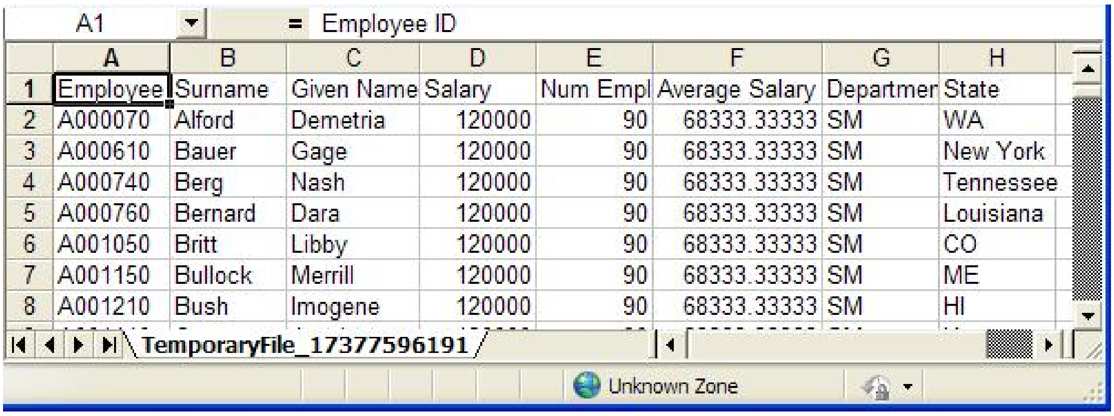
Note how result columns in the summary query have been assigned names (e.g. avgsal) and then referred to in the WHERE condition.

we create a dynamic table containing the SQL statement, like this:

```
    DefineObjectInstance {
      className          = "DynamicTable",
      name               = "SQLExample05",
      source             = "sql",
      selectSQLcommand   =
      [[
          em.XHREMPID, em.XHRSURNME, em.XHRGIVNME, em.XHRSALARY,
          mysummary.empcount, mysummary.avgsal, em.XHRDEPCDE, em.XHRSTATE
          FROM
          AXESDEMO.XHREMPTN em,
          (
             select XHRDEPCDE, COUNT(*) empcount, AVG(XHRSALARY) avgsal
             from AXESDEMO.XHREMPTN
             GROUP BY XHRDEPCDE
             ORDER BY XHRDEPCDE
          ) mysummary
          where
          mysummary.XHRDEPCDE = em.XHRDEPCDE AND
          em.XHRSALARY > (mysummary.avgsal * 1.5)
      ]],
      resultColumnCaptions = { "Employee ID", "Surname", "Given Name", "Salary", "Num
Employees", "Average Salary", "Department", "State"  },
    };
```

Save your table, and modify the onClick routine of the Axes button to point to "SQLExample05"

If you now click the button you should see something like:



A1 = Employee ID

	A	B	C	D	E	F	G	H
1	Employee	Surname	Given Name	Salary	Num Empl	Average Salary	Departmer	State
2	A000070	Alford	Demetria	120000	90	68333.33333	SM	WA
3	A000610	Bauer	Gage	120000	90	68333.33333	SM	New York
4	A000740	Berg	Nash	120000	90	68333.33333	SM	Tennessee
5	A000760	Bernard	Dara	120000	90	68333.33333	SM	Louisiana
6	A001050	Britt	Libby	120000	90	68333.33333	SM	CO
7	A001150	Bullock	Merrill	120000	90	68333.33333	SM	ME
8	A001210	Bush	Imogene	120000	90	68333.33333	SM	HI

SQL Example 6 – Data Manipulation and Date Handling

Manipulation of data in fields, and Dates.

You may need to rearrange data within fields, in order to do comparisons.

The following example (converting a numeric YYYYMMDD date into an ISO date) demonstrates
a) number to alphanumeric conversion (CHAR)
b) substring (SUBSTR)
c) concatenation (||)

```
SELECT XHREMPID, YYYYMMDD,
Date(
  Days( SUBSTR(CHAR(YYYYMMDD),1,4)||'-'|| SUBSTR(CHAR(YYYYMMDD),5,2)|| '-'||
SUBSTR(CHAR(YYYYMMDD),7,2)  )
     )
)
FROM EMPLOYEEFILE
```

It also demonstrates some of the DATE functions that are available when the date is in ISO (YYYY-MM-DD) format
Once a date is in ISO format, you can convert it to a date value, and do date arithmetic, like this:

```
SELECT XHREMPID, XHRSTDTE,
DATE(XHRSTDTE) + 6 MONTHS
FROM AXESDEMO/XHREMPTN
```

Other examples of useful functions are

```
DATE
DAY
DAYNAME
CURDATE
DAYOFWEEK
DAYOFMONTH
DAYOFYEAR
DAYS
MONTH
MONTHNAME
MONTHSBETWEEN
WEEK
YEAR
```

You can add durations of

```
YEARS
MONTHS
DAYS
```

to a date.

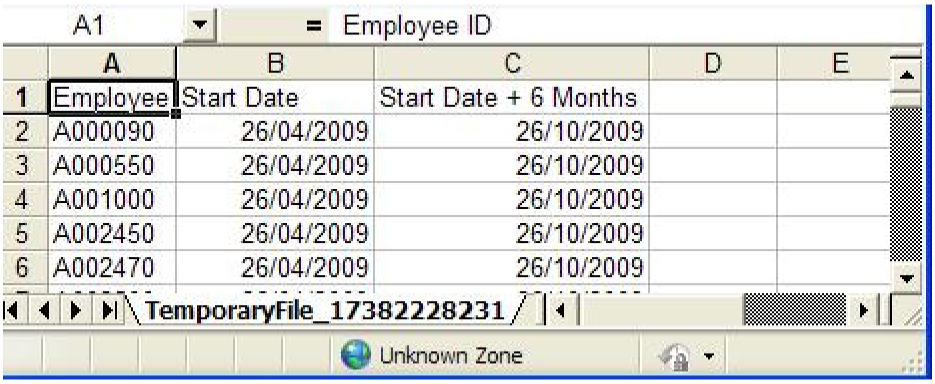
See SQL reference for more details about DateTime arithmetic

we create a dynamic table containing the SQL statement, like this:

```
    DefineObjectInstance {
      className          = "DynamicTable",
      name               = "SQLExample06",
      source             = "sql",
      selectSQLcommand   =
      [[
          XHREMPID, XHRSTDTE,
         DATE(XHRSTDTE) + 6 MONTHS
         FROM AXESDEMO.XHREMPTN
      ]],

      resultColumnCaptions = { "Employee ID", "Start Date", "Start Date + 6 Months"  },
    };
```

Save your table, and modify the onClick routine of the Axes button to point to "SQLExample06"

If you now click the button you should see something like:



A1 = Employee ID

	A	B	C	D	E
1	Employee	Start Date	Start Date + 6 Months		
2	A000090	26/04/2009	26/10/2009		
3	A000550	26/04/2009	26/10/2009		
4	A001000	26/04/2009	26/10/2009		
5	A002450	26/04/2009	26/10/2009		
6	A002470	26/04/2009	26/10/2009		

References:
http://www.w3schools.com/Sql/default.asp
(Good examples, but the syntax is sometimes wrong for iSeries)

http://publib.boulder.ibm.com/infocenter/iseries/v6r1m0/index.jsp
(search for SQL)
Database SQL Programming
DB2 for i SQL Reference

eXtensions Tutorial 7 - Best Practices

Develop Standards Early

You should develop a naming standard for screens, screen fields and scripting variables before commencing a screen modernization project. For example:

Type of Object	**Naming Standard**
Screen	**SN**XXXXXXXXXXX
Alpha Field in Screen	**FA**XXXXXXXXXXX
Numeric Field in Screen	**FN**XXXXXXXXXXX
Alpha Field in Subfile	**SA**XXXXXXXXXXX
Numeric Field in Subfile	**SN**XXXXXXXXXXX
Special fields repeated on many screens such as error message fields, page/up down "+", markers, titles, etc.	**FS**XXXXXXXXXXX
Script string	**s**XXXXXXXXXXX
Script integer / whole number	**i**XXXXXXXXXXX
Script floating point / number with decimals	**n**XXXXXXXXXXX
Script boolean	**b**XXXXXXXXXXX
Script object	**o**XXXXXXXXXXX
Script object referring to HTML element	**h**XXXXXXXXXXX
Script function	**f**XXXXXXXXXXX
Arrays	Prefix with additional **a**
Private member / object	Prefix with additional _ (underscore)

Where XXXXXXXXXXX is a meaningful name with precise case (e.g.: ProductNumber, not Productnumber) using letters from the English alphabet and 0 -> 9 only.

Avoid specifying fonts and colors for individual elements

Avoid specifying fonts and colors for individual screen elements. Create application styling items based on the field's role in the application and use it instead. Refer to Tutorial 3 - Advanced Screen Enhancement and Tutorial 10 - 5250 Screen Styling for more details and examples.

Treat screen Modernization as a Project

It's advisable to treat the modernization and customization of a 5250 applications as real IT project, rather than ad-hoc or part time activities. This means that all the normal IT project activities like end user consultation, detailed design, specifications, test planning, defined goals, etc should be brought into play. Remember the old saying "If you don't have a plan then you are planning to fail".

Define your modernization "Value Proposition"

Modernizing the appearance of a 5250 screen is one thing, but adding real value to a 5250 application is another. If you focus on appearance only, the early enthusiasm and support you may receive from your customers and users could fade as they realize that nothing has been done to make their working lives easier, better, faster or smarter (i.e.: more modern). It's worth spending some time at the start of a project to define *exactly* what value you will add to the existing 5250 application.

Follow the 80/20 rule

In most commercial 5250 applications 20% of the application accounts for 80% of the use. Focus on modernizing and customizing the 20% that is most used, not the 80% that is less used. You are probably better to modernize the Order Entry screens or the Insurance Policy Master Update screens rather than the Country Code Maintenance screens.

Use an Incremental Delivery Plan

If you have a large 5250 application consider using a staged and incremental delivery plan of gradual improvement rather than trying to deliver everything as a single "big bang" project. Typically this encourages earlier feedback from your users and customers, and a more gradual learning and experience curve for your developers.

Use static DBMS code tables in XML documents

Relatively static DBMS content used for field codes and decodes can be (re)published when it changes as XML documents on the IFS. These documents can then be used as static table input to fill drop downs, service scripts, etc. This may be more efficient than using SQL to access the DBMS data.

Always assess screen customizations as a user

You should always test and assess your screen customizations and eXtension scripting signed on as a user. While you can conduct tests logged on as an aXes developer, the tests are not valid because they do not reflect a real execution environment.

Use Two Discrete Cycles: Identify and Customize

In eXtension projects it is best if two clearly separated cycles are used.

- Identify the screens that you intend to enhance and name the fields on them. Verify that all variations of the screen(s) involved are correctly identified.

- Then, individually customize and test the screens.

Mixing the identification and customization cycles together is not a recommended approach.

Document/Communicate USERENV content

The USERENV object is a useful way to reuse and share common properties and logic. If you spend time documenting and communicating the details to all the project team members, this encourages re-use and helps to enforce standards.

Versioning aXes projects

This section outlines some general notes about versioning any product, not just aXes. It does not represent anything that is officially supported or sanctioned by LANSA or aXes.

The advice is folder based. Disk space is cheap and getting cheaper. People's time is expensive and getting more expensive. You can be too clever with elaborate folder structures designed to save space and end up confusing people and causing unnecessary mistakes - KISS is best.

A simple way and low cost way to version aXes projects

Use project folders named like SSS_PPPPP_NNNN_FFF where

SSS = System Type – one of DEV, TST or PRD meaning development, test, production,
PPPPP = Project Name
NNNN = Project Build of Version Number
FFF = Optional hotfix or branch number

Development -> Test -> Production flow

When a development version is completed it is promoted to testing.
For example:

DEV_ProjectName_0001 is promoted to TST_ProjectName_0001
This flow is repeated until version 0001 is considered to be tested and ready.

When testing is complete it is promoted to production: TST_ProjectName_0001 is promoted to PRD_ProjectName_0001. Users directly (or indirectly via a launch page) use PRD_ProjectName_0001 on their start up URLs.

Starting to work on a New Version

When work and testing completes on a project version a new version is started.
For example:
DEV_ProjectName_0001 is procedurally locked and possibly archived.
DEV_ProjectName_0001 is copied to DEV_ProjectName_0002
Developers now all work on DEV_ProjectName_0002
When development completes, repeat the preceding development -> test -> production cycle and then move on to DEV_ProjectName_0003.

Back version Maintenance

The archived and procedurally locked back version is used to perform the change.
For example – a user reports an issue on version 0005 - but people are already testing version 0006 and the developers are already working on version 0007.
DEV_ProjectName_0005 is restored (if required) and procedurally unlocked.
DEV_ProjectName_0005 is corrected.
DEV_ProjectName_0005 is promoted to TST_ProjectName_0005 for final test.
TST_ProjectName_0005 is promoted to PRD_ProjectName_0005 when ready for production
DEV_ProjectName_0005 is procedurally locked and re-archived.
The correction may need to "propagate to the tip". This means that product version 006 (in testing) and 0007 (under development) may also need to be corrected because they require the same correction. This is a very common IT back version correction issue and has nothing at all to do with aXes per se.
The DEV_ProjectName_0007 developers continued to work through all this with no impact.

Hotfixing

A variation that tracks every fix made to every version may be:
DEV_ProjectName_0005 is promoted to DEV_ProjectName_0005_001 (fix number 001)
DEV_ProjectName_0005_001 is promoted to TST_ProjectName_0005 for final test.
TST_ProjectName_0005 is promoted to PRD_ProjectName_0005 when ready for production
This variation maintains version 0005 in its original "as shipped" state and each cumulative fix is also maintained as a complete working unit.

Branching

Use the FFF prefix (or some other variation) in the folder name to also denote a branch as well as a fix.
Branching is only usually required when a major feature or development needs to be back fitted into a preceding version for a special case or customer.
For example - branching may be required if a customer of your application refuses to upgrade to your latest version - but demands that a new version feature be made to work in their old version. This means you need to put a branch into the older product version.

Going live with a new version

Let's say you are using PRD_ProjectName_0005 and you are ready to go live with version 0006.
You can run versions 0005 and 0006 side by side for a while, gradually introducing the new version by selectively changing user desktop URLs to use PRD_ProjectName_0006 instead of PRD_ProjectName_0005.
To avoid changing desktop URLs, you can give users a "launch page" named, for example, ProjectName.htm.
When ready to move from version 0005 to version 0006, change the launch URL in ProjectName.htm.

Tip

Put the version/build and fix numbers into the USRENV object and make them visible on some screen when a button is clicked. This allows you to determine the exact build/version and fix level of your application that they are using.

Supporting Multiple Customer Bespoke Versions

Use alternative name components. For example DEV_CustomerName_ProjectName_0001.

Folder Storage

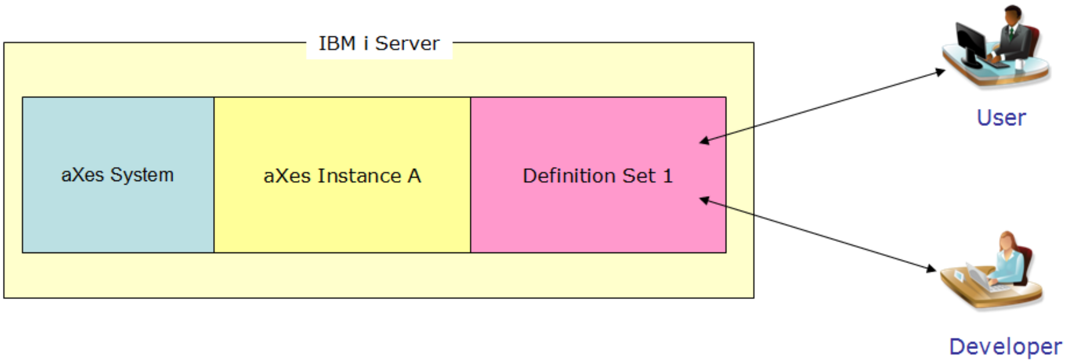
The folder names used here are logical more than physical. Their actual content can be copied to cheap mass storage space on a Windows servers and/or versioned and archived using many different PC systems – for example, Vault.

Establish the Deployment Model

aXes only allows one *system* (at the same version level) to be installed onto an IBM Server. An aXes system can run multiple separate *instances*. An instance can contain multiple *definition sets* - each containing a discrete and independent application.

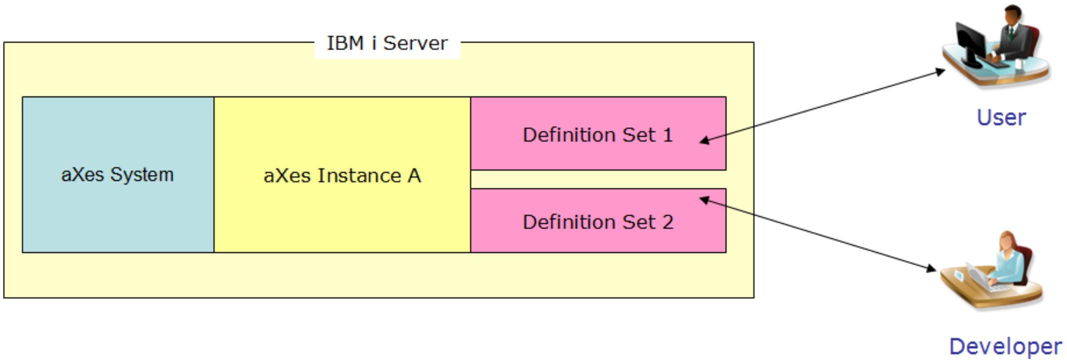
The three levels of separation (system, instance and definition set) allow for many different deployment configuration models. The most typical of these are:

HIGH RISK DEPLOYMENT MODEL



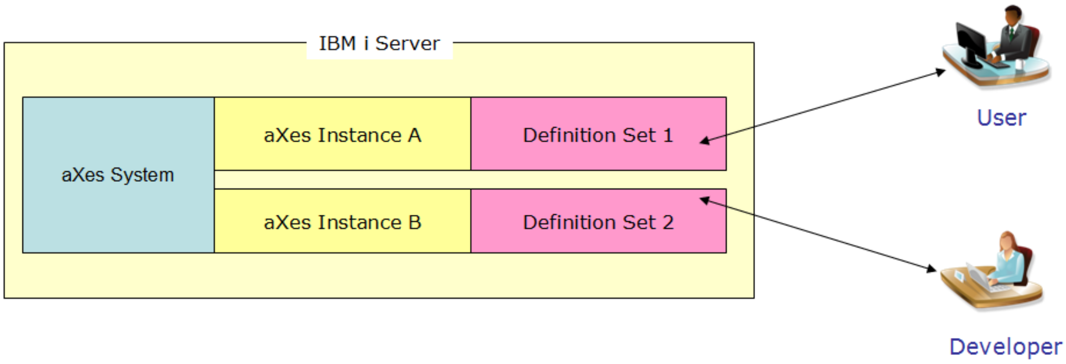
Here the users and developers all work within the same definition set, within the same instance, within the same system. This is a high risk configuration because the activities of the developers will almost certainly interfere with those of the users.

MODERATE RISK DEPLOYMENT MODEL



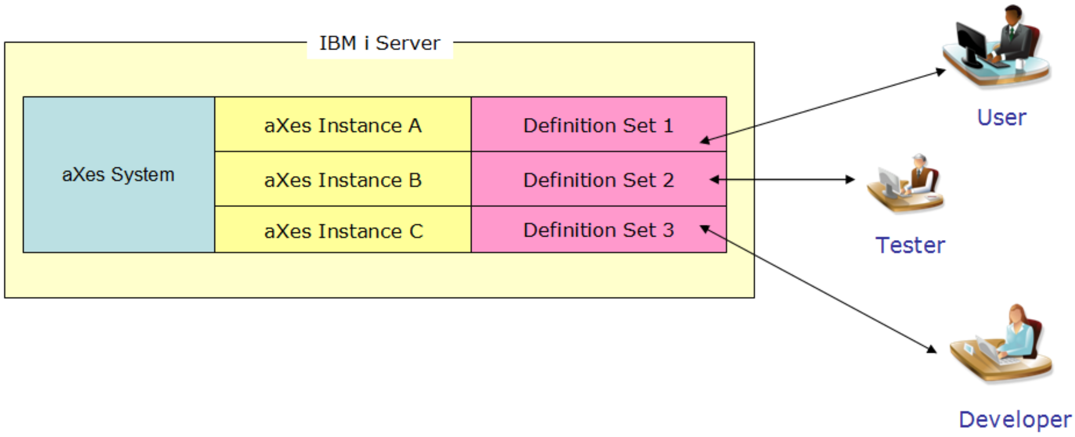
Here the developers and users work from separate definition sets, even though they are still working within the same instance and system. This will reduce interference, except for occasions where developers may need to stop and restart their aXes instance.

TOLERABLE RISK DEPLOYMENT MODEL



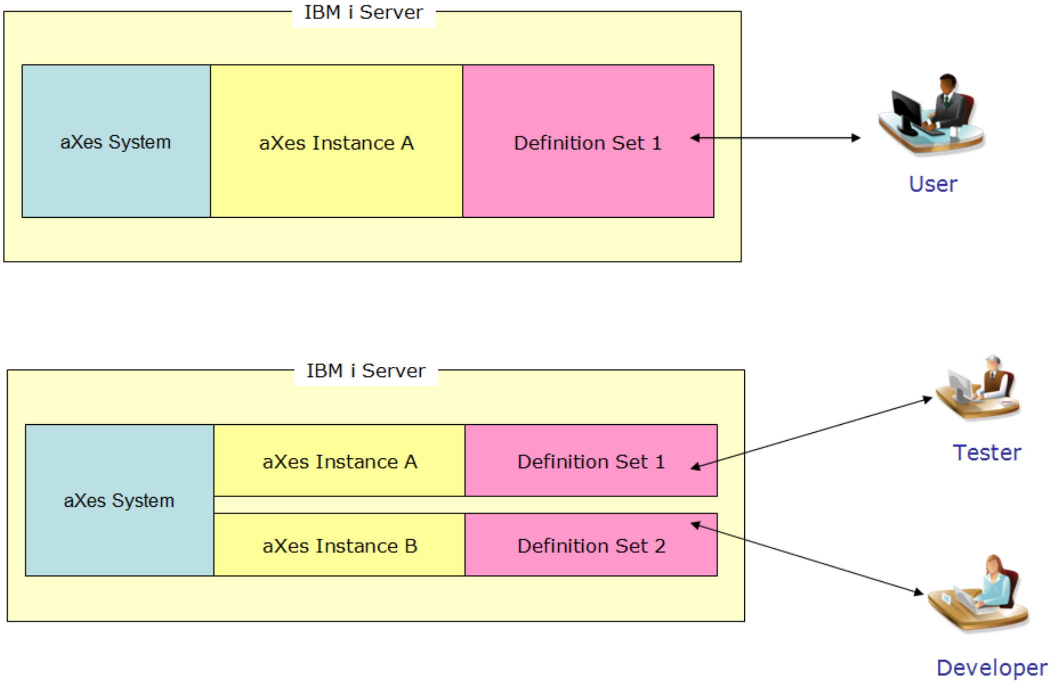
Here the users and developers work in different definition sets and different instances. The major risk in this configuration is of deploying something to a production user without first testing it in a dedicated tested environment.

LOW RISK DEPLOYMENT MODEL



The deployment risk in the preceding model can be reduced by introducing a dedicated test instance and definition set. Production material is bundled and deployed to the test environment for suite testing before it is finally deployed to the production environment.

OPTIMAL DEPLOYMENT MODEL



This type of deployment model minimizes all risks, but requires two physical or logical (LPAR) IBM i servers.

If You Use aXes-eXtensions with aXes-Cloud

aXes-Cloud comes with all of the features of aXes including aXes-eXtensions.

aXes-Cloud uses Telnet to connect with applications on remote servers and therefore aXes requires no installation of objects on remote servers that support the applications.

Screen identification characteristics differ when used on the local server and when used on remote servers via Telnet. Therefore developers must identify screens as they will appear in the production environment on the remote server, in other words in the same context as users will use the application.

This means developers must sign on to the cloud gateway server, connect and sign on to a remote server, start an application and then work on extending the screens. When users use the applications, aXes-Cloud applies the enhancements built with aXes-eXtensions in real time from the cloud gateway server.

The following table summarizes the policies for using aXes-eXtensions with aXes-Cloud:

The aXes-eXtensions projects must reside on the cloud gateway server.
Developers work on the projects on the cloud gateway server.
Developers must be connected to an application on a remote server when working on the projects.

Warning	Do not extend screens locally and assume they will operate as you expect when the applications run on remote servers.

Documentation Library

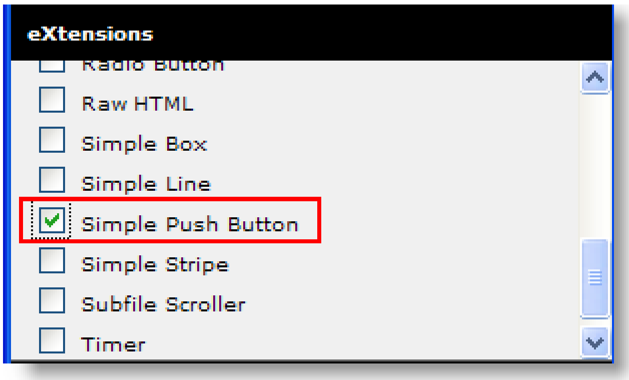
eXtensions Tutorial 8 - Creating your own eXtensions

Getting Started

eXtensions are JavaScript objects with properties, methods and events. They are defined in a JavaScript file.

The name of the JavaScript file that defines the extension must start with **Extension_** followed by your chosen extension name.

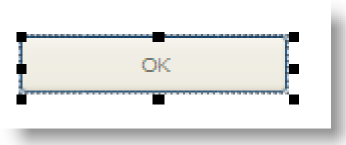
In this tutorial we will create a simple push button extension:



eXtensions

□ Radio Button
□ Raw HTML
□ Simple Box
□ Simple Line
☑ Simple Push Button
□ Simple Stripe
□ Subfile Scroller
□ Timer

Which will look like this:



Clicking the button will send the Enter key by default.

To begin creating this extension, start Notepad and save the file with the name *Extension_SimplePushButton.js*

Remarks:

Do not use any combination of the prefix *ax* (*Ax, AX, etc*) they are reserved for **aXes only** developed extensions.

When adding eXtensions to 5250 screens for subssequent use in an aXes-Cloud environment ensure you are executing them in the same way that your end users will. See If You Use aXes-eXtensions with aXes-Cloud in the Best Practices tutorial for more details.

A basic eXtension skeleton

Depending on your JavaScript skills, when writing the first extension you may choose an existing extension as the starting point. Otherwise, you can use this basic eXtension skeleton:

```
function DEFINE_EXTENSION_XYZ()
{
        var definition =
        {

                properties:
                {

                },

                init: function(element, elementContainer)
                {
                   this._element = element ;
                   this._elementContainer = elementContainer ;




                   if (AXES.isTS2Engine) element.addListener("AddedToDOM",
                this._handleAddedToDOM, this);
                },

                render: function(element, elementContainer)
                {




                },

                destroy: function(element, elementContainer)
                {
                   element.removeListener("AddedToDOM", this._handleAddedToDOM);
                   delete this._element;
                   delete this._elementContainer;
                },


                _handleAddedToDOM: function(event)
                {

                },

      <other event handler(s)>

        };

        AXES.Extensions.add(definition);
}

DEFINE_EXTENSION_XYZ()
```

Because we know the name of the extension to create we can replace XYZ with our chosen name of *SimplePushButton*.

Copy this skeleton into your *Extension_SimplePushButton.js* file:

```
function DEFINE_EXTENSION_SimplePushButton()
{
        var definition =
        {
                name: "SimplePushButton",

                properties:
                {

                },

                init: function(element, elementContainer)
                {
                   this._element = element ;
                   this._elementContainer = elementContainer ;





                   if (AXES.isTS2Engine) element.addListener("AddedToDOM",
                this._handleAddedToDOM, this);
                },

                render: function(element, elementContainer)
                {


                },

                destroy: function(element, elementContainer)
                {
                   element.removeListener("AddedToDOM", this._handleAddedToDOM);
                   delete this._element;
                   delete this._elementContainer;

                },

                _handleAddedToDOM: function(event)
                {

                },

              <other event handler(s)>


        };

        AXES.Extensions.add(definition);
}

DEFINE_EXTENSION_SimplePushButton ()
```

The aXes defined properties section

The beginning of the eXtension object definition specifies the properties that identify it in aXes:

```
function DEFINE_EXTENSION_SimplePushButton()
{
        var definition =
        {
```

In this section you specify these predefined properties:

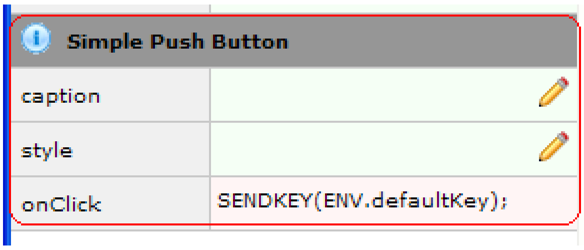
- *name*: the name of this extension
- *type:* whether the extension applies to an *element* or to a *screen.*
- *display:* the description that is visible when you select the extension
- *group*: determines whether when selecting this extension all other extensions in the same group for the same element are deselected.
- *subtype*: determines whether the extension is used **only** for New elements or for both new and existing fields on the screen.

In your extension file, copy and paste these properties and values:

```
        var definition =
        {
                name: "SimplePushButton",
                group: "vis",
                type: "element",
                display: "Simple Push Button",
                subType: "new",
```

The User-defined properties section

In the next section you specify the properties and events that are under your control, the ones that are visible to the developer in aXes Designer:



Simple Push Button	
caption	
style	
onClick	SENDKEY(ENV.defaultKey);

We are going to add three properties as in the above picture to the button extensions:

- *caption*: the button caption
- *style*: to allow the user to apply a style to the button
- *onClick*: the onClick event

Properties have a *type*. They can also have a *default value* and a *description*.

```
   properties:
   {
     caption: { type: "String", defaultStatic:"OK", description: "Text to appear on the
button." },
     style:   { type: "Style", description: " Style to apply to the extension." },
     onClick: { type: "Event", defaultDynamic: "SENDKEY(ENV.defaultKey);", description: " The
button's onclick event." }
   },
```

In your extension copy and paste the above three properties. Your extension should now look like this:

```
function DEFINE_EXTENSION_SimplePushButton()
{
    var definition =
    {
       name: "SimplePushButton",
       group: "vis",
       type: "element",
       display: "Simple Push Button",
       subType: "new",

       properties:
       {
        caption: { type: "String", defaultStatic:"OK", description: "Text to appear   on the
button." },
        style:   { type: "Style", description: "Style to apply to the extension." },
        onClick: { type: "Event", defaultDynamic: "SENDKEY(ENV.defaultKey);", description: "
The button's onclick event." }
       },
```

The program section

You write the executable code in the methods in the next section.

An extension has at least three methods, *init*, *render* and *destroy* and event handler *_handleAddedToDOM* and any other event handlers that have been defined.

Initialization

The *init* method executes at design time and runtime:

```
init: function(element, elementContainer)
{
    this._element = element ;
    this._elementContainer = elementContainer ;


    if (AXES.isTS2Engine) element.addListener("AddedToDOM", this._handleAddedToDOM, this);
},
```

The init method is where you usually create the HTML element that your extension will visualize itself as.

It receives two parameters:

element – a reference to the aXes element object
elementContainer – usually the element's parent DIV element

In this example you use the standard createElement dhtml method to create a button:

```
var btn = document.createElement("button");
```

Get the value of the *caption* property:

```
    var btnCaption = this.getPropertyValue("caption");
```

Set the dhtml innerHTML button property to the value of the *caption*:

```
    btn.innerHTML = btnCaption;
```

Add a user-defined property to the button reference. Here we set the default function key we want to send.

```
    btn.sendKey   = "Enter";
```

Add the onclick event to the html button. Because the *init* logic is executed at design time but we don't want the onclick to fire when we are editing the screen, we make sure to attach the event only when not in design mode:

```
    if (!this.isDesignMode())
    {
      var self = this;
      btn.onclick = function() { self._onClick(btn); };
    }
```

Make the button part of the tabbing order:

```
    this.applyTabIndex(btn);
```

Append the html button as a child of the container:

```
    elementContainer.appendChild(btn);
```

Save a reference to the html button for easier access in other routines:

```
    this.buttonElem = btn;
```

The init method should look like this:

```
init: function(element, elementContainer)
{
   this._element = element ;
   this._elementContainer = elementContainer ;

   var btnCaption = this.getPropertyValue("caption");

   var btn = document.createElement("button");
   btn.innerHTML = btnCaption;
   btn.sendKey   = "Enter";

   if (!this.isDesignMode())
   {
      var self = this;
      btn.onclick = function() { self._onClick(btn); };
   }

   this.applyTabIndex(btn);

   elementContainer.appendChild(btn);

   this.buttonElem = btn;


   if (AXES.isTS2Engine) element.addListener("AddedToDOM", this._handleAddedToDOM,     this);
},
```

Rendering

The *render* method executes at design time and runtime, after the *init* method. In this example we are going to use it to size the button.

When customizing a screen, you size the button on the screen by stretching it. When the screen is saved the size needs to be recorded as a property of the element:

```
    var size = element.getSize();
    button.style.width = (size.width).toString() + "px";
    button.style.height = (size.height).toString() + "px";
```

When sizing controls, both TS1 and TS2 modes have to be handled. TS1 sizing is done within the render method. TS2 sizing is done in the _handleAddedToDOM event handler.

TS1

```
if (!AXES.isTS2Engine)
{
   /* TS1 sizing */
   var size = element.getSize();
   button.style.width = (size.width).toString() + "px";
   button.style.height = (size.height).toString() + "px";
}
```

TS2

```
_handleAddedToDOM: function(event)
{
/* TS2 sizing */
/* To improve rendering performance, The TS2 engine doesn't add a field to the */
/* DOM until all the extensions have been rendered. Elements do not have */
/* dimensions until they are added to the DOM so we use this event to do */
/* anything that requires knowledge of the size of things. */
/* For best rendering performance, avoid using this unless absolutely necessary. */

/* we have to refer to the element and the button via "this" */

   var size = this._element.getSize();
   this.buttonElem.style.width = (size.width).toString() + "px";
   this.buttonElem.style.height = (size.height).toString() + "px";

}
```

We are also going to apply the styles the developer may have specified in the style property:

```
   var style = this.getPropertyValue("style");
   this.applyStyle(style, btn);
```

The render method should look like this:

```
   render: function(element, elementContainer)
   {
      var button = this.buttonElem;
      if (!AXES.isTS2Engine)
      {
         /* TS1 sizing */
         var size = element.getSize();
         button.style.width = (size.width).toString() + "px";
         button.style.height = (size.height).toString() + "px";
      }
      var style = this.getPropertyValue("style");
      this.applyStyle(style, button);
   },
```

The _handleAddedToDOM event handler should look like this:

```
_handleAddedToDOM: function(event)
{
/* TS2 sizing */
/* To improve rendering performance, The TS2 engine doesn't add a field to the */
/* DOM until all the extensions have been rendered. Elements do not have */
/* dimensions until they are added to the DOM so we use this event to do */
/* anything that requires knowledge of the size of things. */
/* For best rendering performance, avoid using this unless absolutely necessary. */

/* we have to refer to the element and the button via "this" */

   var size = this._element.getSize();
   this.buttonElem.style.width = (size.width).toString() + "px";
   this.buttonElem.style.height = (size.height).toString() + "px";
},
```

Destroying

The *destroy* method is crucial for your extension to work properly. In this method you should remove any object references, objects, events and anything else that is not locally defined in one of the methods.

If you created a variable of any type in the *render* method like this:

var x = "aaa";

then you don't need to set it to null or do anything to it in the destroy method.

However, you may have added it as a reference to some other object, typically the **this** pointer or some html element. For example, you may have done something like this:

this.Xvar = x;

In this extension button we made a reference to the buttonElem:

```
   this.buttonElem = btn;
```

All references to it should be destroyed to avoid memory leaking. Add this code to the destroy method:

```
   destroy: function(element, elementContainer)
   {
      this.buttonElem.onclick = null;
      this.buttonElem.sendKey = null;
      this.buttonElem = null;
      delete this.buttonElem;

      element.removeListener("AddedToDOM", this._handleAddedToDOM);
      delete this._element;
      delete this._elementContainer;
   },
```

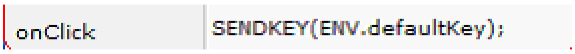
Events

For this extension we will define an onClick event.

When we attached the event to the button, we will pass the button as a parameter:

```
self._onClick(btn)
```

All we need to do in the extension is raise the event. The developer writes the event code at design time. When we raise the event, the event code is executed. In this button extension it will execute this code by default:



onClick SENDKEY(ENV.defaultKey);

When we set up the extension in the init method, we specified Enter as the default key, so this button will now automatically send the Enter key.

Replace this text:

```
<event handler(s)>
```

With this event routine:

```
_onClick: function(button)
{
   var env = { defaultKey : button.sendKey };
   this.raiseEvent("onClick", env);

}
```

Save the changes to your extension file.

Solution

The entire extension code should look like this:

```
function DEFINE_EXTENSION_SimplePushButton()
{
var definition =
        {

       name: "SimplePushButton",
       group: "vis",
       type: "element",
       display: "Simple Push Button",
       subType: "new",

       properties:
       {
        caption: { type: "String", defaultStatic:"OK", description: "Text to appear   on the
button." },
        style:   { type: "Style", description: "CSS to apply to the extension." },
        onClick: { type: "Event", defaultDynamic: "SENDKEY(ENV.defaultKey);", description: "
The button's onclick event." }
       },

       init: function(element, elementContainer)
       {
          this._element = element ;
          this._elementContainer = elementContainer ;

          var btnCaption = this.getPropertyValue("caption");

          var btn = document.createElement("button");
          btn.innerHTML = btnCaption;
          btn.sendKey   = "Enter";

          if (!this.isDesignMode())
          {
             var self = this;
             btn.onclick = function() { self._onClick(btn); };
          }

          this.applyTabIndex(btn);

          elementContainer.appendChild(btn);

          this.buttonElem = btn;


          if (AXES.isTS2Engine) element.addListener("AddedToDOM", this._handleAddedToDOM,
this);
       },

       render: function(element, elementContainer)
       {
          var button = this.buttonElem;
          if (!AXES.isTS2Engine)
          {
             /* TS1 sizing */
             var size = element.getSize();
             button.style.width = (size.width).toString() + "px";
             button.style.height = (size.height).toString() + "px";
          }
          var style = this.getPropertyValue("style");
          this.applyStyle(style, button);
       },

       destroy: function(element, elementContainer)
       {
          this.buttonElem.onclick = null;
          this.buttonElem.sendKey = null;
          this.buttonElem = null;
          delete this.buttonElem;

          element.removeListener("AddedToDOM", this._handleAddedToDOM);
          delete this._element;
          delete this._elementContainer;
       },

      _handleAddedToDOM: function(event)
      {
      /* TS2 sizing */
      /* To improve rendering performance, The TS2 engine doesn't add a field to the */
      /* DOM until all the extensions have been rendered. Elements do not have */
      /* dimensions until they are added to the DOM so we use this event to do */
      /* anything that requires knowledge of the size of things. */
      /* For best rendering performance, avoid using this unless absolutely necessary. */

      /* we have to refer to the element and the button via "this" */

         var size = this._element.getSize();
         this.buttonElem.style.width = (size.width).toString() + "px";
         this.buttonElem.style.height = (size.height).toString() + "px" ;

      },


     _onClick: function(button)
     {
       var env = { defaultKey : button.sendKey };
       this.raiseEvent("onClick", env);

     }

  };

  AXES.Extensions.add(definition);
}

DEFINE_EXTENSION_SimplePushButton ()
```

Testing the Extension

To test your new extension copy *Extension_SimplePushButton.js* and add it to your project folder (\ts\screens\<<your folder>>) or if you do not have one, to the \ts\screens folder. See The program development lifecycle.

Use the WRKLNK command in your IBM i to ensure that the file only has *PUBLIC *R authorities.

Start aXes as a developer.

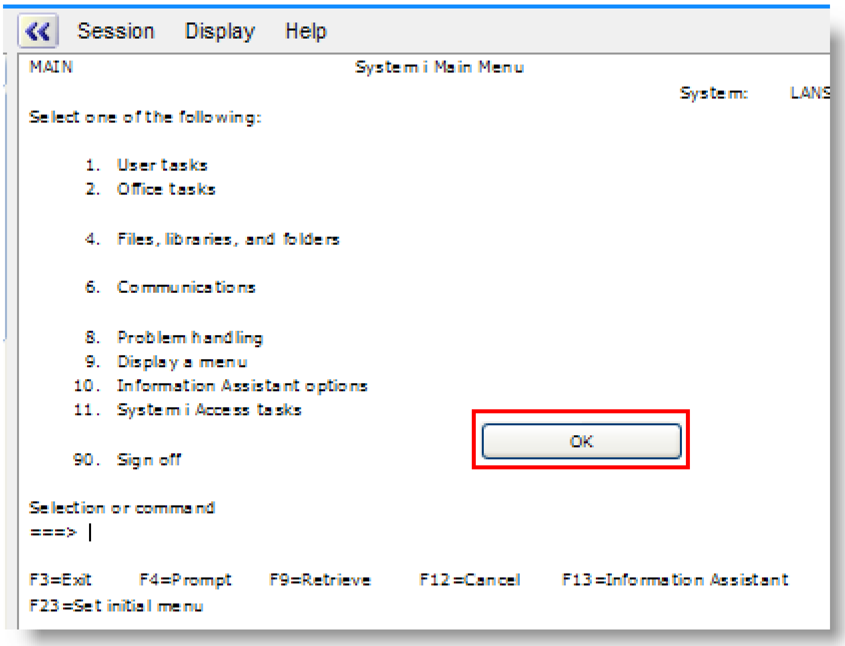
Start customizing a screen and add a new element.

Verify that the Simple Push Button extension is present in aXes Designer.

Select the Simple Push Button extension.

Size and position the button, then save your customized screen.

Test that clicking the button sends the Enter key.



```
MAIN                                System i Main Menu
                                                              System:     LANS
Select one of the following:

     1. User tasks
     2. Office tasks

     4. Files, libraries, and folders

     6. Communications

     8. Problem handling
     9. Display a menu
    10. Information Assistant options
    11. System i Access tasks

    90. Sign off

Selection or command
===> |

F3=Exit   F4=Prompt   F9=Retrieve   F12=Cancel   F13=Information Assistant
F23=Set initial menu
```

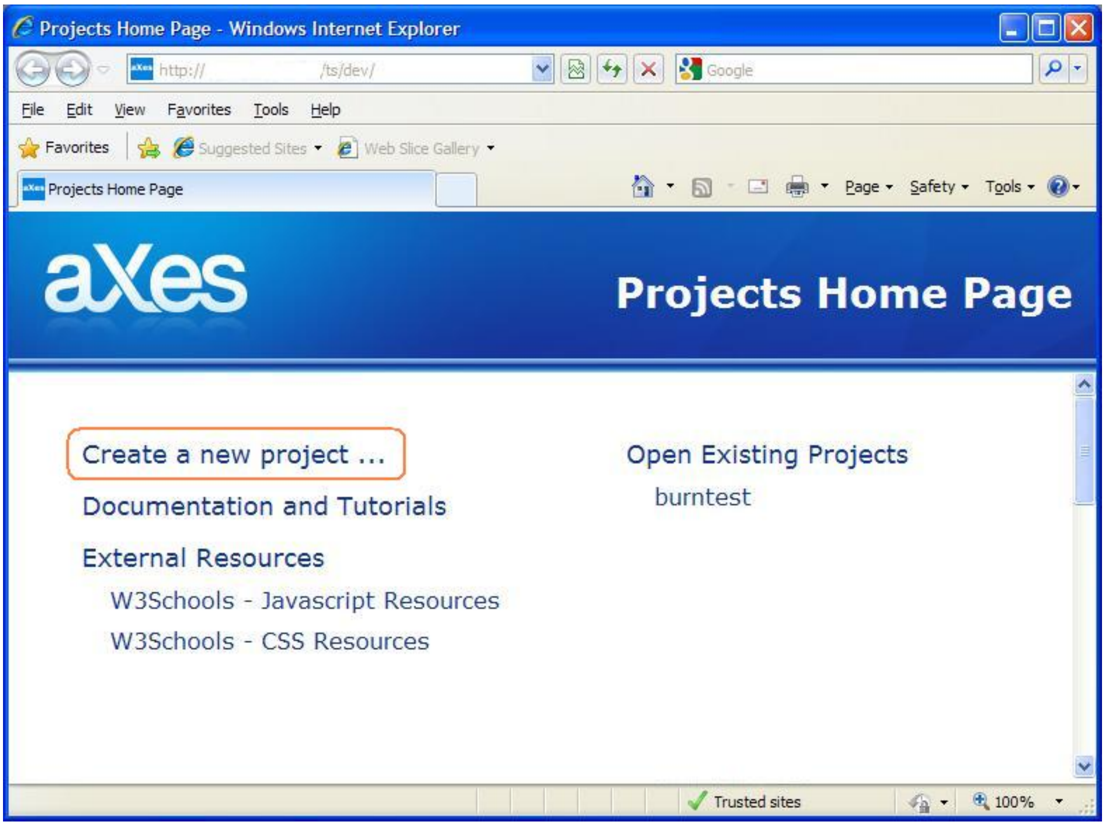
The program development lifecycle

Develop / Test

During the extension's develop / test phase, you should put your new extension in your project folder. This is found on the server, under

\\<<your axes installation name (typically axes)>>\ts\screens\<<your folder>>

If you don't yet have a project folder, you can sign on as a developer and create one from the projects home page:



By keeping your work in this folder, any mistakes you make will not impact on other developers or users.

Tested

When your extension is tested, if you want to make it available to users and other developers, you can promote your extension file to the screens folder:

\\<<your axes installation name (typically axes)>>\ts\screens

and delete it from your project folder.

(if you want the extension to be available only for your project you can leave it in the project directory and not promote it to the screens folder)

Deploy

To deploy your extension to another site, you just need to ensure that the Extension_XXXX.js file is moved into the screens folder in the target axes installation.

See Extension_Tutorial 9 for detailed instructions of how to create a complete deployment package which will include your new extension.

The extension checklist

Check / Consideration	**Okay**
The main wrapper function is named DEFINE_EXTENSION_XXEEEEEEEEEEE (where XX is NOT ax AX aX or Ax)	
The EEEEEEEEEEE portion of the name uses upper and lowercase	
The properties section identifies the name, group, type, display and subType properties of the extension	
There are standard published methods init, render and destroy	
All unpublished (private) functions are prefixed with _	
There is a call to DEFINE_EXTENSION_XXEEEEEEEEEEE();	
Properties	
• All published properties have a description property that refers to an appropriate long description (used as a tooltip) for the property	
• All published properties are named in camel case	
• All unpublished (private) properties are prefixed _	
The destroy method	
• Nullifies all references kept in private properties, e.g. this._myProperty	
• Destroys all arrays	
• Nullifies all references kept in HTML elements (very important to prevent leakage)	
• Unattaches all events previously attached	
The render method	
• All UI visualization logic is placed in render method so it correctly responds to property changes made at design time	
• Design mode awareness exists in logic, especially UI event handling	
• Visualization logic can handle the change of associated field content coming from places other than UI (eg: another script changes field content associated with a checkbox, checkbox should change)	
Tracing	
• Tracing has been added to complex or error prone areas via AXES.Trace.Output() functions	
• Tracing has not been added for basic execution of init, render or destroy routines - this is done automatically by the environment	

Tab positioning logic has been added in Init	
Tested in these situations:	
❖ Normal field – alpha input	
❖ Normal field – alpha output	
❖ Normal field – numeric input	
❖ Normal field – numeric output	
❖ Subfile field – alpha input	
❖ Subfile field – alpha output	
❖ Subfile field – numeric input	
❖ Subfile field – numeric output	
❖ New screen element	
❖ Multiple usage on same 5250 screen	
❖ Multiple usage on multiple 5250 screens	
Promoted to Screens folder	
Standard documentation sheet produced	

Questions?

If you have any questions about creating an extension, please contact your product vendor or aXes Support.

aXes® Documentation Library

eXtensions Tutorial 9 - Deployment

This document describes how you move files from your development system to your target system and how you provide your end-users access to the application (URL or icon).

Assumptions

- That deployment using these instructions is to a complete and operational production aXes system of the correct version and that no aXes screen development of any kind will be carried out with the deployed definition set.
- That at least Tutorials 0 through to 3 have been completed.
- That you have carefully considered how to Establish the Deployment Model as described in Tutorial 7 Best Practices.
- That in addition to the modified files listed in the ***Files that need to be Deployed*** section, all required unmodified files as outlined in Tutorial 0 - Getting Started are also deployed to the target application folder.

Deployment of Files to the Target Application Folder

To deploy all required files from an aXes development project follow these steps:

Deployment Step **Okay**

On the target system, create an application folder as a subfolder of the \axes\ts\screens folder using the ibm i CRTDIR command.

For example to create folder MyApplication1 in \axes\ts\screens\ folder:

```
CRTDIR DIR('axes\ts\screens\myapplication1')
```

The name for this folder:

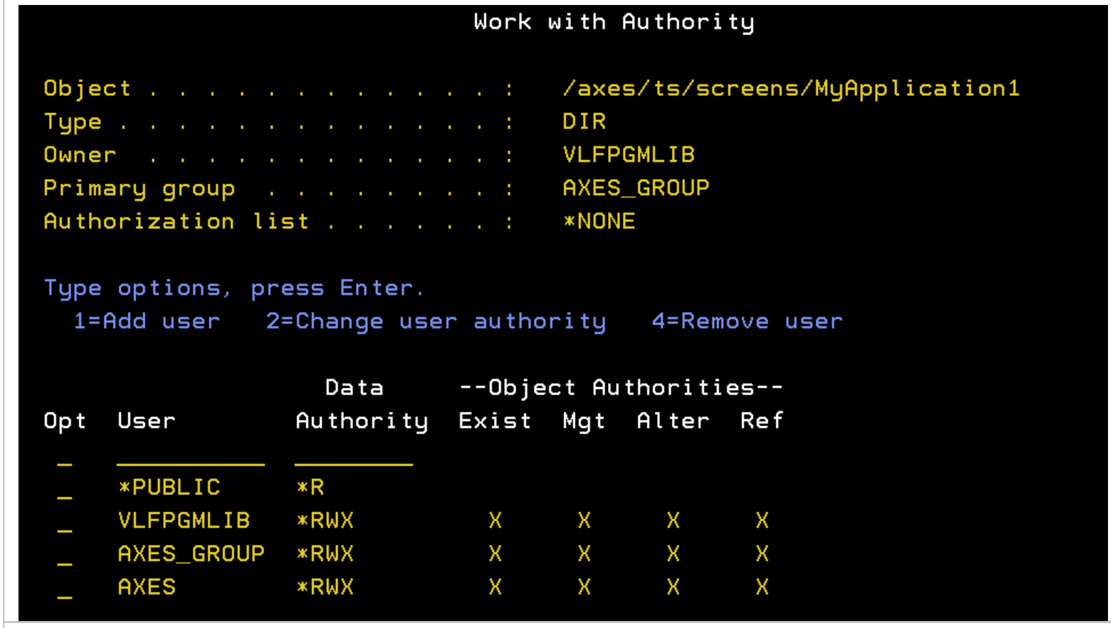
- Should not contain blanks.
- Should contain only letters from the English alphabet or numbers.
- Is referred to as your application's ***Definition Set***.

Remember, like aXes development projects, aXes deployed applications are discrete and indivisible:
merge definition sets together.
copy a definition set into another definition set.
split a definition set up into other definition sets.

It is normal to have multiple users using the same definition set.

Use the IBM i WRKLNK command and make sure that folder MyApplication1 has *R rights for user *PUBLIC and no other rights.

For example: Use WRKLNK OBJ('axes\ts\screens\ MyApplication1') then use option 9=Work with authority to display and alter the authority to folder MyApplication1. It should look like this when displayed by the WRKLNK command:



```
                               Work with Authority

 Object . . . . . . . . . . . . :   /axes/ts/screens/MyApplication1
 Type . . . . . . . . . . . . . :   DIR
 Owner  . . . . . . . . . . . . :   VLFPGMLIB
 Primary group  . . . . . . . . :   AXES_GROUP
 Authorization list . . . . . . :   *NONE

 Type options, press Enter.
   1=Add user   2=Change user authority   4=Remove user

                       Data      --Object Authorities--
 Opt  User           Authority   Exist  Mgt  Alter  Ref

  _   __________     ________
  _   *PUBLIC        *R
  _   VLFPGMLIB      *RWX          X     X     X     X
  _   AXES_GROUP     *RWX          X     X     X     X
  _   AXES           *RWX          X     X     X     X
```

Check that any static or dynamic table definitions used for testing have been removed from the respective definition files, Tables_Static.txt and Tables_Dynamic.txt.

These would typically be unused in the application now - and may fail if they are deployed to a production environment.

Check that the user the aXes server is executing under has READ authorisation to any required SQL/database tables.

SQL commands may be used to load static or dynamic tables in your application.
The commands execute under the user profile that the aXes server is executing under.

You should authorise that user profile for read access to the SQL tables required for this purpose. This user

does not need any rights to other data base tables that may be on the system.	

Next, copy the files listed in the Files That Need to be Deployed section from your project folder to the application folder on the target system.
 	
Use the IBM i CPY commands and/or save/restore commands (using Windows copy commands can cause code page errors). You need to save the files to a save file and move the save file onto a PC to move it and then restore it to the target iSeries.
 	
For example this command saves the contents of the /axes directory to a save file:
`    SAV DEV('/qsys.lib/savefilelib.lib/saveFile.file') OBJ(('/axes'))`
 	
This command saves the contents of /axes and /axesdemo directory:
`    SAV DEV('/qsys.lib/savefilelib.lib/saveFile.file') OBJ(('/axes') ('/axesdemo')) -`
 	
To restore the save file:
`    RST DEV('/qsys.lib/savefilelib.lib/saveFile.file') OBJ(('/axes'))`
 	
CPY examples:
`    CPY  OBJ('/axesbuild/build/110/default_FCGI.conf') TODIR(/TGT_PATH)`
`    CPY  OBJ(/WRKPATH/*) TODIR(/TOPATH) SUBTREE(*ALL) REPLACE(*YES) OWNER(*KEEP)`	

Finally, copy the application launch page, **/axes/index.html**, and the icon used in the browser and on desktops, **/axes/favicon.ico**, to the corresponding folder in the target aXes instance if these files have been customized. Note that this step is only required when deploying to a different aXes instance.	

Files That Need to be Deployed

The following files need to be copied from the aXes development project folder to the application folder on the target aXes server. Some of these may not exist in the project folder.

- application_definition.css (plus application_definition_*.css)
- application_definition.js
- Extension_*.js
- screen_*.js
- screens.jsn
- Tables_Static.txt (see note 3)
- Tables_Dynamic.txt (see note 3)
- Userenv.js
- *.xml

NOTES:

- The *.scn files and the count.txt file do not need to be deployed from the development project folder into **PRODUCTION** application folders because these files are amalgamated into the screens.jsn file. **No aXes screen development can be carried out within production application folders because the absence of *.scr files will cause all previously deployed screen definitions to be lost.**

  If the application is to be displayed in a language other than English the modified text files, Texts_Cust_*.txt also need to be copied. See the Application Internationalization section for the location of these files and additional deployment steps. Note that if you change a Texts_Cust_ll.txt file you should clear your browser cache to pick up the new file version.

- If you have used additional static or dynamic table definition files - deploy these files as well. If you have changed the names of these files - deploy the renamed files instead.

aXes eXtension Files

When aXes starts up, eXtensions are loaded from the axes\ts\screens folder first. Then eXtensions are loaded from the application definition set if present.

This means that the following extra steps need to be taken for deployment of aXes eXtensions:

Check	Okay
aXes eXtensions that are specific for this application only are to be copied to the application folder (as previously described).	
aXes eXtensions that are to be used by all applications on this system are to be copied from the axes/ts/screens folder of the development system to the axes/ts/screens folder of this system. **NOTE:** If the same eXtension exists in the screens folder and the application folder, the eXtension in the application folder should take precedence. However, this cannot be guaranteed and you should check which version is being used.	

Starting aXes on the Target System

You will remember from a previous tutorial that to start aXes to access applications without eXtensions, you use a URL of the following format to get you to your Server:

```
http://<aXes_Host>:<aXes_Port_Number>/ts/skins/ts_basic.html
```

Replacing
<aXes_Host> with the Host Name of your aXes Server
<aXes_Port_Number> with the Port Number required to access your aXes Server

Using aXes to access an application on your Server that has been modernized with aXes eXtensions, you use a URL with a different format:

```
http://<aXes_Host>:<aXes_Port_Number>/ts/skins/ts_basic.html?definitionset=<Application_folde
r>&lang=<LL>
```

Replacing
<aXes_Host> with the Host Name or IP Address of your aXes Server
<aXes_Port_Number> with the Port Number required to access your aXes Server

<application_folder> with the name of the folder that contains your application's Definition Set (see later)
<LL> with the appropriate language code from the Language Codes table. If not specified it will default to the Windows default language on the PC.

Note that when you start aXes using ts_basic.html, you are starting an aXes-TS 5250 terminal session without access to the aXes-WS Web Spooler capability.

To start a full aXes-TS terminal session which allows the viewing of spool files as HTML, PDF, XML or Text documents, start aXes using ts.html:

```
http://<aXes_Host>:<aXes_Port_Number>/ts/skins/ts.html?definitionset=<Application_folder>&lan
g=<LL>
```

The above URL applies to the aXes-TS engine. If you are using the aXes-TS2 engine, the url is:

```
http://<axes_host>>:<aXes_Port_Number>/ts/ts2/index.html?definitionSet=<Application_folder>&l
ang=<LL>
```

When the deployed application is to be used in IE, you can choose whether to use the aXes-TS or the aXes-TS2 engine. However, if your end-users' browser is Firefox, Google Chrome or Safari you must use the aXes-TS2 engine.

Put a Start Icon on End User Desktops

As a convenient way to start the completed aXes application you can provide a start icon for the end user's desk top. The shortcut doesn't need to be created from scratch: it can simply be emailed to the user and dragged on to the user's desktop.

Ensure that the shortcut location correctly refers to the deployed application folder.

Don’t forget that the format of the URL required to access the application that you have modernized with aXes eXtensions is:

```
http://<aXes_Host>:<aXes_Port_Number>/ts/skins/ts_basic.html?definitionset=<Application_folde
r>&lang=<LL>
```

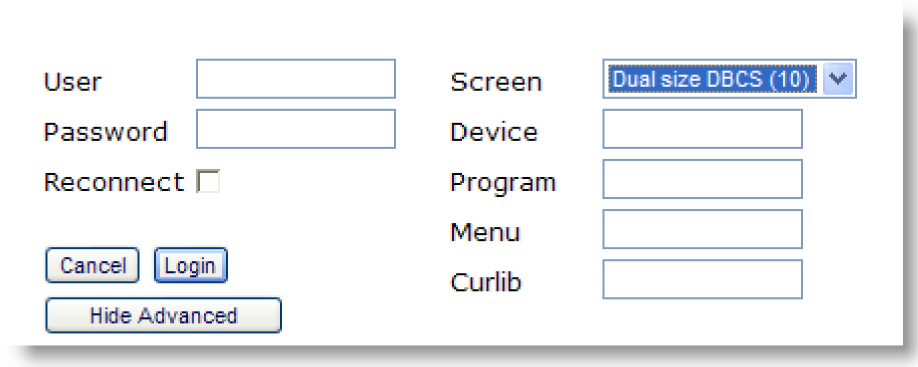
URL Parameters

You can specify parameters such as the user id and the device in the aXes URL:

Parameter	Description	Applicable to TS2 only
curlib	IBM i library	
definitionset	The name of the folder that contains your application's definition set.	
dev	Turns on developer mode. Currently developer mode supports defining screen ids and changing AutoGUI settings. Extension design must be done in TS1.	Yes
device	Device to connect to	
lang	The appropriate language code from the Language Codes table. If not specified it will default to the Windows default language on the PC.	
menu	Initial IBM i menu	
noexpires	By default, all files loaded by aXes (that are not expected to change) are cached by the browser to make future loads faster. Adding the noexpires parameter tells aXes not to do this (can be useful during development). This affects long-term caching by controlling whether the server sets an "expires" header on the files it sends. Browsers may do some internal session level caching of their own that may still result in changes not being loaded. Even clearing the cache may not fix this (Google Chrome). Quitting and re-launching the browser usually will.	
popups	This parameter can have one of these values: 0 – Off all enhanced window handling. 1 – Turns on the detection of degraded DDS windows only. 2 – Turns on the "keep the previous screen background" behaviour only. 3 – Turns on both. From version 2.11 onwards, 3 is the default value used if this parameter is omitted. See PopUp Windows and Screen Rendering in aXes-TS2.	Yes
Program	Initial IBM i program	
pwd	Password. For security reasons you need to consider carefully whether you want to expose a password in a url.	
reconnect	Reconnect to a previously disconnected session. Can be true or false.	
screentype	IBM API screen type identifier	
signon	Automatically sign on. Can be true or false.	
trace	Turn on tracing and set the level. Valid values are: s/sys/system u/user/app/application 1/y/a/all/true	
user	Specifies and prefills the user profile to be used in the aXes logon dialog. If the user and password are both specified then the log on is performed automatically.	

This URL assigns user, password, device and auto-signon:
http://<aXes_Host>:<aXes_Port_Number>/ts/skins/ts.html?device=ARB&user=alick&pwd=XXXX&signon=true

The URL parameters correspond to the options you can specify when logging on to aXes:



User

Password

Reconnect □

Screen: Dual size DBCS (10)

Device

Program

Menu

Curlib

Note that the LUA *tslogonexit.lua* on the server controls whether client overrides are allowed. If the LUA variable *allowClientOverride* is set to False, the URL parameters are ignored.

Tracing Your Application

If you have a problem with a deployed application, you may want to trace it by adding the Trace parameter to the URL. For example:

```
http://lxsteam:8080/ts/skins/ts_basic.html?definitionset=myfolder&trace=all
```

Possible values of the trace parameter are:

ALL	Displays both system trace messages and application trace messages.
SYS	Displays only system trace messages
APP	Displays only application trace messages
NO	No trace messages are displayed. You can use this value by default to make it quick to turn on tracing: `http://lxsteam:8080/ts/skins/ts_basic.html?definitionset=myfolder&trace=no`

System trace messages are those that are outputted by aXes core system.

Application trace messages are generated for eXtensions when TRACE commands are encountered in aXes code:

```
TRACE("The value in the job title field is: ", value, " more text ", " and more text" );
```

To trace aXes outside extensions use AXES.**Trace**. For example:

```
AXES.Trace.output("This is an application trace");
```

Application Internationalization

During the development of an aXes project designed to be executed in languages other than English, any customer visible text must be defined into the aXes TEXT object.

Note that if you change a Texts_Cust_ll.txt file you should clear your browser cache to pick up the new file version.

Follow these steps as a guide to application internationalization:

Internationalization Step	Okay
All application text that is to be visible to the end user is defined in the file Texts_Cust_en.txt using the prescribed "key" : "text" format. Note that if you change a Texts_Cust_ll.txt file you should clear your browser cache to pick up the new file version. **NOTE:** See Tutorial 3 for an example of this technique.	
All application text that is to be visible to the end user is displayed using the TEXT object. For example **`CTEXT("<text key>")`** Where **`<text key>`** is substituted with the key of the required text in the Texts_Cust_en.txt file	
The Texts_Cust_en.txt file is translated in the required language and the translations are stored in a file named Texts_Cust_LL.txt where LL is substituted with the appropriate language code from the table in the Language Codes section.	
The file Texts_Cust_en.txt plus any translated versions of this file, e.g. Texts_Cust_fr.txt for French translations, need to be copied from the \axes\ts\lang folder on the development system to the corresponding folder of the target system.	
Include the appropriate language code in the start icon for the end-user's desktop or any aXes URL used to start the application. For example: `http://<aXes_Host>:<aXes_Port_Number>/ts/skins/ts_basic.html?definitionset=<Application_folder>&lang=<LL>` Replacing **<aXes_Host>** with the Host Name or IP Address of your aXes Server **<aXes_Port_Number>** with the Port Number required to access your aXes Server **<application_folder>** with the name of the folder that contains your application's Definition Set **<LL>** with the appropriate language code from the Language Codes table. If not specified it will default to the Windows default language on the PC.	

Language Codes

The following is a list of language codes to be used with internationalised applications. They are always lowercase:

af Afrikaans	sq Albanian
ar-sa Arabic (Saudi Arabia)	ar-iq Arabic (Iraq)
ar-eg Arabic (Egypt)	ar-ly Arabic (Libya)
ar-dz Arabic (Algeria)	ar-ma Arabic (Morocco)
ar-tn Arabic (Tunisia)	ar-om Arabic (Oman)
ar-ye Arabic (Yemen)	ar-sy Arabic (Syria)
ar-jo Arabic (Jordan)	ar-lb Arabic (Lebanon)
ar-kw Arabic (Kuwait)	ar-ae Arabic (U.A.E.)
ar-bh Arabic (Bahrain)	ar-qa Arabic (Qatar)
eu Basque	bg Bulgarian
be Belarusian	ca Catalan
zh-tw Chinese (Taiwan)	zh-cn Chinese (PRC)
zh-hk Chinese (Hong Kong SAR)	zh-sg Chinese (Singapore)
hr Croatian	cs Czech
da Danish	nl Dutch (Standard)
nl-be Dutch (Belgium)	en English
en-us English (United States)	en-gb English (United Kingdom)
en-au English (Australia)	en-ca English (Canada)
en-nz English (New Zealand)	en-ie English (Ireland)
en-za English (South Africa)	en-jm English (Jamaica)
en English (Caribbean)	en-bz English (Belize)
en-tt English (Trinidad)	et Estonian
fo Faeroese	fa Farsi
fi Finnish	fr French (Standard)
fr-be French (Belgium)	fr-ca French (Canada)
fr-ch French (Switzerland)	fr-lu French (Luxembourg)
gd Gaelic (Scotland)	gd-ie Gaelic (Ireland)
de German (Standard)	de-ch German (Switzerland)
de-at German (Austria)	de-lu German (Luxembourg)
de-li German (Liechtenstein)	el Greek
he Hebrew	hi Hindi
hu Hungarian	is Icelandic
id Indonesian	it Italian (Standard)
it-ch Italian (Switzerland)	ja Japanese
ko Korean	ko Korean (Johab)
lv Latvian	lt Lithuanian
Macedonian (FYROM)	ms Malaysian
mt Maltese	no Norwegian (Bokmal)
no Norwegian (Nynorsk)	pl Polish
pt-br Portuguese (Brazil)	pt Portuguese (Portugal)
rm Rhaeto-Romanic	ro Romanian
ro-mo Romanian (Moldavia)	ru Russian
ru-mo Russian (Moldavia)	sz Sami (Lappish)
sr Serbian (Cyrillic)	sr Serbian (Latin)
sk Slovak	sl Slovenian
sb Sorbian	es Spanish (Spain ? Traditional)
es-mx Spanish (Mexico)	es Spanish (Spain ? Modern)
es-gt Spanish (Guatemala)	es-cr Spanish (Costa Rica)
es-pa Spanish (Panama)	es-do Spanish (Dominican Republic)
es-ve Spanish (Venezuela)	es-co Spanish (Colombia)
es-pe Spanish (Peru)	es-ar Spanish (Argentina)
es-ec Spanish (Ecuador)	es-cl Spanish (Chile)
es-uy Spanish (Uruguay)	es-py Spanish (Paraguay)
es-bo Spanish (Bolivia)	es-sv Spanish (El Salvador)
es-hn Spanish (Honduras)	es-ni Spanish (Nicaragua)
es-pr Spanish (Puerto Rico)	sx Sutu
sv Swedish	sv-fi Swedish (Finland)

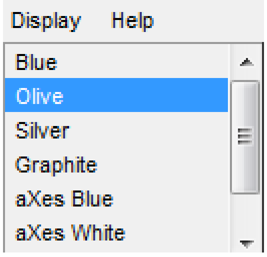
th Thai	ts Tsonga
tn Tswana	tr Turkish
uk Ukrainian	ur Urdu
ve Venda	vi Vietnamese
xh Xhosa	ji Yiddish
zu Zulu	

aXes Documentation Library

eXtensions Tutorial 10 - 5250 Screen Styling

The Shipped 5250 Basic Themes and Customized Styles

aXes ships with a set of basic 5250 themes. These are accessible from the aXes menu:



Once you start to customize an application you should ***stop*** using the basic 5250 themes and develop your own customized role based styles and themes. Tutorial 3 introduced this concept. The following tutorial covers it in much more depth.

Initially some of this material may seem complex to you. However the time you invest in understanding it will greatly improve the final appearance of your application.

Using Role Based Styles

By now you should have encountered the concept of using role based styles for screen elements.

This means you might create a style called ***Instruction*** that uses the Verdana font, in blue, using 10 pt italics.

On customized 5250 screens, rather than specify style elements over and over for screen instructions, you simply associated the style ***Instruction*** with the screen instructions.

The role based approach is faster, more accurate and much easier to change than applying individual style elements to the screen fields.

Here's a list of some of the common roles that information on a 5250 screen has:

Panel Identifier
Panel Title
Instruction
Field Label
Field Column Heading
Group Heading
Normal Text
Emphasised Text
Input field – Normal
Input Field – Emphasised
Output field – Normal
Output field - Emphasised
Scrolling Information
Separator
Function Key
Error Message

This basic list of screen element roles is traceable right back to the IBM CUA (Common User Access) standards developed in the late 1980s. Role based styling is not a new idea.

Do not have too many styles

You may end up with 10, 20 or even 30 individual role based styles.

If you have more than 30, you may be misinterpreting how to use them.

Using Themes

A further refinement of the style approach is to define overarching themes for your whole application.

Themes are just names, and typically they reflect an overall theme or style of appearance across a whole application.

You might dream up exotic theme names like ***Midnight***, ***Sunset*** and ***Moonlight***, but commonly designers use themes that are influenced by Microsoft - like their ***Blue***, ***Silver*** and ***Olive*** themes. First introduced with Windows XP these themes are known to millions of people throughout the world.

In this example we are going to use 4 themes:

Blue
Silver
Olive
Graphite

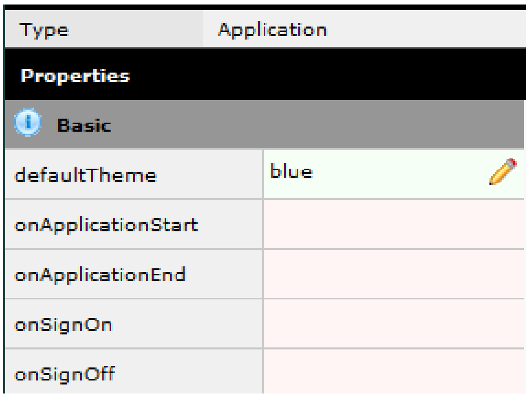
Usually the first thing you want to associate with a theme is a primary background color - so here is what we are going to use in this example:

Blue	#A9CAF7	🟦
Silver	#E7E8EB	⬜
Olive	#CDDCC5	🟩
Graphite	#B8BABE	⬛

In the application properties you would then define 4 basic styles like this (in fact this is really one style named **BasicWindowBackground** which has 4 themes):

Name	**styleFor**	**htmlTag**	**Style**	**Theme**
BasicWindowBackground	Application Window		Background-color: #A9CAF7	Blue
BasicWindowBackground	Application Window		Background-color: #E7E8EB	Silver
BasicWindowBackground	Application Window		Background-color: #CDDCC5	Olive
BasicWindowBackground	Application Window		Background-color: #B8BABE	Graphite

Next the basic application property **defaultTheme** is set to indicate the default theme is blue:



Type	Application
Properties	
Basic	
defaultTheme	blue
onApplicationStart	
onApplicationEnd	
onSignOn	
onSignOff	

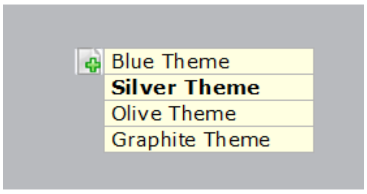
Note that the default theme is a property that can be derived from script - so you can change the defaultTheme at application start up.

> Note: If you are using aXes with RAMP-TS you would normally not bother to set up screen background colors. RAMP-TS will set the background color of its own current theme automatically.

Dynamically Changing Themes

You can also change the theme being used dynamically.

In this example a *quick pick menu* extension has been added to the System i Main Menu like this:



The quick pick menu's **onItemSelection** property has this script:

```
var sTheme = "";

switch (ENV.itemNumber)
{
   case 2:  sTheme = "SILVER"; break;
   case 3:  sTheme = "OLIVE"; break;
   case 4:  sTheme = "GRAPHITE"; break;
   default: sTheme = "BLUE"; break;
}

SETAXESTHEME(sTheme);
```

This script is using the function SETAXESTHEME() to alter the theme that is being used dynamically.

If you do this you will see the main system menu dynamically change color.

You can also use var sTheme = GETAXESTHEME() to find out what the current theme is in your scripts.

> Note: If you are using aXes with RAMP-TS you should not dynamically change themes this way. Instead you should follow the theme that RAMP-TS is using. It can be accessed in an eXtension script as RAMP.GLOBAL_VL_Theme. It will contain one of strings "BLUE", "OLIVE", "SILVER" or "GRAPHITE". Additionally the background color that RAMP-TS is using can be accessed as RAMP.GLOBAL_HTML_BackColor which will contain a #RRGGBB style color value.

Applying Themes to Screen Elements

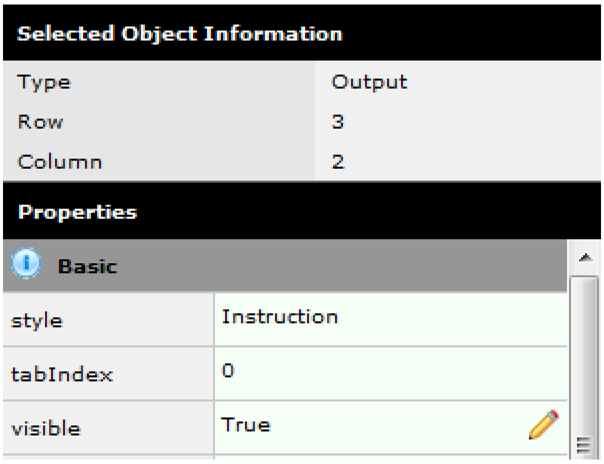
Next we are going to use the same approach to handle a role based screen element.

A new style named **Instruction** (in all 4 themes) is defined like this:

Name	styleFor	htmlTag	Style	Theme
Instruction			color: red	Blue
Instruction			color: green	Silver
Instruction			color: blue	Olive
Instruction			color: white font-weight: bold	Graphite

> Note: These color choices are to make this example clear. They are not good choices for a real application because they are not sympathetic to the overall theme, with the possible exception of the Graphite one.

Next, the instruction line on the System i Main menu has the style **Instruction** associated with it, like this:



Selected Object Information

Type	Output
Row	3
Column	2

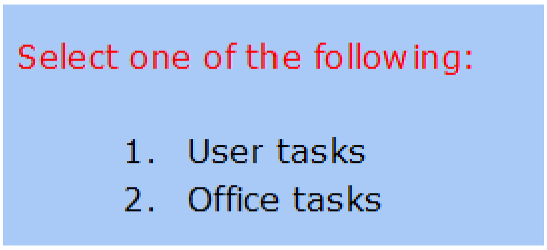
Properties

Basic

style	Instruction
tabIndex	0
visible	True

Now as the theme is changed by the quick pick menu, the style of the instruction line changes as well:

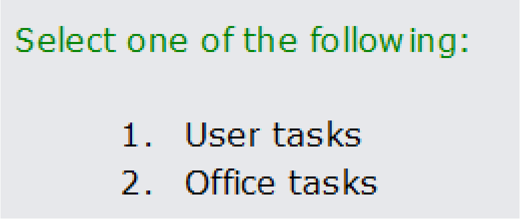
Blue Theme



Select one of the following:

1. User tasks
2. Office tasks

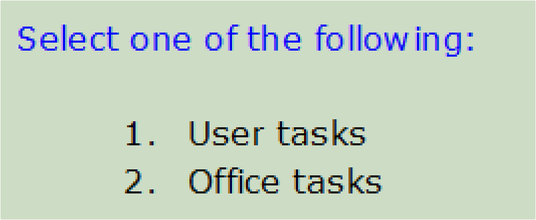
Silver Theme



Select one of the following:

1. User tasks
2. Office tasks

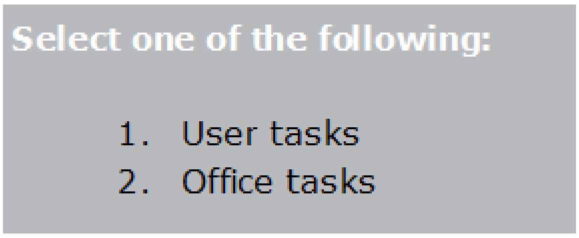
Olive Theme



Select one of the following:

1. User tasks
2. Office tasks

Graphite Theme



Select one of the following:

1. User tasks
2. Office tasks

Next, we are going to create a new style named **Title**.

This time we are going to just define these 2 entries:

Name	styleFor	htmlTag	Style	Theme

Title			color: blue;
			font-weight: bold;
			font-style: italic;	

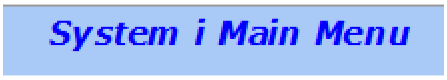
Title			color: white;
			font-weight: bold;
			font-style: normal;
				Graphite

Title is defined once with no theme - and again, but only for the theme Graphite.

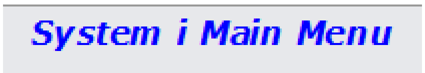
Next the style **Title** is associated with the title on the IBM i Main Menu.

As the 4 themes are cycled it appears as:

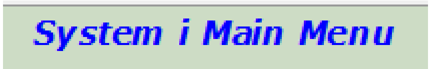
Blue Theme


System i Main Menu

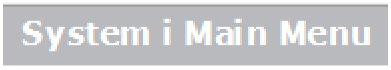
Silver Theme


System i Main Menu

Olive Theme


System i Main Menu

Graphite Theme

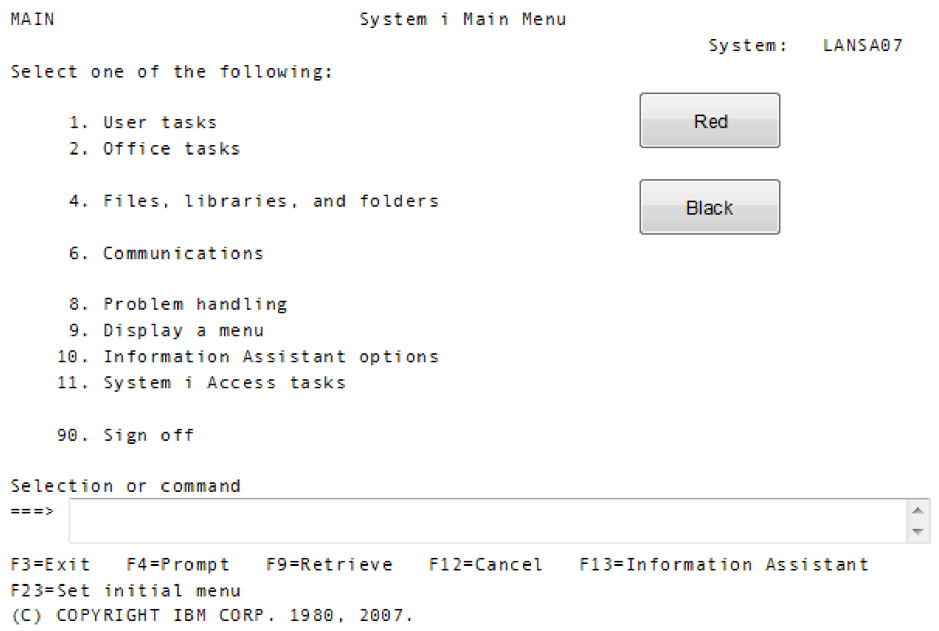

System i Main Menu

Where most of the themes use the same style elements - you only need to define the styles that are different. Here blue, silver and olive all use the Title style that has no associated theme.

Setting Styles Dynamically – Basic Concepts

The starting point for this example is the System I Main Menu, where:

- The 5250 screen is named MAIN
- The selection or command field is named CommandLine.
- Two push button eXtensions, captioned Red and Black, have been added.



```
MAIN                              System i Main Menu
                                                                  System:    LANSA07
Select one of the following:

     1. User tasks
     2. Office tasks

     4. Files, libraries, and folders

     6. Communications

     8. Problem handling
     9. Display a menu
    10. Information Assistant options
    11. System i Access tasks

    90. Sign off

Selection or command
===>

F3=Exit   F4=Prompt   F9=Retrieve   F12=Cancel   F13=Information Assistant
F23=Set initial menu
(C) COPYRIGHT IBM CORP. 1980, 2007.
```

The onClick property of the Red push button is set to this:

```
var oStyle = { "color":"yellow", "background-color":"red" };

var cmdfld= FIELDS("CommandLine");

cmdfld.setProperty("axdv.style",oStyle);

cmdfld.refresh();
```

The onClick property of the Black push button is set to this:

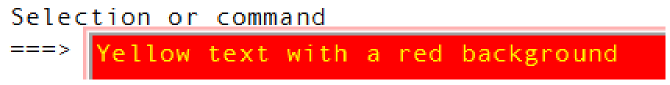
```
var oStyle = new Object();
oStyle["color"] = "white";
oStyle["background-color"] = "black";

var cmdfld= FIELDS("CommandLine");

cmdfld.setProperty("axdv.style",oStyle);

cmdfld.refresh();
```

These button click scripts first create a script object (named oStyle here) that specifies a text color and a background color.

Next they get a reference to the field CommandLine on the screen and alter its style (in fact the style of axdv - its default visualization).

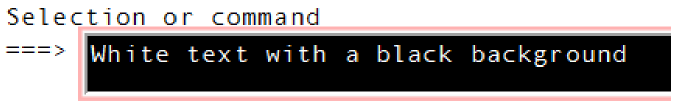
Finally the command line field is then instructed to refresh its visualization.

If you click the Red button the command line entry field should look like this:



```
Selection or command
===> Yellow text with a red background
```

If you click the Black button the command line entry field should look like this:



```
Selection or command
===> White text with a black background
```

In eXtension scripting styles are defined in *JavaScript objects*.

So to create a style dynamically you could code this:

```
var oStyle = { "color":"yellow", "background-color":"red" };
```

or you could code this - which may be more familiar:

```
var oStyle = new Object();

oStyle["color"] = "white";

oStyle["background-color"] = "black";
```

Note: The style element names like color and background-color that you use are identical to those you see when you set up a style and correspond to the standard CSS style names. They are case sensitive names.

Both these style object construction techniques are functionally identical, but the second technique generally makes it easier to put if/else logic into the set up.

You can also mix and match the techniques like this:

```
var oStyle = { "background-color":"red" };

if (condition) oStyle["color"] = "white";
else           oStyle["color"] = "palestraw";
```

Setting Styles Dynamically – Self Styling

The preceding Basic Concepts example uses two push buttons to alter the style of the command line. This is typical of the situation where a user's action like clicking the Save button may cause screen elements to change their style.

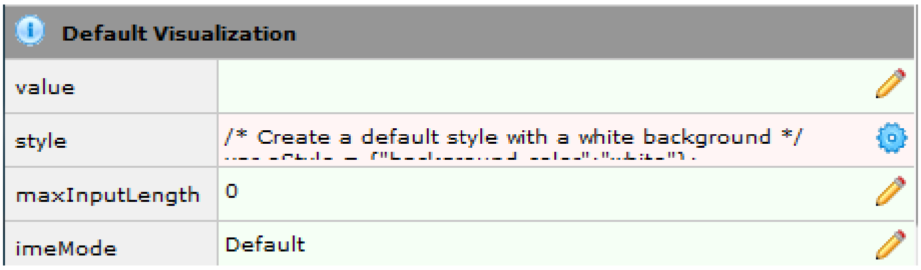
Another way to style screen style elements is to let them style themselves.

Typically they do this by looking at their own content and style themselves to attract the user's attention.

For example, a YES/NO field related to whether a customer should be extended more credit might change itself to red when it contains NO.

Self styling can be demonstrated by altering the example started in the preceding Basic Concepts section as follows.

First, the CommandLine field is selected, and the Style property of its Default Visualization is changed to be evaluated by executing script:



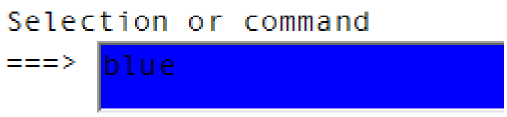
Default Visualization	
value	
style	/* Create a default style with a white background */
maxInputLength	0
imeMode	Default

Where the script used is like this:

```
/* Create a default style with a white background */

var oStyle = {"background-color":"white"};

/* See what is in this field (ie: CommandLine) */

var sValue = FIELD.getValue().toUpperCase();

/* If it contains blue or green make it the background color */

if ((sValue == "BLUE") || (sValue == "GREEN")) oStyle["background-color"] = sValue;

/* Return the style object to be used for this field (ie: CommandLine) */

ENV.returnValue = oStyle;
```

This change is saved and then the System I Main Menu is executed.

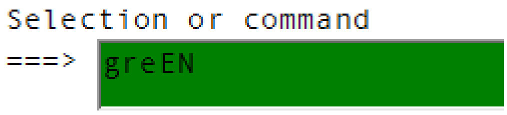
If you type the word green or blue into the command line and press enter (or any key that causes the screen to be redisplayed by the server), you should see the style change like this:



Selection or command

===> blue

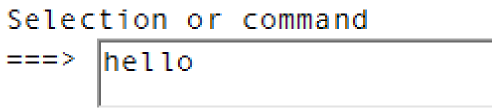
Or like this:



Selection or command

===> greEN

Type anything else into the command area and the default background color of white will be used:



Selection or command

===> hello

Setting Styles Dynamically – Using the USERENV object

The preceding Basic Concepts example used hard coded style elements for the color and background color.

By using the USERENV object you can centralize and abstract style logic.

First, edit your USERENV object.

Add the highlighted code:

```
var USERENV =
{

   REDStyle   : { "color" : "yellow", "background-color" : "red"   },
   BLACKStyle : { "color" : "white",  "background-color" : "black" },
```

This code defines new objects USERENV.REDStyle and USERENV.BLACKStyle.

Save your USERENV object changes. Then start a new aXes developer session to make sure you pick up the modified USERENV.JS file.

Change the onClick property of the Red push button to this:

```
var cmdfld= FIELDS("CommandLine");

cmdfld.setProperty("axdv.style",USERENV.REDStyle);

cmdfld.refresh();
```

Change the onClick property of the Black push button to this:

```
var cmdfld= FIELDS("CommandLine");

cmdfld.setProperty("axdv.style",USERENV.BLACKStyle);

cmdfld.refresh();
```

It's worth noting this about this new code:

- USERENV.REDStyle and USERENV.BLACKStyle are defined in one place. Where many scripts refer to them this makes changing them much simpler.

- This code version is marginally more efficient. The original version had to repeatedly create style objects. This version just refers to a pre-existing style - eliminating the cost of creating objects.

- Using style names like RED and BLACK is not recommended - but they suit the purpose of making this example simple.

  In a real application your style names should generally not reflect the physical characteristics of the style, such as its color.

Instead they should reflect its role - like EMPHASIZED or ERROR. Using the role based approach abstracts the definition of how an ERROR is actually visualized away from individual scripts.

- You can probably start to see how you can reduce your scripting even more by putting progressively more compound functions into the USERENV object.

  You should be able to reduce the scripting required for the Red button to a single line like this:

  ```
  USERENV.setStyle("CommandLine",USERENV.REDStyle);
  ```

  And to this for the black button:

  ```
  USERENV.setStyle("CommandLine",USERENV.BLACKStyle);
  ```

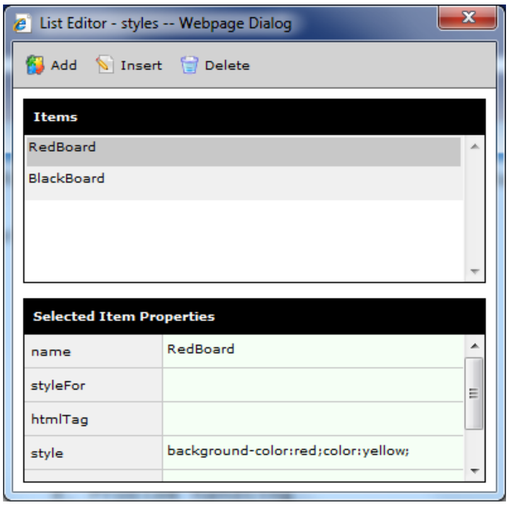
Setting Styles Dynamically – Using Application Styles/Themes

The preceding Basic Concepts example used hard coded style elements for the color and background color.

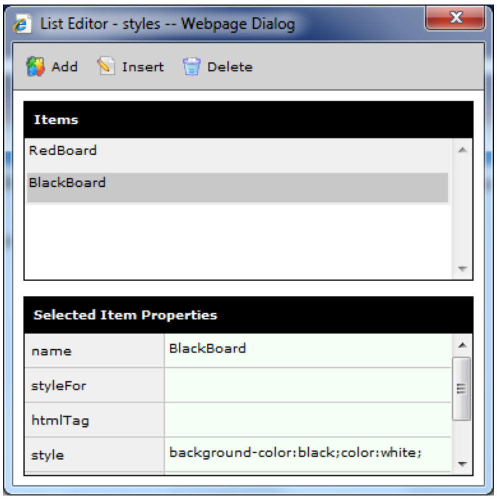
A better solution may be to use application level styles.

This approach also allows the styles to be ***dynamically themed***.

If the application level styles named *RedBoard* and *BlackBoard* are defined as:



name	RedBoard
styleFor	
htmlTag	
style	background-color:red;color:yellow;



name	BlackBoard
styleFor	
htmlTag	
style	background-color:black;color:white;

Then the code for the Red button could be changed to this better code:

```
var oStyle = { _base : "RedBoard" };
var cmdfld= FIELDS("CommandLine");
cmdfld.setProperty("axdv.style",oStyle);
cmdfld.refresh();
```

and for the black button:

```
var cmdfld= FIELDS("CommandLine");
cmdfld.setProperty("axdv.style", { _base : "BlackBoard" } );
cmdfld.refresh();
```

Here the special property _***base*** in the style object is used to specify the application level style name.

Generally using an application level style is better because it centralizes the definition of the style and it performs slightly better.

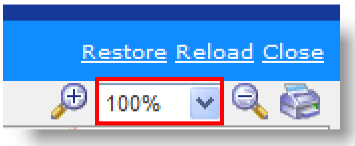
Understanding Screen Sizing and the Thin Red Line

aXes is designed to display 5250 screens, which are typically 24x80 or 27x132 lines of characters/columns. By default aXes will display an area that represents this type of screen size (also see the next section about row / column size to understand how the default 5250 screen size is calculated).

When enhancing screens you can create more screen space - but before doing this it is best to understand how the screen sizing process works.

Display the System i Main Menu (named MAIN).

Set the aXes zoom factor to 100% using the drop-down in the right top corner of the screen.

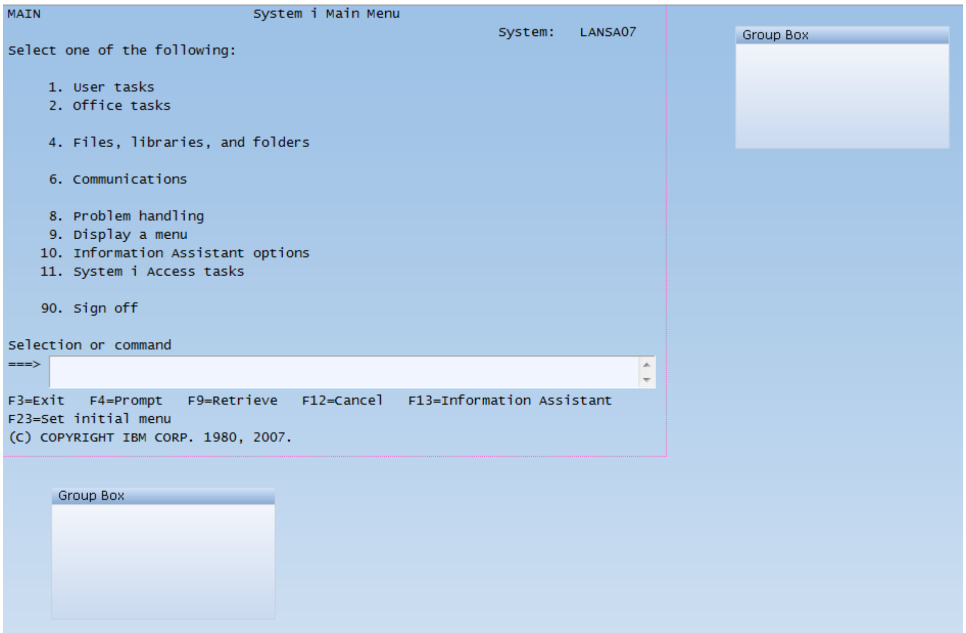


Stretch the browser window out so that what you are seeing is much larger than the 5250 screen display area.

Put the screen in edit mode by clicking on Edit Screen in the aXes Designer.

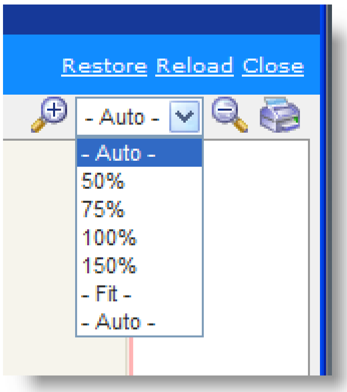
Add two new group boxes to the screen.

Position them outside the thin red line screen boundary - something like this:


```
MAIN                          System i Main Menu
                                                              System:   LANSA07
Select one of the following:

     1. User tasks
     2. Office tasks

     4. Files, libraries, and folders

     6. Communications

     8. Problem handling
     9. Display a menu
    10. Information Assistant options
    11. System i Access tasks

    90. Sign off

Selection or command
===>

F3=Exit   F4=Prompt   F9=Retrieve   F12=Cancel   F13=Information Assistant
F23=Set initial menu
(C) COPYRIGHT IBM CORP. 1980, 2007.
```

Save your screen changes.

Now start changing the screen zoom settings: Try the 100%, 75%, Auto and Fit options.



What you should notice is that when you use Auto or Fit, that the new group boxes are not visible. This is because the screen size used by the Auto and Fit options is defined by the screen size inside the thin red line.

Initially the area inside the thin red line represents the "5250 size" of this screen.

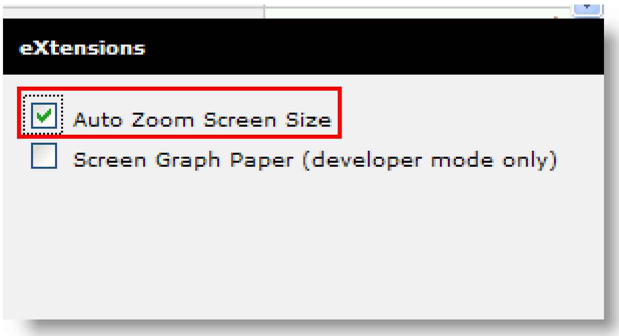
You should never position things outside the thin red line - they will not be visible to people using the Auto or Fit screen zoom options.

However, you can make more space on enhanced 5250 screens.

Set the zoom back to 100% and stretch out the screen so you can see the group boxes again.

Put the MAIN screen in edit mode.

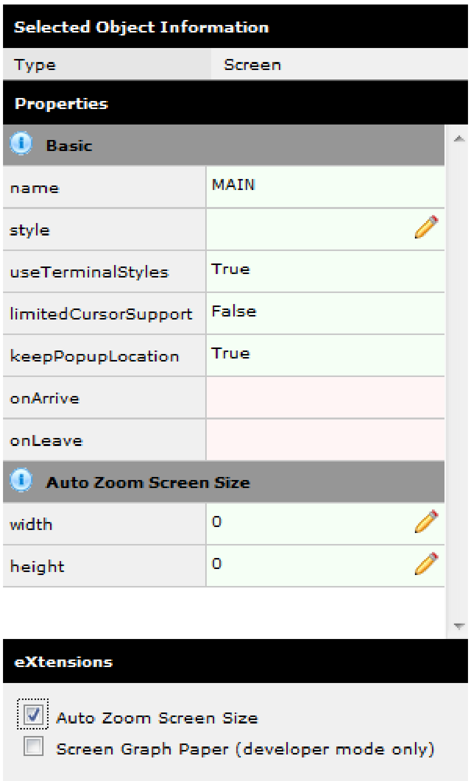
Click anywhere on the screen (except on a screen element) to display your *screen level* eXtensions, then select the ***Auto Zoom Screen Size*** extension in the aXes Designer.



eXtensions

☑ Auto Zoom Screen Size
☐ Screen Graph Paper (developer mode only)

The screen customization window will now look like this:



Selected Object Information

Type	Screen

Properties

Basic	
name	MAIN
style	
useTerminalStyles	True
limitedCursorSupport	False
keepPopupLocation	True
onArrive	
onLeave	

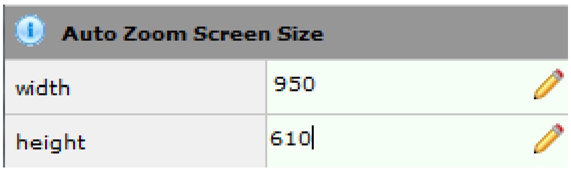
Auto Zoom Screen Size	
width	0
height	0

eXtensions

☑ Auto Zoom Screen Size
☐ Screen Graph Paper (developer mode only)

The width and height properties of the Auto Zoom Screen Size extension allow you to specify the logical dimensions of this enhanced screen. In effect these properties allow you to move the thin red line boundary.

Specify width and height properties, something like this:

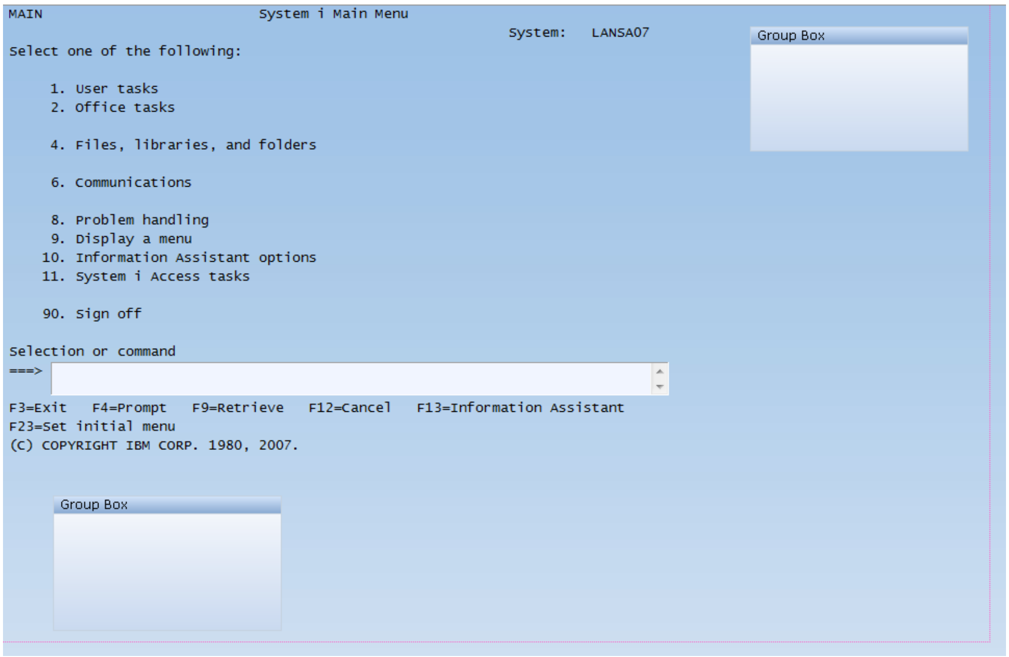


Auto Zoom Screen Size	
width	950
height	610

Content © LANSA Group
All Rights Reserved

Note: These values are logical sizes, in pixels. It does not mean you have to have a screen that wide or high to display the screen. The auto zoom feature will zoom your screen in or out to fit the available browser display space at any time, no matter how large or how small it is.

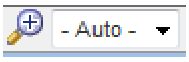
You should be able to move the boundaries out, something like this:



```
MAIN                           System i Main Menu
                                                              System:   LANSA07
Select one of the following:

     1. User tasks
     2. Office tasks

     4. Files, libraries, and folders

     6. Communications

     8. Problem handling
     9. Display a menu
    10. Information Assistant options
    11. System i Access tasks

    90. Sign off

Selection or command
===>

F3=Exit   F4=Prompt   F9=Retrieve   F12=Cancel   F13=Information Assistant
F23=Set initial menu
(C) COPYRIGHT IBM CORP. 1980, 2007.
```

Save your screen changes.

Navigate off this screen and back again to make sure the sizing information is reset, properly.

Now start changing the  screen zoom setting again.

You should also try logging on to aXes as a user and checking your screen there when using various zoom sizes.

You should find that your "big" 5250 screen is correctly visible at all zoom sizes, particularly the Auto and Fit sizes.

There are some important things to note:

1. Do not make the boundaries smaller than the 5250 screen default size. This is not supported in any form. Note that the 5250 screen boundaries will depend on whether the 5250 displayed screen is 24x80 or 27x132.

2. You can make your screen very wide and very high, but as you do this the real 5250 screen area will proportionally shrink. If it shrinks too much users may not be able to read it and the screen caret may stop displaying correctly.

3. Try to avoid a plethora of different "big" screen sizes. Decide on a single "big" screen size, or on a small standard set of them. Use them selectively.

   A simple way to do this is to change the defaults for the Auto Screen Size's height and width properties. Then all a developer needs to do is check the Auto Screen Size eXtension to get the correct default "big" screen size.

   Alternatively, the Auto Zoom Screen Size property values could be set by invoking a USERENV function like USERENV.iscreenHeight("medium"), USERENV.iscreenHeight("large") and USERENV.iscreenHeight("xlarge"). This would allow 3 "big" screen sizes to be centrally defined.

Using wide screens (132 x 27)

When you have 5250 wide screens you should design and execute your applications in wide screen mode.

In effect, any 24x80 mode of operation that your application may support would no longer be used when it is displayed by aXes.

Using Extended/Enhanced 5250 DDS attributes

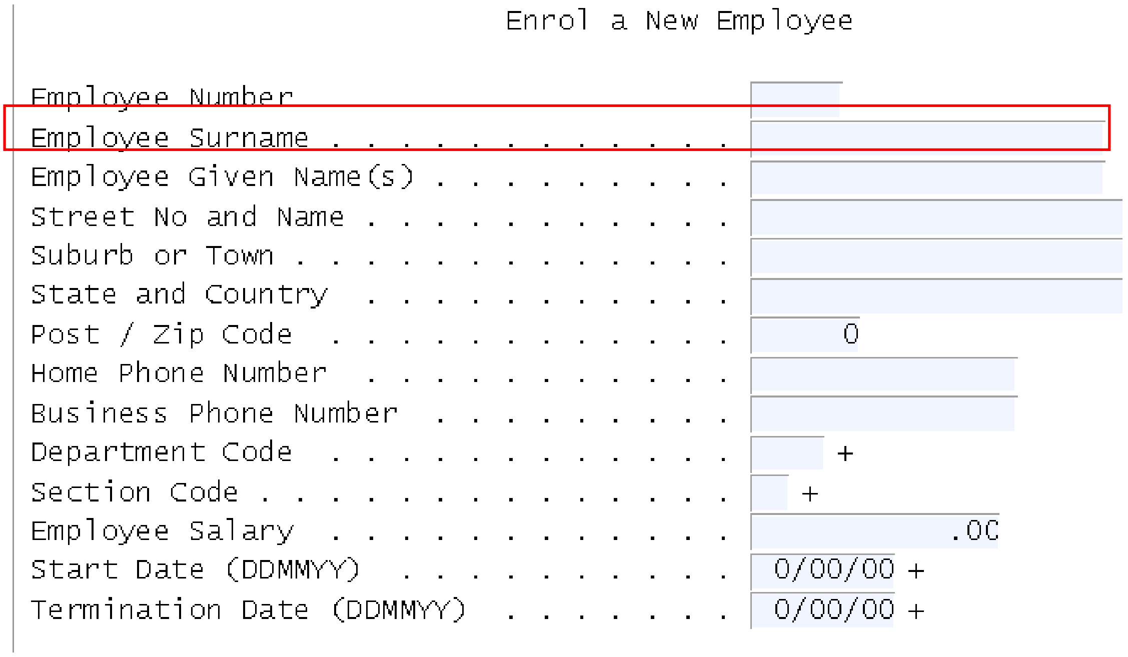
5250 DDS sometimes contain extended/enhanced DDS attributes to display fields as radio buttons, drop downs, etc in the limited range of products that support the options.

aXes supports these extended/enhanced DDS options - but you should not add eXtensions to such screens as the GUI capabilities are different and contradictory.

Leave the screens to be displayed natively by aXes, but do not add eXtensions to them.

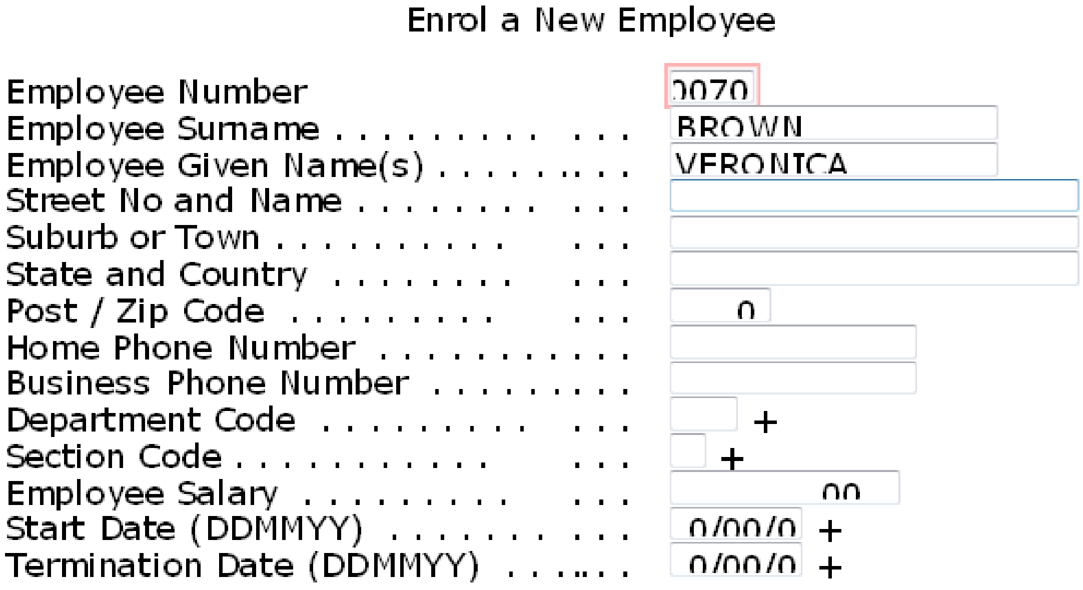
Changing 5250 Row / Column Size

By default rows in AXES 5250 screens will be 15 pixel high and columns will be 8 pixel wide.



```
                       Enrol a New Employee

Employee Number
Employee Surname . . . . . . . . . . . .
Employee Given Name(s) . . . . . . . . .
Street No and Name . . . . . . . . . . .
Suburb or Town . . . . . . . . . . . . .
State and Country  . . . . . . . . . . .
Post / Zip Code  . . . . . . . . . . . .         0
Home Phone Number  . . . . . . . . . . .
Business Phone Number  . . . . . . . . .
Department Code  . . . . . . . . . . . .          +
Section Code . . . . . . . . . . . . . .       +
Employee Salary  . . . . . . . . . . . .                 .0C
Start Date (DDMMYY)  . . . . . . . . . .   0/00/00 +
Termination Date (DDMMYY)  . . . . . . .   0/00/00 +
```

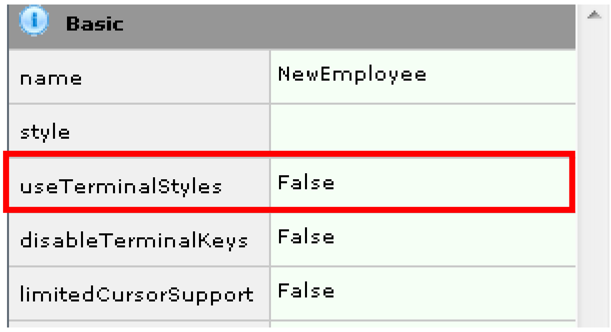
If for some reason you decided to use larger font and changed the global style font to Verdana, 12pt, your screen will look similar to the one below:



Enrol a New Employee

Employee Number	0070
Employee Surname	BROWN
Employee Given Name(s)	VERONICA
Street No and Name	
Suburb or Town	
State and Country	
Post / Zip Code	0
Home Phone Number	
Business Phone Number	
Department Code	+
Section Code	+
Employee Salary	00
Start Date (DDMMYY)	0/00/0 +
Termination Date (DDMMYY)	0/00/0 +

Reminder: you need to set the **useTerminalStyles** property of this screen to **false** to allow custom styles to take effect.



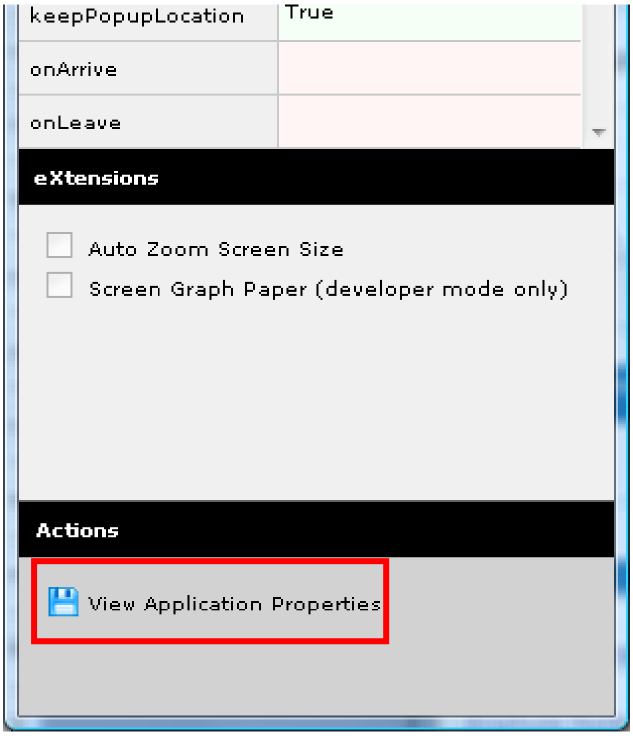
Basic	
name	NewEmployee
style	
useTerminalStyles	False
disableTerminalKeys	False
limitedCursorSupport	False

You can see in the screenshot above that the screen looks too packed and the texts are too big for the edit boxes. One obvious but tedious way to fix this is by resizing each textbox and reposition them.

A better alternative is to use an application-level eXtension Application Terminal . This eXtension allows you to easily change the default height of rows in 5250 screens.

Follow these steps to use this eXtension:

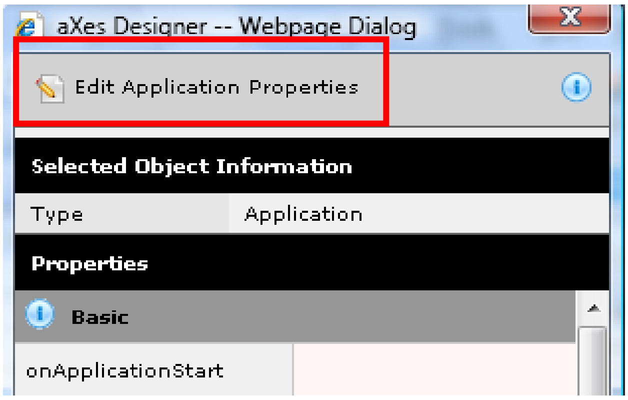
- Go to the **aXes Designer** window.
- Ensure the screen is not in edit mode.
- Switch to **Application Properties** view by clicking on the **View Application Properties** button at the bottom of the window.



keepPopupLocation	True
onArrive	
onLeave	

eXtensions
□ Auto Zoom Screen Size
□ Screen Graph Paper (developer mode only)
Actions
View Application Properties

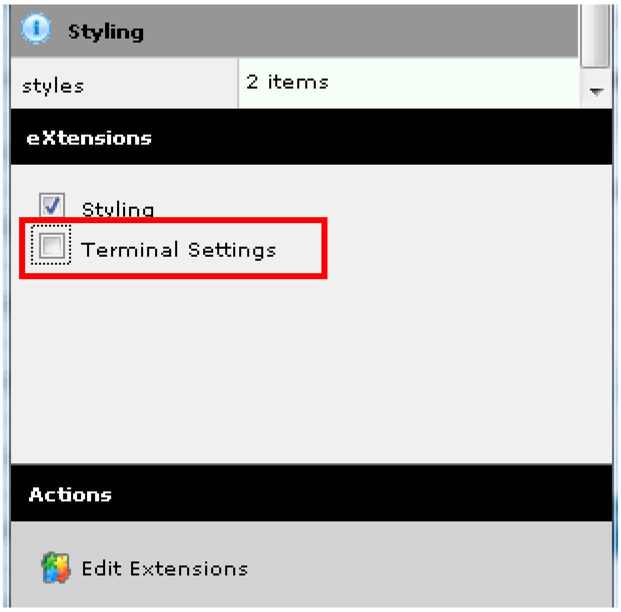
- Switch to **edit mode** by clicking the **Edit Application Properties** button at the top of the window.



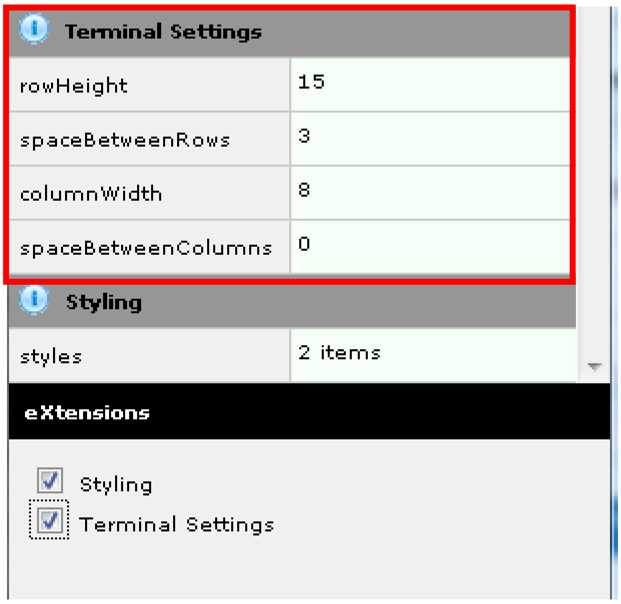
Type	Application

aXes Designer -- Webpage Dialog
Edit Application Properties
Selected Object Information
Properties
Basic
onApplicationStart

- If you have never customized a screen before, the Application Terminal Settings eXtension will have not been ticked (AXES automatically includes this eXtension the first time you save a screen customization), otherwise you see it ticked already.



If this is the case, tick the **Terminal Settings** eXtension to include it.
Now you see the **Terminal Settings** properties in the property sheet.


Terminal Settings

rowHeight	15
spaceBetweenRows	3
columnWidth	8
spaceBetweenColumns	0

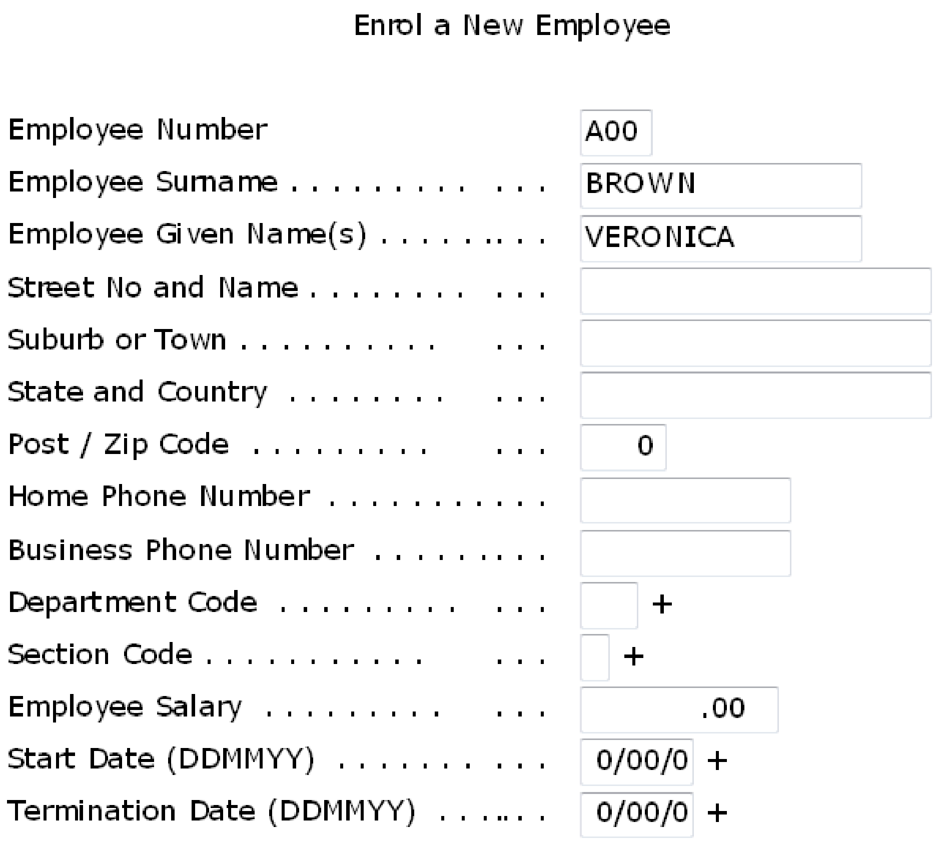
Styling

styles	2 items

eXtensions

☑ Styling
☑ Terminal Settings

Set the **rowHeight** to 25 and **spaceBetweenRows** to 5. Save the settings (click the **Save** button at the top of the window). You will see your screen changes immediately to the one below:



Enrol a New Employee

Employee Number	A00
Employee Surname	BROWN
Employee Given Name(s)	VERONICA
Street No and Name	
Suburb or Town	
State and Country	
Post / Zip Code	0
Home Phone Number	
Business Phone Number	
Department Code	+
Section Code	+
Employee Salary	.00
Start Date (DDMMYY)	0/00/0 +
Termination Date (DDMMYY)	0/00/0 +

About Fonts and Font Sizes

We recommend that you do not change fonts or font sizes via shipped style sheets or use customized style sheets to change fonts or font sizes.

If you change a style sheet you may need to clear your browser cache to get the changes.

We also recommend that you do not use variable width fonts on un-customized screens because somewhere down the track you will encounter a problem on a screen with subfile content or headings that appear to become misaligned.

- In Arial, compare the widths:

  WWWWW
  IIIII

  The string of W's is almost 3 times as wide as the I's.

- In Lucida Console, compare the widths:

  ```
  WWWWW
  IIIII
  ```

  They are the same width.
  This is the essence of the variable width/pitch font problem.

- If an RPG program puts out these subfile headings as a *single* text output field:

  ```
  Customer Product     Order    Quantity    Value
  18181818 6373737     8484848       700    1,234.56
  18181818 6373737     8484848       700    1,234.56
  ```

  You may end up with a display like this, when using a variable width font:

  ```
   Customer Product    Order   Quantity   Value
  18181818 6373737     8484848       700    1,234.56
  18181818 6373737     8484848       700    1,234.56
  ```

  There is no resolution here. The 5250 DDS output strings ...

  ```
  'Customer Product     Order    Quantity    Value' (in Lucida Console)
  ```

  'Customer Product   Order   Quantity   Value' (in Arial)

  are exactly the same length – they simply have different display lengths.

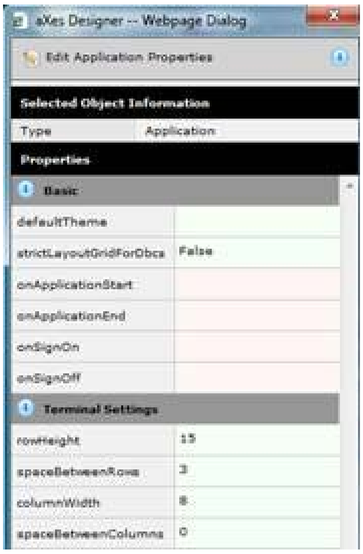
  If the program also outputs the subfile data columns as one long field instead of as individual fields (as some do) then the problem is even more exacerbated:

  ```
  Customer Product     Order    Quantity       Value
  1123     BOLTS       8484848         3        4.56
  1123     NUTS        8484848       700    1,234.56
  1123     HAMMER      8272         2000   11,234.56
  Customer Product     Order    Quantity       Value
  1123   BOLTS    8484848   3   4.56
  1123   NUTS    8484848   700   1,234.56
  1123   HAMMER    8272   2000   11,234.56
  ```

- To change fonts or font sizes use the application definition eXtension. See Tutorial 3 – Setting up your Styles and also Tutorial 10 – 5250 screen styling. Changes you make to fonts and font sizes here may be applied to customized and non-customized screens.

- A 5250 screen is inherently a fixed width device which uses fixed width characters. A 5250 screen has 24 x 80 or 27 x 132 fixed width and height cells that may contain a character. Starting to use variable width characters in this model is always problematic and needs to be treated with care. aXes uses specific dimensions that indicate how wide and high a 5250 screen "cell" is when it has to perform screen layout calculations.

- If you use a large font, or a font that is wider, you may need to adjust these cell sizes. This is done here:



Selected Object Information

Type	Application

Properties

Basic

defaultTheme	
strictLayoutGridForObcs	False
onApplicationStart	
onApplicationEnd	
onSignOn	
onSignOff	

Terminal Settings

rowHeight	15
spaceBetweenRows	3
columnWidth	8
spaceBetweenColumns	0

If you change these values you need to check several customized and un-customized 5250 screens to see what impact your changes have made and how they operate in conjunction with your font and font size.

- These cell size changes impact customized and non-customized screens. You should set your font and font size early and adjust the screen cell sizes accordingly. Changes made to these values when a project is in progress can impact the layout of screens already customized.

5250 Attributes Bytes

The color and visual characteristics of fields on 5250 screens are controlled by *attribute bytes*. In a 5250 data stream the attribute byte (8 bits) precedes the field on the 5250 screen. An attribute byte will have one of these hexadecimal values:

```
x20 Green
x21 Green/Reverse Image
x22 White
x23 White/Reverse Image
x24 Green/Underscore
x25 Green/Underscore/Reverse Image
x26 White/Underscore
x27 Nondisplay
x28 Red
x29 Red/Reverse Image
x2A Red/Blink
x2B Red/Reverse Image/Blink
x2C Red/Underscore
x2D Red/Underscore/Reverse Image
x2E Red/Underscore/Blink
x2F Nondisplay
x30 Turquoise/Column Separators
x31 Turquoise/Column Separators/Reverse Image
x32 Yellow/Column Separators
x33 Yellow/Column Separators/Reverse Image
x34 Turquoise/Underscore
x35 Turquoise/Underscore/Reverse Image
x36 Yellow/Underscore
x37 Nondisplay
x38 Pink
x39 Pink/Reverse Image
x3A Blue
x3B Blue/Reverse Image
x3C Pink/Underscore
x3D Pink/Underscore/Reverse Image
x3E Blue/Underscore
x3F Nondisplay
```

When you use display file DDS keywords like underline - DSPATR(UL), reverse video - DSPATR(RI), or blue - COLOR(BLU) - you are telling the DDS display file compiler what attribute byte to put in front of the field on the 5250 screen.

5250 Attributes Bytes on Un-Customized 5250 screens

Note: This material applies to aXes-TS only.

When you display an un-customized screen the 5250 attribute bytes are mapped to web page characteristics by using a *cascading style sheet*, defined in a CSS document.

There is a cascading style sheet for each aXes theme.

For example - if you open *axes\ts\skins\axes_blue.css* with NOTEPAD you will see the style sheet used for the aXes blue theme.

If you search for X28 (the attribute byte for color RED) you will find the style details that define how a 5250 screen field with a X28 attribute byte is to be displayed by the web browser:

```
.x28, .Infield .x28
{
    color: #cc0000 ;
}
```

This CSS definition says that the browser color #CC0000 🟥 should be used for text in this field.

> Note: Detailing the complete use of low level CSS attributes is beyond the scope of this tutorial. Extensive information is available from the W3C Schools link on the aXes Projects Home page.

The key things to understand about uncustomized screens and 5250 attribute bytes are:

- Web browsers have no implicit understanding of 5250 attribute byte X28. What actually happens is that attribute byte X28 is ***mapped*** to a style that specifies the web browser characteristics to be used.
- 5250 colors are quite stark. Often the style sheets are used to map them to softer colors since the web browser has many more color possibilities.
- Sometimes several 5250 colors are mapped to the same web color. For example, the 5250 colors blue and green may both map to web color black – often because it is unusual to find basic Windows GUI forms that make use of multi-colored text lines in the same way that 5250 screens do.
- If you change an aXes shipped style sheet (CSS document) you should back up and also version your changes so that you can back out unwanted changes and won’t lose your changes when they are overwritten in an aXes upgrade.

5250 Attributes Bytes on Customized 5250 screens

When you customize a 5250 screen the CSS style sheets mentioned in the preceding section are normally not applied to the screen.

By customizing a screen you are indicating your intention to make it into a Windows style GUI form - often quite different to the original 5250 screen – so basic 5250 attribute bytes do not and cannot be applied in the same way.

The appearance of a customized screen is controlled purely by styles. Generally the use of styles and themes has many advantages over using 5250 attribute bytes.

However - for customized screens it is still possible to supply an attribute byte to style mapping - which will be automatically applied when it is possible to do so.

The process for doing this goes like this:

- You define a normal eXtension style (as described in the preceding section).
- You use the for5250Attributes property of the eXtension style to associate the style with ***one or more*** 5250 attribute bytes - in either an input and/or output field context.
- When a 5250 field is presented on a customized form its attribute byte is used to decide whether an eXtension style should be automatically applied.

There are some key things to know about this process:

1. The automatic application of a default style only makes sense when the 5250 field is still visualized like a 5250 field. Applying the style to a field that is visualized as something very different (eg: a drop down, radio buttons, a push button) is not possible.

2. When a brand new element is added to a customized 5250 screen it has no 5250 attribute byte – so it is not possible to apply a style automatically. However - you can specify the style yourself, reusing one of the 5250 ones if you like.

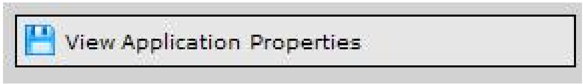
3. You can theme these styles just like any other style. So 5250 attribute COLOR(GRN) may automatically be interpreted differently if you are using a theme of blue or olive (say).

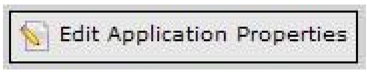
One of the key uses of this 5250 attribute mapping feature is in the handling of DSPATR(RI) – reverse video – which is often used to indicate an error state for a 5250 field.

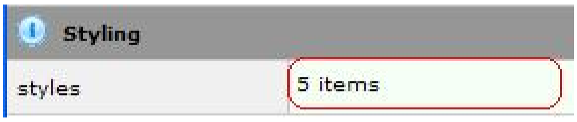
> Note: There is no such thing as an “error state” for a 5250 field. By convention many sites use DSPATR(RI) - but some sites alter a field’s color instead.

Here is an example of using this 5250 attribute byte mapping feature to handle DSPATR(RI) for error displays:

1. Open the styles property of the application:

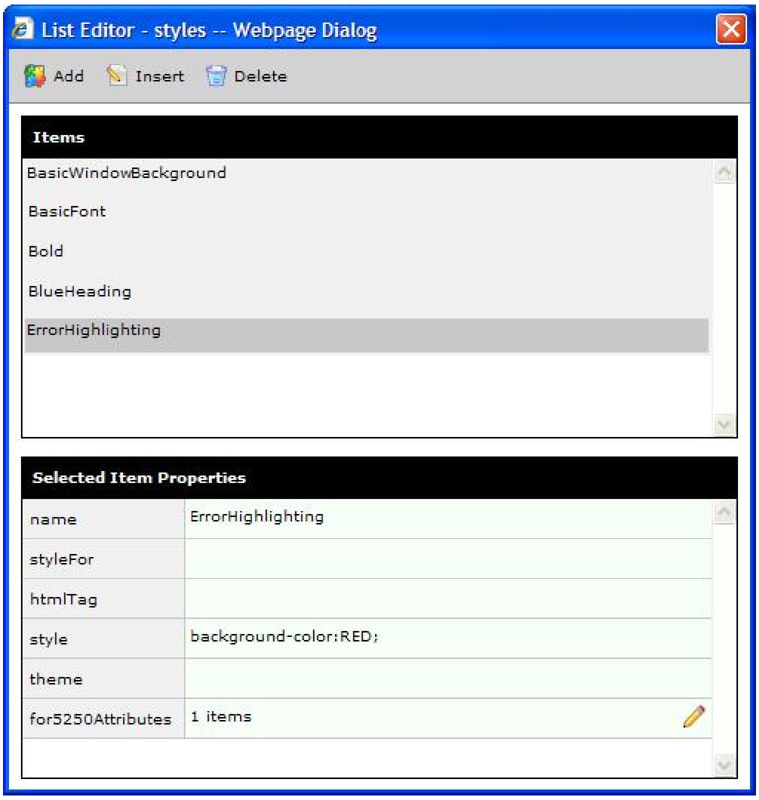






styles: 5 items

2. Create a style called ErrorHighlighting as shown:



Items

BasicWindowBackground
BasicFont
Bold
BlueHeading
ErrorHighlighting

Selected Item Properties

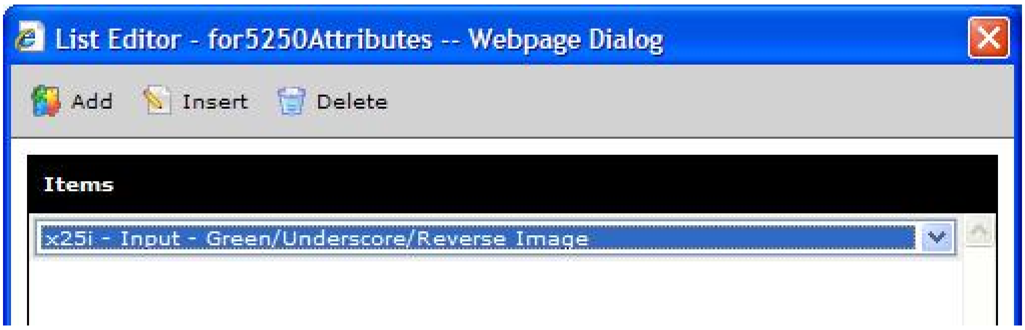
name	ErrorHighlighting
styleFor	
htmlTag	
style	background-color:RED;
theme	
for5250Attributes	1 items

3. Click on the for5250Attributes property.

Add an item. Make it display attribute 25 - for an input field

(x25i - Input - Green/Underscore/Reverse Image)

This display attribute is commonly used by applications to indicate that a field is in error

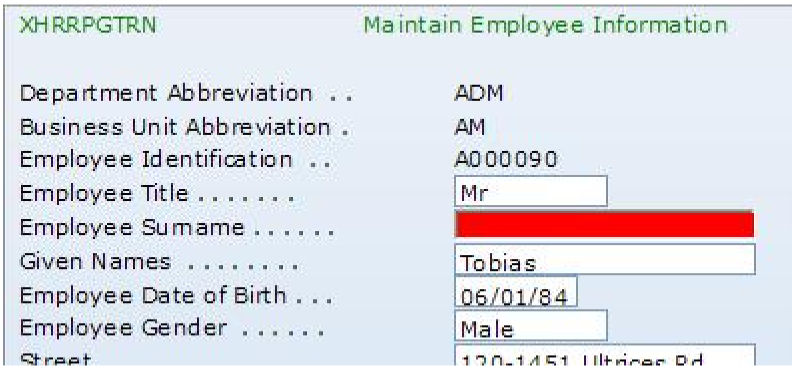


x25i - Input - Green/Underscore/Reverse Image

4. Exit the style properties and save Application properties. You are now ready to test your change.

5. If you run the Axes demo program using commands ADDLIBLE AXESDEMO and then CALL XHRRPGTRN.

Select an employee, and edit their details, blank out the surname and press enter, you should see the field in error highlighted in red.



XHRRPGTRN Maintain Employee Information

Department Abbreviation . . ADM
Business Unit Abbreviation . AM
Employee Identification . . A000090
Employee Title . . . . Mr
Employee Surname . . . . . .
Given Names . . . . Tobias
Employee Date of Birth . . . 06/01/84
Employee Gender . . . . . . Male
Street 120-1451 Ultrices Rd

Now trying editing the ErrorHighlighting style.

Change the background color to Yellow an save your changes.

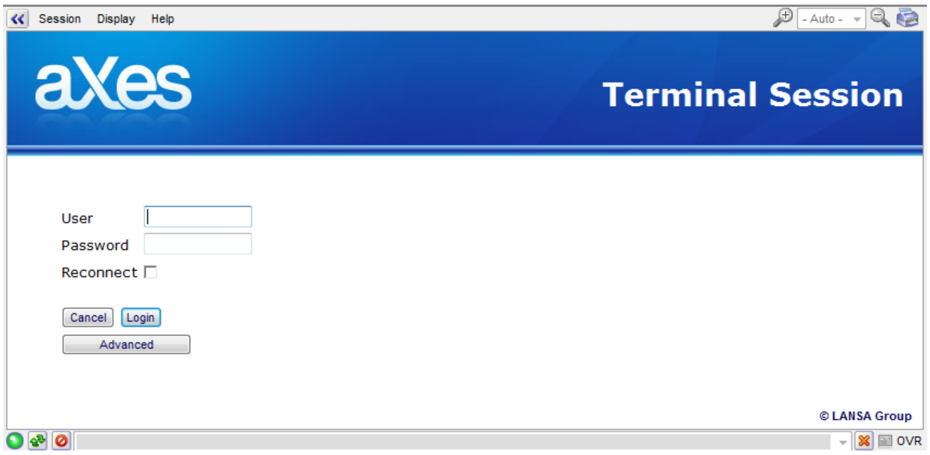
Restart the 5250 application again.

The background color should have changed from Red to Yellow.

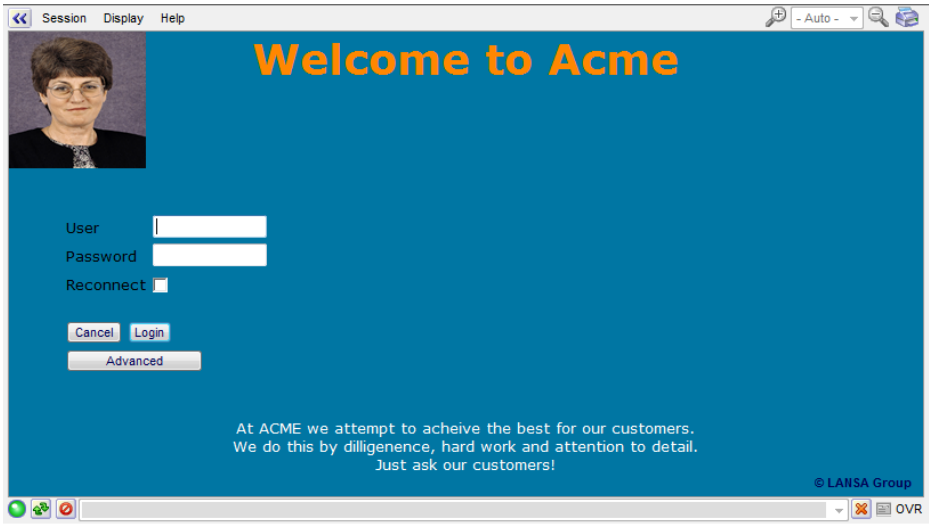
Customizing the aXes Logon Screen in aXes-TS

Note that in this version of aXes the following section applies to aXes-TS only, not to aXes-TS2. See Customizing the aXes Logon Screen in aXes-TS2 for details on customizing the logon screen in aXes-TS2.

Starting Point – The Shipped Logon Screen



Result – Your Customized Logon Screen



At ACME we attempt to acheive the best for our customers.
We do this by dilligenence, hard work and attention to detail.
Just ask our customers!

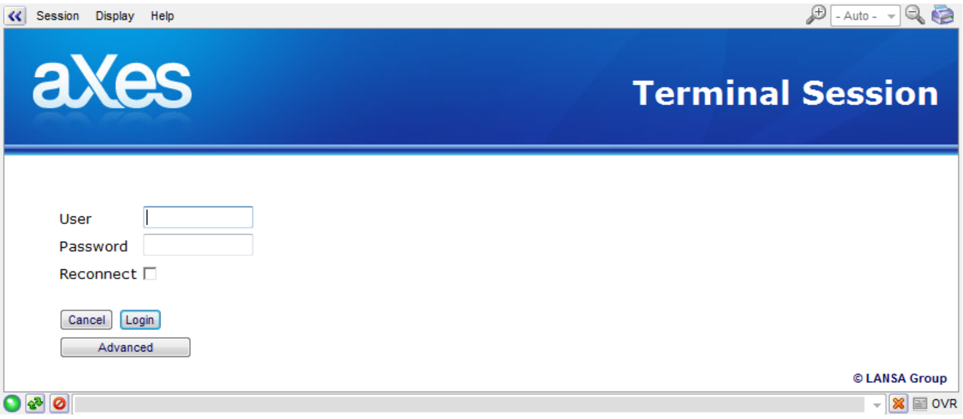
Visually this example is not pretty - but structurally it has the classic elements that customers often desire – an image at the top left, a title/heading area and some additional details displayed as a footer. For the best visual results we recommend that you consult with a professional graphic designer.

Copy the Shipped Logon Screen to Make your own Version

In the aXes ts/skins folder locate file axes_logon.xml.
Copy it to produce a custom login file named ***MyTest_login.xml*** (say) in the same folder.
Using the IBM i WRKLNK command check that user *PUBLIC has *R rights only to the new file MyTest_login.xml.

Set Up a Desktop Short Cut to Test Your Custom Sign On Screen

Set up a desktop short cut to test your custom logon. The URL it uses should be like this:
Error! Hyperlink reference not valid.
This will start an aXes basic terminal session using your custom log in screen instead of the standard shipped one.
Test your desktop shortcut. The result should look like this (ie: exactly like this shipped log on):



Have a quick look at the structure of MyTest_Logon.XML

Open your MyTest_Logon.XML document with a source editor, like NOTEPAD:
Structurally it is an XML document that is transformed to create a HTML page.
The HTML page is very simple, consisting of

- a set of styles – defined immediately after this line:

```
<style type="text/css">
```

- Some simple JavaScript – defined immediately after these lines:

```
<script>
  function advancedLogin ()
```

- A simple web form – defined immediately after this line:

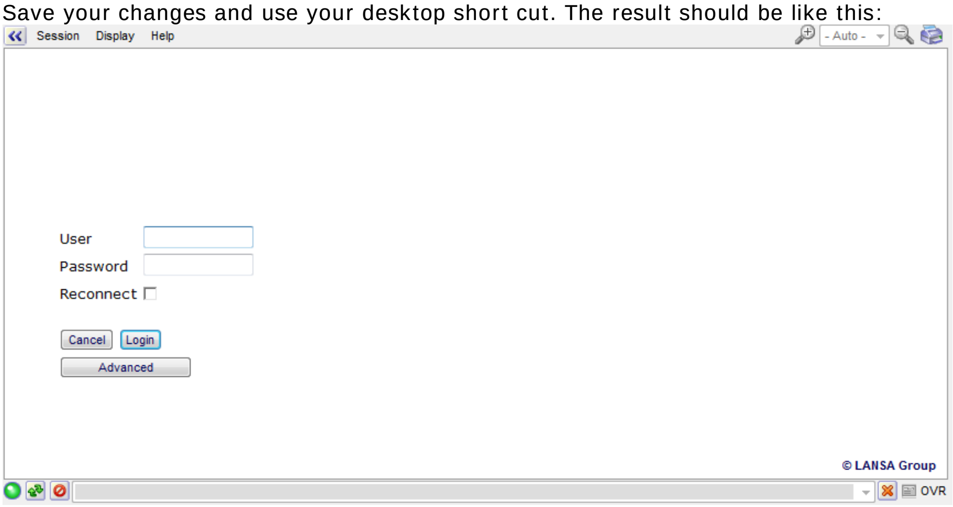
```
<div id="loginMainDIV" style="width:100%;height:100%;background: white;">
```

Remove the Shipped Images

The first thing to do is strip out the images used in the shipped header, by ***removing*** these lines:

```
<div style="position:absolute; width:100%; top:0; left:0; height:115px; background: white
url(/ts/skins/images/axnewheaderbg.jpg) repeat-x left top;">
  </div>
  <div style="position:absolute; top:0; left:0; height:115px; background: transparent
url(/ts/skins/images/axnewheader.jpg) no-repeat left top;">
   <table style="width:100%; height:115px;">
      <tr>
        <td style="width:100%;"></td>
        <td ax_txtscope="core" ax_txt="Terminal Session" nowrap="true" style="padding-
right:15px;" id="newLogoTitle">Terminal Session</td></tr>
    </table>

  </div>
```

Save your changes and use your desktop short cut. The result should be like this:



Add in your own Styles

Set up some styles to define the various parts of your custom logon, by inserting these new style elements just ***before*** the </style> tag:

```
  #custom_LoginHeader
  {
  top : 0;
  left: 0;
  width: 100% ;
  position: absolute;
  text-align: center;
  color: #ff8700;
  font-size: 28pt ;
  font-weight: bold ;
  font-family:Verdana;
  }

  #custom_LoginFooter
  {
  top : 340;
  left: 0;
  width: 100% ;
  position: absolute;
  text-align: center;
  color: white;
  font-size: 10pt ;
  font-weight: normal;
  font-family:Verdana;
  }
```

Save your changes and use your desktop short cut to check that your custom logon still works okay.

Add in your own Header Area

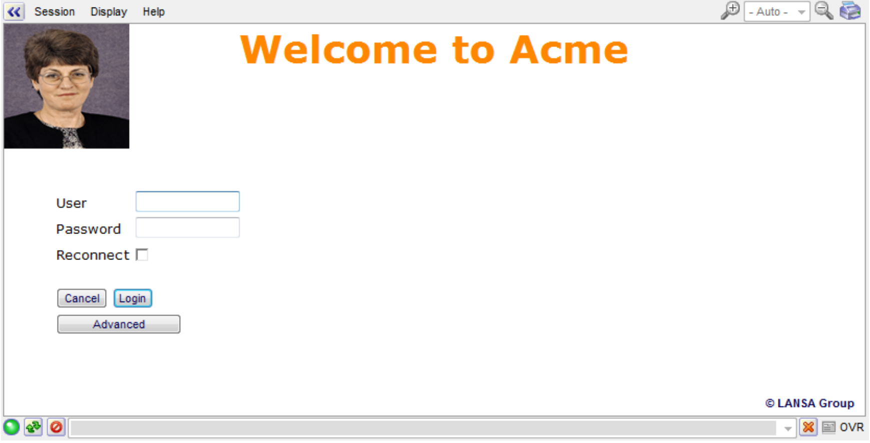
Immediately ***after*** this tag:

```
<div id="loginMainDIV" style="width:100%;height:100%;background: white;">
```

Insert these new lines to insert your custom header area:

```
  <div id="custom_LoginHeader">
    <img src="/ts/skins/images/examplephoto.gif"
style="height:120px;width:120px;position:absolute;left:0px;"></img>
    <span>Welcome to Acme</span>
  </div>
```

Save your changes and use your desktop short cut. The result should be like this:



Change the Background Colour

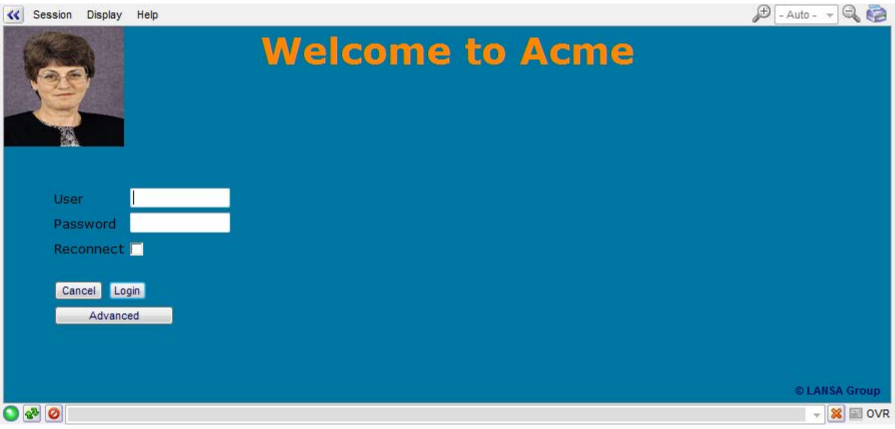
Now change this line:

```
<div id="loginMainDIV" style="width:100%;height:100%;background: white;">
```

To This:

```
<div id="loginMainDIV" style="width:100%;height:100%;background: #0076a3;">
```

Save your changes and use your desktop short cut. The result should be like this:



Add in your own Footer Area

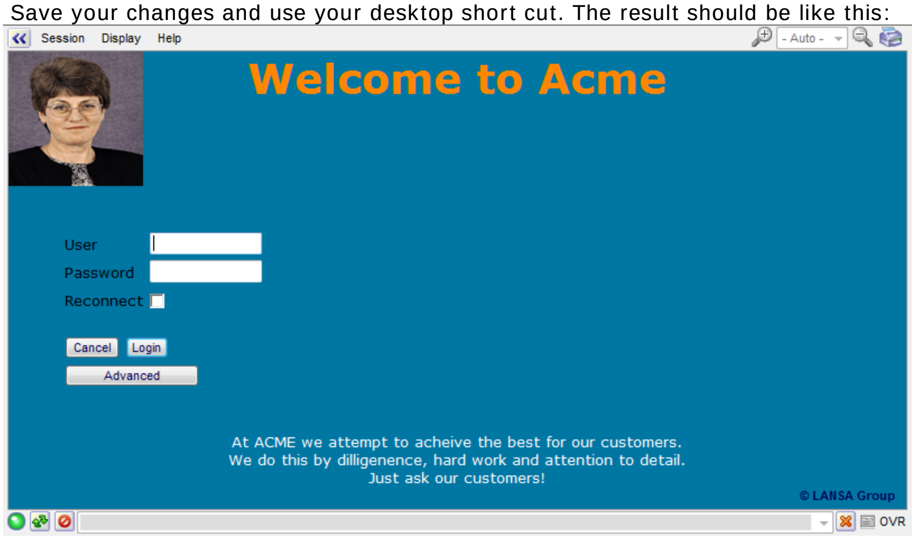
Immediately ***before*** this tag in:

```
  <div id="bylansa">&#169; LANSA Group</div>
```

Insert these new lines to insert your custom footer area:

```
<div id="custom_LoginFooter">
    <div>At ACME we attempt to achieve the best for our customers.</div>
    <div>We do this by diligence, hard work and attention to detail.</div>
    <div>Just ask our customers!</div>
</div>
```

Save your changes and use your desktop short cut. The result should be like this:



At ACME we attempt to acheive the best for our customers.
We do this by diligence, hard work and attention to detail.
Just ask our customers!

Now do whatever you like

This example should give you enough basic information to create what ever look you desire in your own custom logon screen. All you have to do is make sure to test your changes. For the best visual results we strongly recommend that you consult with a professional graphic designer.

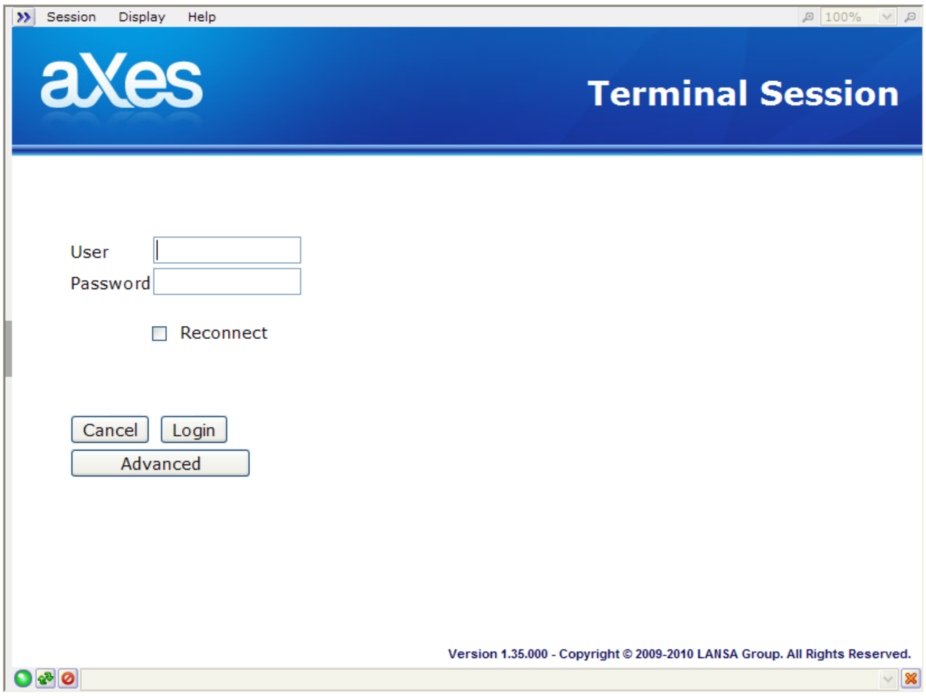
Don't Forget to Back Up your Work

Remember to back up your custom logon XML document.

Customizing the aXes Logon Screen in aXes-TS2

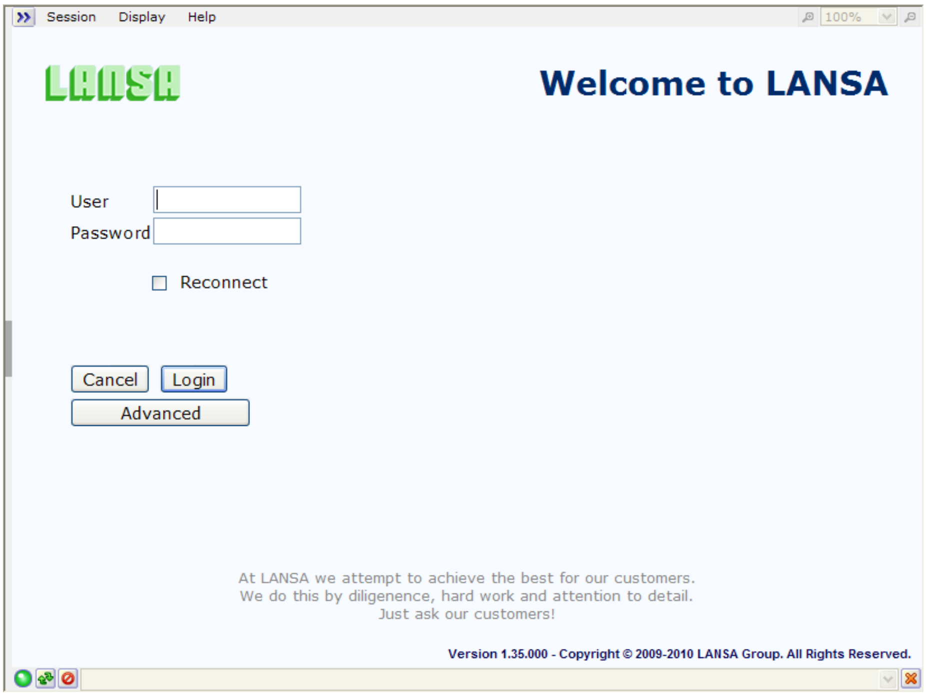
Note that the following section applies to aXes-TS2 only.

Starting Point – The Shipped Logon Screen



Version 1.35.000 - Copyright © 2009-2010 LANSA Group. All Rights Reserved.

Result – Your Customized Logon Screen



At LANSA we attempt to achieve the best for our customers.
We do this by diligenence, hard work and attention to detail.
Just ask our customers!

Version 1.35.000 - Copyright © 2009-2010 LANSA Group. All Rights Reserved.

Visually this example is not pretty - but structurally it has the classic elements that customers often desire – an image at the top left, a title/heading area and some additional details displayed as a footer. For the best visual results we recommend that you consult with a professional graphic designer.

Copy the Shipped Logon On Screen to Make your own Version

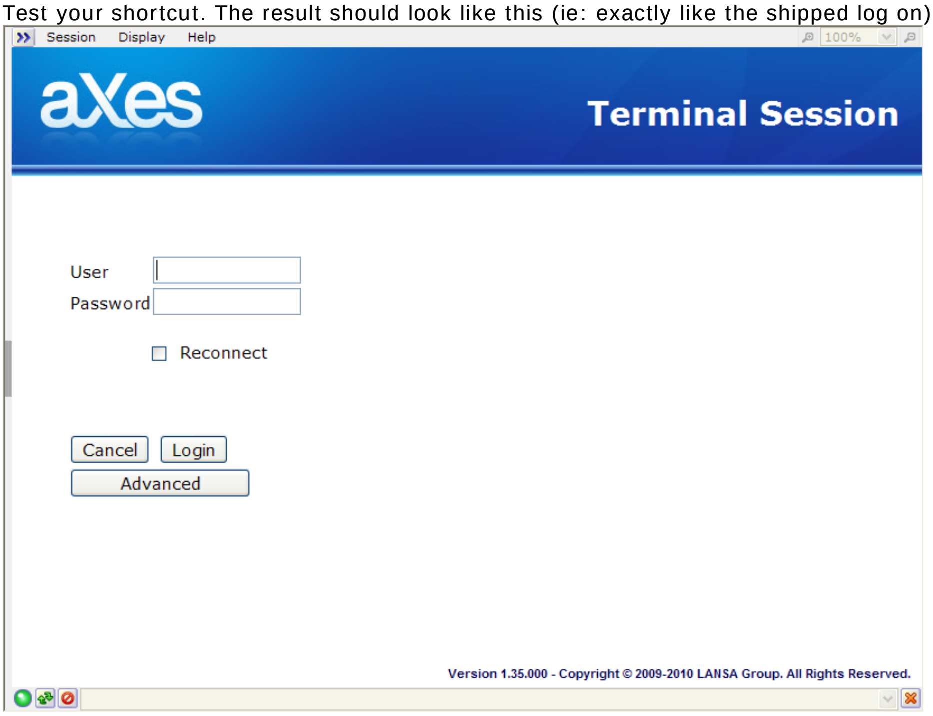
In the aXes ts/ts2 folder locate the files login.html and login.css. Copy them to produce custom login files named ***MyTest_login.html*** and ***MyTest_login.css*** (say) in the same folder. *Note: it is important that the left hand part of the name (before the extension) is the same for both files*. Using the IBM i WRKLNK command check that user *PUBLIC has *R rights only to the new files.

Set Up a Desktop Short Cut to Test Your Custom Sign On Screen

Set up a desktop short cut or browser bookmark to test your custom logon. The URL it uses should be like this:

http://<aXesHost>:<aXesPort>/ts/ts2/index.html?login=mytest_login

This will start an aXes terminal session using your custom log in screen instead of the standard shipped one. Test your shortcut. The result should look like this (ie: exactly like the shipped log on):



Have a quick look at the structure of MyTest_Login.html

Open your MyTest_Logon.html document with a source editor, for example NOTEPAD.

The important thing to note about this file which is different from most other HTML files you will see is that its <head> section is empty. This file is loaded by aXes and its <body> is inserted into the existing HTML document. When doing this, some browsers will ignore the content of the <head> section. Others will attempt to process it in inconsistent ways. So, for a good cross-browser experience, leave the <head> section empty.

The first thing you will see is a <script> section containing some JavaScript. This JavaScript hooks up the fields in the HTML with the corresponding fields in the aXes virtual terminal and provides various other user interface functionality. Getting into the details of what is going on here is beyond the scope of this tutorial.

The <script> is then followed by the HTML required to build the logon page. The part that is of interest to us are the first few lines:

```
<div id="loginPage">

<div id="loginLogo">
  <div id="loginTitleText" ax_txtscope="core" ax_txt="Terminal Session"></div>
</div>
```

Now have a look at MyTest_Login.css to see how these elements have been styled:

```
#loginPage {
    width:100%;
    height: 100%;
    position: absolute;
    top: 0px;
    left: 0px;
    background: transparent url(css/images/axnewheaderbg.jpg) repeat-x left top;
}
#loginLogo {
    margin-bottom: 70px;
    display: block;
    height: 115px;
    background: transparent url(css/images/axnewheader.jpg) no-repeat left top;
    text-align: right;
}
#loginTitleText {
    font-weight: bold;
    color: white;
    font-family: Verdana;
    font-size: 22pt;
    white-space: nowrap;
    margin-right: 20px;
    display: inline;
    line-height: 115px;
}
```

The first <div> (id="loginPage") wraps all the content of the login page. It has been sized and positioned to fill the entire terminal area and given a background image that repeats horizontally. This creates a blue stripe across the top of the <div>.

The loginLogo <div> has been given a background image that doesn't repeat and has positions in the top,left of the <div>. This is the aXes logo. The <div> has also been given a height to make sure the image is visible and a bottom margin to create a bit of space before the screen fields.

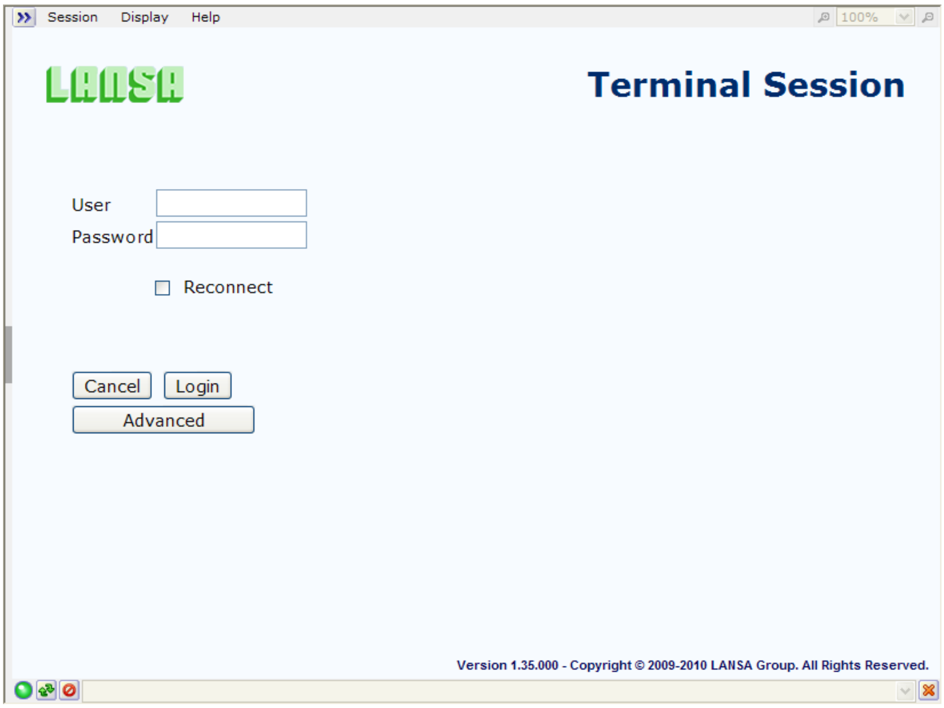
The loginTitleText <div> has been given some appropriate font and color settings. Note the "line-height" property. This is set to the same height as the loginLogo <div> so that the text will center vertically.

Change The Background Images

Change the first 3 style entries as follows:

```
#loginPage {
        width:100%;
        height: 100%;
        position: absolute;
        top: 0px;
        left: 0px;
        background: #f7fbff;
}
#loginLogo {
        margin: 30px;
        margin-bottom: 70px;
        display: block;
        height: 40px;
        background: transparent url(http://www.lansa.com/images/layout/lansa_logo.gif) no-
repeat left center;
        text-align: right;
}
#loginTitleText {
        font-weight: bold;
        color: #002b6b;
        font-family: Verdana;
        font-size: 22pt;
        white-space: nowrap;
        margin-right: 0px;
        display: inline;
        line-height: 40px;
}
```

Save your changes, clear your browser cache and reload the page. The result should be like this:



Changing the Header Text

Take another look at the loginTitleText <div> in MyTest_Login.html:

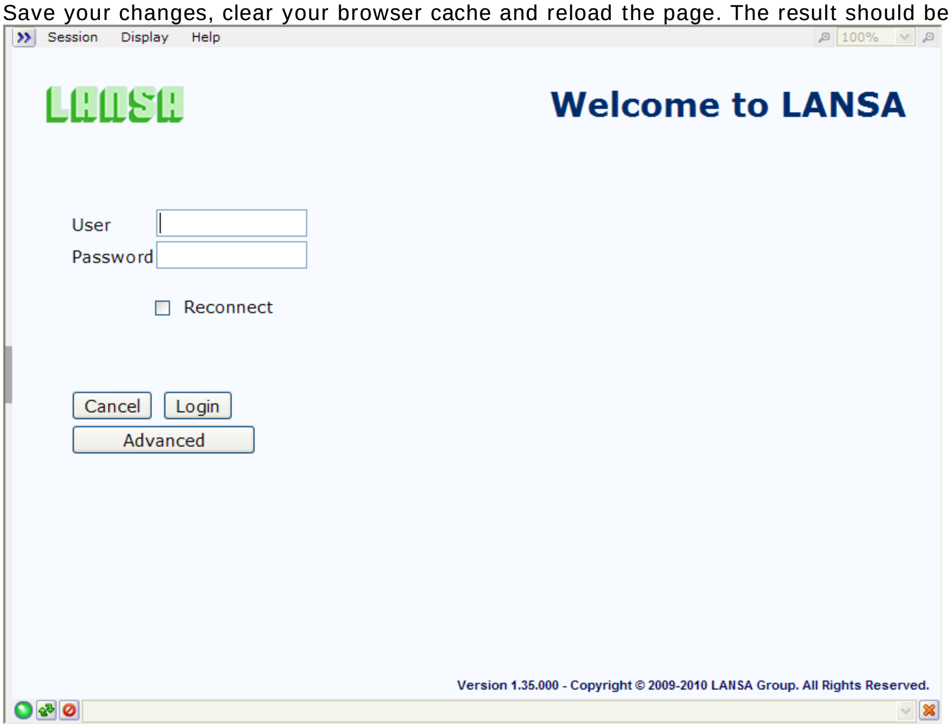
```
<div id="loginTitleText" ax_txtscope="core" ax_txt="Terminal Session"></div>
```

The <div> is empty but the “ax_txt” attribute tells aXes to fill it with the text “Terminal Session” or a localized translation if one is available. If you have your header text available in multiple languages, change “ax_txtscope” value to “cust” and the “ax_txt” value to the key for your string. See Tutorial 9 for more details on setting up translation files.

For the purposes of this tutorial, we will insert a fixed string:

```
<div id="loginTitleText">Welcome to LANSA</div>
```

Save your changes, clear your browser cache and reload the page. The result should be like this:



Add in your own Footer Area

Immediately ***before*** this tag in MyTest_Login.html:

```
  <div class="lansaGroupCopyright">
```

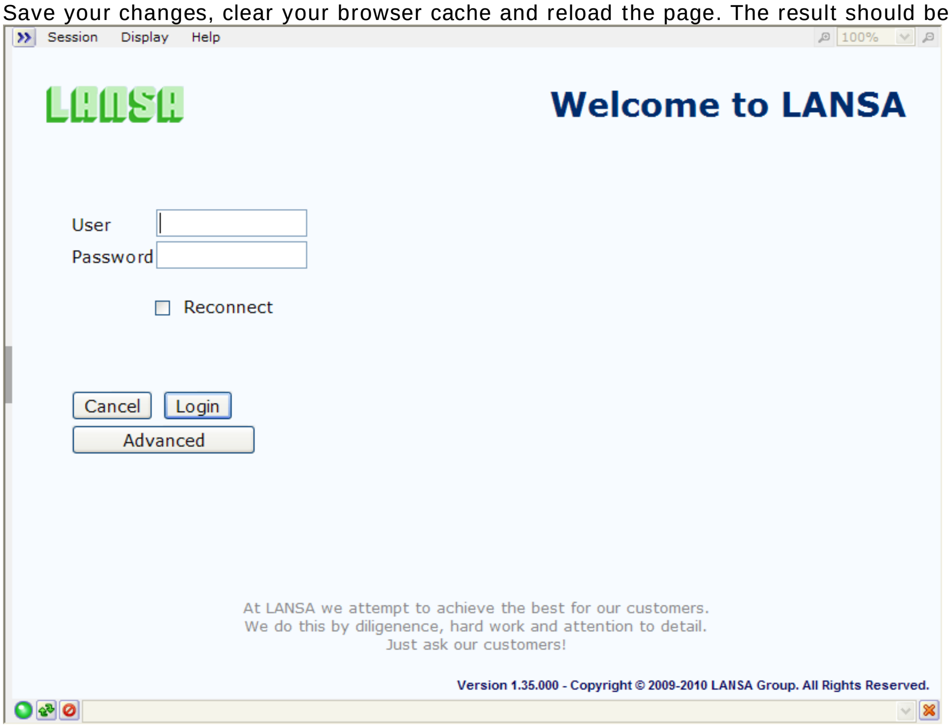
Insert these new lines to add your custom footer area:

```
<div id="custom_LoginFooter">
At LANSA we attempt to achieve the best for our customers.<br />
We do this by diligence, hard work and attention to detail.<br />
Just ask our customers!
</div>
```

Add the following declaration to MyTest_Login.css:

```
#custom_LoginFooter {
    position: absolute;
    bottom: 40px;
    width: 100%;
    text-align: center;
    color: #868686;
    font-size: 10pt ;
    font-weight: normal;
    font-family: Verdana;
}
```

Save your changes, clear your browser cache and reload the page. The result should be like this:



At LANSA we attempt to achieve the best for our customers.
We do this by diligence, hard work and attention to detail.
Just ask our customers!

Version 1.35.000 - Copyright © 2009-2010 LANSA Group. All Rights Reserved.

Now do whatever you like

This example should give you enough basic information to create the look you want in your own custom logon screen. All you have to do is make sure to test your changes. For the best visual results we strongly recommend that you consult with a professional graphic designer.

Don’t Forget to Back Up your Work

Remember to back up your custom logon documents.

aXes Documentation Library

eXtensions Tutorial 11 - FAQ and Examples

Assumed Knowledge Level

The following material assumes the reader has completed at least Tutorials 0 - 4.

FAQ – Configuration and Standards

Should I give each developer their own Definition Set / Project Folder?

For training and experimentation purposes you might give each developer their own private definition set / project folder. *When doing a real project you should* *never* *do this*. Projects should be set up on a discrete project basis. Work done in a definition set / project folder cannot be merged with work done in another definition set / project folder. It is normal for multiple developers to be working on the same project with the same definition set.

Can a screen's customization be completely removed?

Yes. Locate the file named Screen_xxxxxxxxxx.js in your definition set folder and delete it.

Before doing this end all aXes developer sessions.

Can a screen's customization be reverted to an earlier version?

Yes.

All the previously saved versions of screen xxxxxxxxxxx are stored in your ScreenVersions definition set folder's subfolder with names like Screen_xxxxxxxxxx_YYYYMMDD_HHMMSS_mmmmmmm.js.

Locate the one you want, delete the existing Screen_xxxxxxxxxx.js file as in previous question, and then copy the screen version file into your definition set folder. Rename it to Screen_xxxxxxxxxx.js.

Before doing this end all aXes developer sessions.

If I rename a screen, does AXES automatically remove the old screen file?

No, AXES will leave the old file intact. You will have to remove it manually.

FAQ – Scripting

Can I create a script that runs when my application starts up and when the user has signed on?

Yes. See Application Level Basic Properties. Refer to the *onApplicationStart*, *onApplicationEnd*, *onSignOn* and *onSignOff* properties.

Can I create a script that runs when a particular screen arrives?

Yes. See axScreenBasicProperties - Screen Level Basic Properties. Refer to the *onArrive and onLeave* properties.

How can I hide AXES menu bar and status bar?

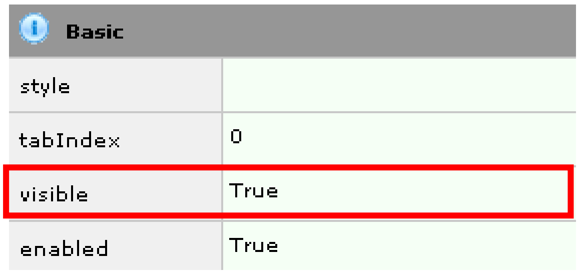
SHOWAXESMENUBAR and ***SHOWAXESSTATUSBAR*** functions control the visibility of menu & status bar. If you want to always hide the menu / status in your application, put the following code in the *onApplicationStart* event of your application.

```
SHOWAXESMENUBAR(false); // false = hide, true = show
SHOWAXESSTATUSBAR(false);
```

My script needs to change a property of an element's eXtension at runtime. Can it do that?

Yes, using the **setProperty** method of the element.

For example, a script under a certain condition needs to hide another element named "AdditionalDetails". We can do this by setting the **visible** property of the element to false (**visible** property can be found in the property sheet under **Basic Properties**).



Basic	
style	
tabIndex	0
visible	True
enabled	True

The script would look something like this:

```
// get a reference to the AdditionalDetails element
var F = FIELDS("AdditionalDetails");

// change the visible property to false
F.setProperty("visible", false);

// very important, must call refresh for property changes to take effect
F.refresh();
```

Note the last line – you need to call the **refresh** method of the element after changing a property of an element in order for the changes to be reflected visually.
Note how this is different from changing the current value of an element (remember that every element has a text value) using the **setValue** method – call to **refresh** is **not required** after **setValue**.

To get a property value, use the **getProperty** method.
Example:
If you wanted to show/hide according to its current state – if visible → hide otherwise show:

```
var F = FIELDS("AdditionalDetails");
var bVisible = F.getProperty("visible");

if (bVisible)
{
  F.setProperty("visible", false);
}
else
{
  F.setProperty("visible", true);
}

F.refresh();
```

Note: you could write the above in only 3 lines of code:

```
var F = FIELDS("AdditionalDetails");
F.setProperty("visible",!(F.getProperty("visible")))
F.refresh();
```

For more detailed information about **getProperty** and **setProperty** methods refer to the Scripting Reference.

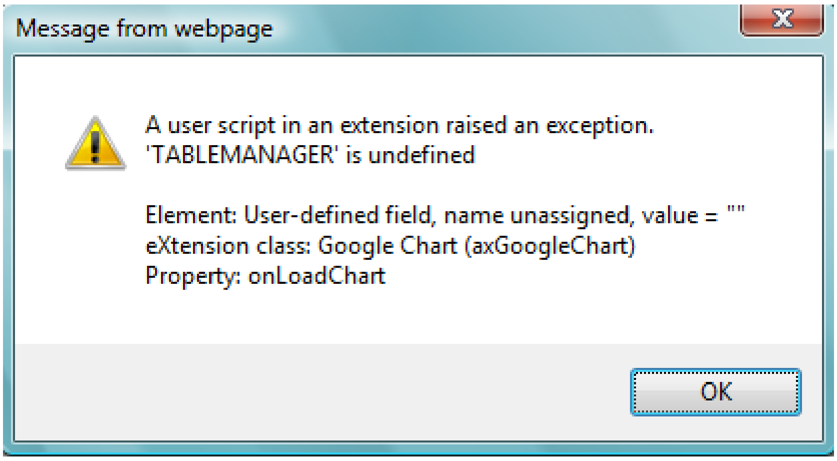
Can I stop the user pressing a function key?

Yes. Try this example as the **onLeave** script in a customized screen:

```
if (ENV.key == "F1")
{
   var bOK = window.confirm("Press OK to send F1=Help request. Click Cancel to stop it.");
   if (!bOK) ENV.returnValue = false;
}
```

When F1 is used this script confirms the usage - selectively cancelling it by setting ENV.returnValue = false;

When I reference the TABLEMANAGER in a USERENV.js function I get a runtime error, why?



Message from webpage

A user script in an extension raised an exception.
'TABLEMANAGER' is undefined

Element: User-defined field, name unassigned, value = ""
eXtension class: Google Chart (axGoogleChart)
Property: onLoadChart

Please refer to the section eXtension Scripting Global Functions & Objects in the Extension Scripting Reference. You must either pass the ENV object as a parameter to the UserEnv.js function or alternatively a direct reference to the table manager.
For example – in an extension property script you call a function in UserEnv.js called makeChart() and makeChart() which needs to refer to the table manager:

In the property script use

```
makeChart(ENV, …)
```

and in the UserEnv.js function

```
makeChart : function(env, …)
{
   ENV.TABLEMANAGER.loadStaticTables(this.staticTablesFile)
}
```

OR

In the property script use

```
makeChart(ENV.TABLEMANAGER, …)
```

and in the UserEnv.js function

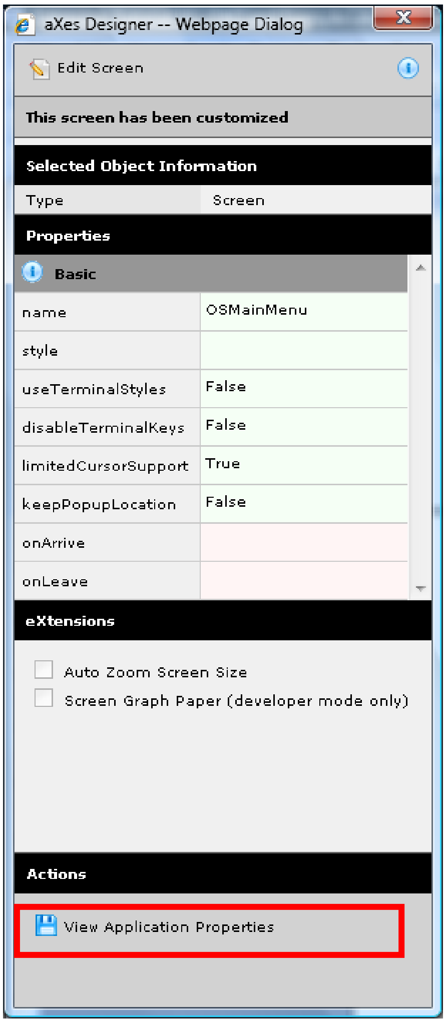
```
makeChart : function(tableManager, …)
{
   tableManager.loadStaticTables(this.staticTablesFile)
}
```

FAQ – Customizing eXtensions

I want to always use the “Modern” look for all my group box eXtensions, can I do this without having to change the “look” property of each group box?

Yes. When an eXtension is applied to a screen element, it is created based on a template. Every eXtension template is customizable – so what you need to do is to customize the group box template. Follow these steps to change the group box template’s “look” property to “modern”:

1. Switch the aXes Designer's view to **Application Properties** by clicking on the **View Application Properties** button at the bottom of the window.



aXes Designer -- Webpage Dialog

Edit Screen

This screen has been customized

Selected Object Information

Type	Screen

Properties

Basic

name	OSMainMenu
style	
useTerminalStyles	False
disableTerminalKeys	False
limitedCursorSupport	True
keepPopupLocation	False
onArrive	
onLeave	

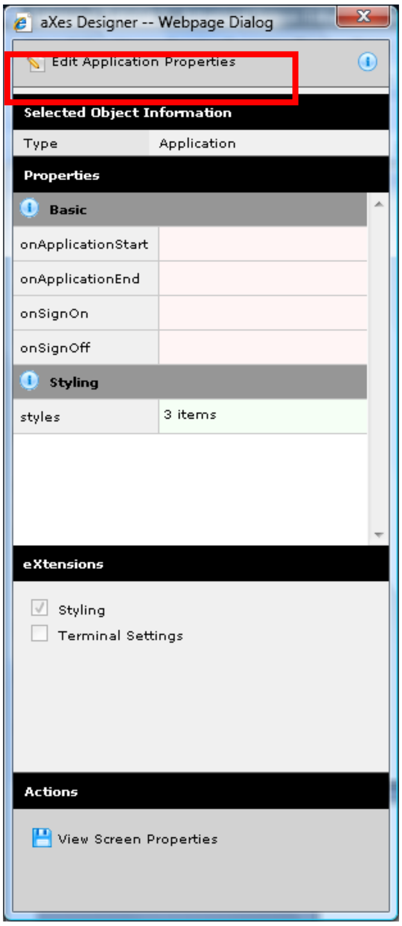
eXtensions

□ Auto Zoom Screen Size
□ Screen Graph Paper (developer mode only)

Actions

View Application Properties

2. Click on the **Edit Application Properties** button to switch to edit mode.



aXes Designer -- Webpage Dialog

Edit Application Properties

Selected Object Information

Type	Application

Properties

Basic

onApplicationStart	
onApplicationEnd	
onSignOn	
onSignOff	

Styling

styles	3 items

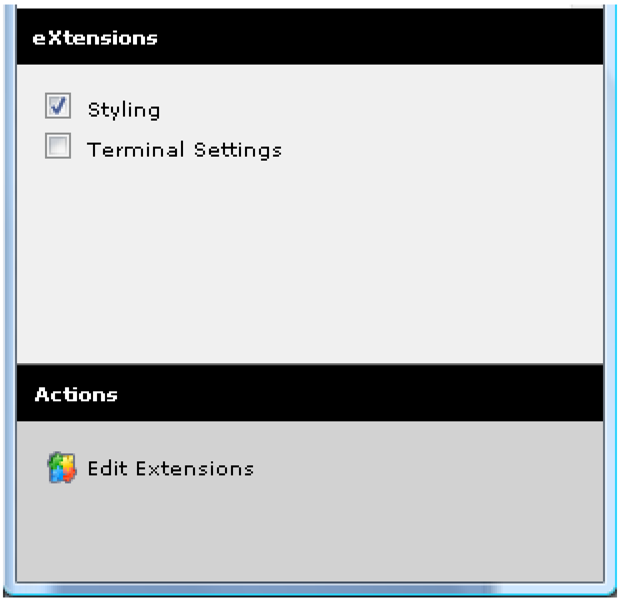
eXtensions

☑ Styling
□ Terminal Settings

Actions

View Screen Properties

3. Click on the **Edit Extensions** button at the bottom of the window.



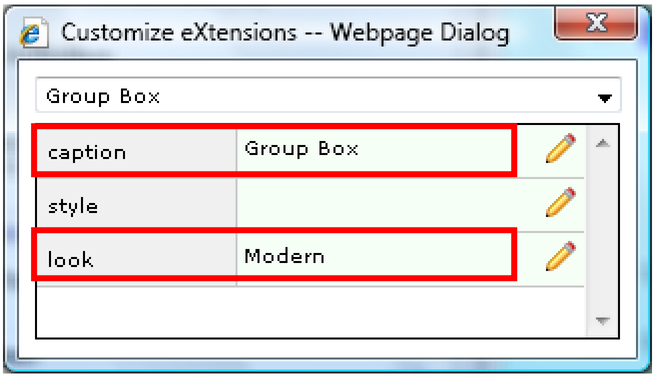
eXtensions

☑ Styling
□ Terminal Settings

Actions

Edit Extensions

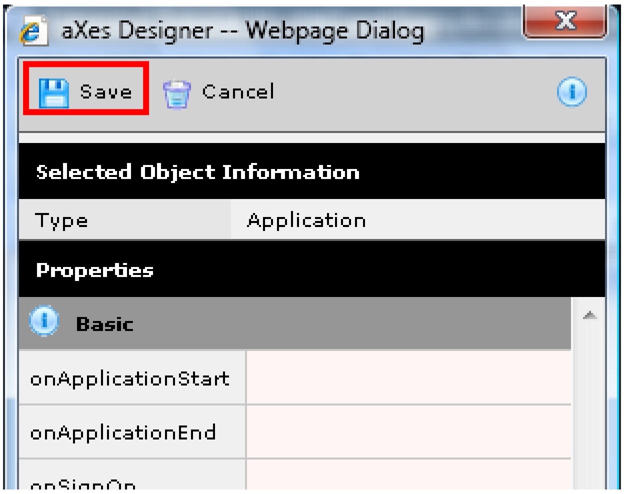
4. The **Customize eXtensions** window will appear, showing customizable eXtensions and a property sheet showing their current default values. Select **Group Box** in the drop down. Change the “look” property to “Modern”


Customize eXtensions -- Webpage Dialog

Group Box

caption	Group Box
style	
look	Modern

5. Close this window. Now save the changes you made by clicking on the **Save** button at the top of the aXes Designer window. You should see a message indicating a successful save.


aXes Designer -- Webpage Dialog

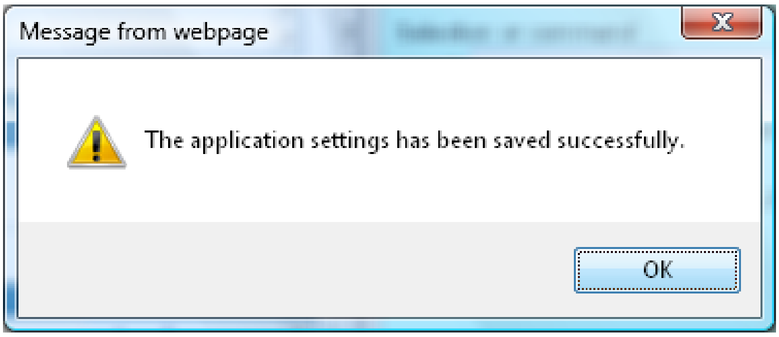
Selected Object Information

Type	Application

Properties

Basic

onApplicationStart	
onApplicationEnd	
onSignOn	


Message from webpage

The application settings has been saved successfully.

Now whenever you apply the group box eXtension to a screen element, you will see that the “look” property is always set to “Modern” by default.

When I have modified a property in an eXtension template, can I override the property value in certain eXtension instances?

Yes you can. Remember that what you modify in the template is just a default value. You can always set a different value in your actual eXtension instances.

What else do I need to know about customizing eXtension templates?

- Customize as early as possible before you start applying eXtensions to the screen elements.

- Customizing templates instead of modifying properties of each eXtension instances will result in a smaller screen definition file (which means faster download from the server) as AXES only stores properties whose values are different from the default.
- The “style” property of various eXtensions (e.g. button eXtensions) is a particularly good example of a property that should be set once only in the template as you would normally want all buttons to have the same style.

When I change the default “style” property of button eXtension, would this affect those buttons I created prior to this point?

Yes – as long as you haven’t modified the “style” property of those buttons. Remember that AXES physically stores only values that are different from their default values. So if you put various button eXtensions on your screen and you didn’t touch the “style” property of those buttons, any changes made to “style” property in the button template will immediately be reflected in all existing button instances.

FAQ – 5250 Popups

PopUp Windows and Screen Rendering in aXes-TS2

Over the years various 5250 pseudo window techniques have been utilized by 5250 developers. There are some important things you need to understand about them:

- In no viable way are they really desktop windows - or even particularly viable substitutes for Windows desktop windows (in the sense of the full Windows API capabilities).
- In new application design terms you should always view 5250 windows as the poor cousins of real Windows desktop windows (and even many other windows controls when used for field prompting).

These 5250 pseudo windows come in three basic forms:

- Imitations: the developer creates the appearance of a box using characters such as . : _ | (or a space with a background colour) to create a border. Recognising and dealing with windows of this type is not covered here.
- DDS(-defined) windows: the developer uses the DDS WINDOW keyword to define the window and the operating system uses graphics characters to draw the window on the terminal.
- Degraded DDS windows: the developer has defined the window in DDS but failed to leave space around it for the attribute bytes to turn graphics characters on/off. In this scenario, the operating system will draw it like a window imitation.

On most terminal emulators, degraded DDS windows look quite different to properly defined DDS windows so they should be easy to identify.

URL Popups Parameter

Until Version 2.1 aXes-TS2 treated every screen as a new screen and rendered it from scratch.

In aXes-TS2 V2.1 a new parameter *popups* has been introduced to the aXes URL to identify and handle degraded DDS windows and to provide the option of keeping the previous screen background unchanged when a popup window is drawn over it.

Recognising Degraded DDS Windows

Until Version 2.1 aXes-TS2 did not recognise degraded DDS windows. This means they were not drawn in boxes with borders with shadows that make the windows stand out clearly. aXes was also unable to prevent user interaction with the background.

To address this issue the aXes-TS2 engine has been modified to identify degraded DDS windows. It checks the details of the DDS record formats of the frontmost window sent by the aXesTS server, and if a record format of type "window" exists, but the screen contains no windows, aXes knows that a degraded DDS window exists and it can create one.

This behavior is enabled by default. The developer can use the URL popups parameter to instruct aXes to disable degraded DDS window detection:

```
http:\<host>\ts\ts2\index.html?popups=0
```

or

```
http:\<host>\ts\ts2\index.html?popups=2
```

See URL Parameters.

Limitations

If a degraded DDS window is drawn on top of another similar window, this technique cannot tell if it is a new window or the same window and will replace the currently rendered window.

Degraded DDS windows cannot be identified in TELNET/PASSTHRU sessions because DDS level record format information is not available.

Keeping the Background Customizations

Prior to version 2.1, when a window is drawn over the current window, any customizations applied to the background window are lost because aXes redraws the screen from scratch and the background is redrawn as uncustomized. This is not a functional problem but the changes in the background window may be confusing to the user.

Therefore aXes-TS2 screen rendering has been modified so that instead of automatically redrawing the screen from scratch, it can check if the virtual device has any windows. It compares the number found with the number currently rendered, deletes any excess (i.e. if the previous screen had a window now closed) and then redraws the topmost window. This way, the background along with any customizations remains intact.

In the majority of situations the background should be kept. This is the default behavior. However, to allow for special circumstances the developer can use the popups parameter to instruct aXes to not keep the background:

```
http:\<host>\ts\ts2\index.html?popups=0
```

or

```
http:\<host>\ts\ts2\index.html?popups=1
```

It should be noted that when the screen background is kept, there is a possibility of screen level eXtensions interfering with each other in unexpected ways. A window is seen as a new screen and, when customized, gets its own set of screen eXtensions. Where extensions were previously applied to a "clean slate", they are now being applied on top of the extensions from one or more previous screens. This may produce some unexpected results.

See URL Parameters.

When I start customizing a popup screen, it completely covers the whole screen, is this normal?

Yes. Popup screens will only show the previous screen in the area surrounding the popup if the popup screen is uncustomized. Once you start to customize a popup screen, it will cover up the surrounding area.

I have customized a screen (screen A) that can invoke a popup (screen B). When the popup appears, what was shown in the background is the original uncustomized screen A instead of the customized one. What should I do?

From version 2.1.11 onwards aXes-TS2 will keep the background customizations. Check that this has not been turned off with the *popups* parameter of the axes URL. See Keeping the Background Customizations.

In aXes-TS1 make the popup screen (screen B) into a customized screen - even when you don't change anything in the popup screen.
See the FAQ item above – when a popup screen (screen B) is customized, it will completely cover the whole screen & the previous screen (screen A) will not show in the background anymore.

My popup is making use of the cursor to allow the user to select an item from a list, how will this work after the screen has been customized as the cursor does not show anymore?

If your popup allows the user to select a value by displaying a list of values, and the user selects one by putting the cursor on the list entry and pressing enter, then the best alternative is to customize the popup screen and add a hyperlink extension to the first field in the popup's list, as explained in the section above (Example – Working with 5250 Cursor).
You could also enable the **limitedCursorSupport** option on the popup screen, though visually, the first option is much better. Note that the limited cursor support is meant as a temporary measure only and should not be used as a permanent solution.
If you are enabling the **limitedCursorSupport**, make sure that you set the **keepPopupLocation** property of the popup screen to true.

PopupWindow Extension

You can use the axPopupWindow extension to customize the position, size and style of a 5250 window in TS2. You may want to use the extension for example when you have a customised background that has rearranged the screen and you also need to move the initial position of popups.

FAQ – Calling IBM i Server programs from eXtension scripts

The TABLEMANAGER object provided by aXes allows you to call RPG or CL programs directly from eXtension JavaScript. The programs that you call need to be thread safe. See the end of this section for more details. Typically the calls are made when a button or hyperlink is clicked.

Example 1 – Call with no parameters

Server CL program

This small CL program named LOTST009 sends a message to the system history log:

```
PGM
SNDPGMMSG  MSG('Hello from LOTST009') TOMSGQ(QHST)
ENDPGM
```

Button onClick JavaScript coding in Axes

```
TABLEMANAGER.callProgram("LOTST009","QGPL");
```

Result

Display the system history log (DSPLOG command) and locate message "Hello from LOTST009".

Example 2 – Call with one parameter

Server CL program

This small CL program named LOTST011 sends a message to the system history log. The message text is passed into the program as a parameter.

```
PGM (&MSG)
   DCL &MSG *CHAR 132
   SNDPGMMSG  MSG(&MSG) TOMSGQ(QHST)
ENDPGM
```

Button onClick JavaScript coding in Axes

```
TABLEMANAGER.callProgram("LOTST011","QGPL",
             {type:"alpha",len:132,value:"This is a test message"}  );
```

Result

Display the system history log (DSPLOG command) and locate message "This is a test message".

Example 3 – Call with a returned value

Server CL program

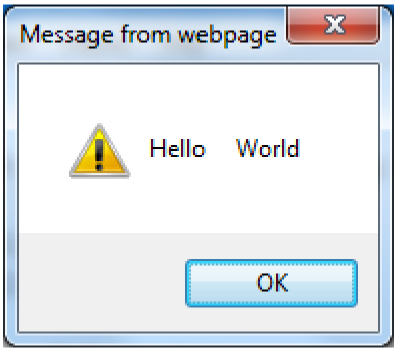
This small CL program named LOTST001 concatenates two alpha parameters and returns the result.

```
PGM (&P1 &P2 &P3)
   DCL &P1 *CHAR 10
   DCL &P2 *CHAR 10
   DCL &P3 *CHAR 20
   CHGVAR &P3 (&P1 *CAT &P2)
ENDPGM
```

Button onClick JavaScript coding in Axes

```
var result = TABLEMANAGER.callProgram("LOTST001","QGPL",
                          {type:"alpha",len:10,value:"Hello"},
                          {type:"alpha",len:10,value:"World"},
                          {type:"alpha",len:20,pass:false,ret:true} );

if (result.error) alert("Call Failed");
else alert(result.returnParms[3]);
```

Result



Message from webpage

Hello   World

Example 4 – Numeric Parameters passed and returned

Server CL program

This small CL program named LOTST002 adds up 3 packed decimal parameters and returns the result.

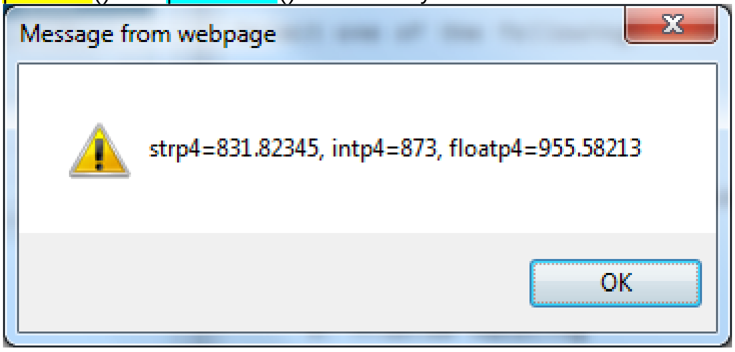
```
PGM (&P1 &P2 &P3 &P4)
DCL &P1 *DEC (7 0)
DCL &P2 *DEC (9 2)
DCL &P3 *DEC (12 5)
DCL &P4 *DEC (15 5)
CHGVAR &P4 (&P1 + &P2 + &P3)
ENDPGM
```

Button onClick JavaScript coding in Axes

```
var result = TABLEMANAGER.callProgram("LOTST002","QGPL",
                  { type:"packed",len:7,dec:0,value: 4 },
                  { type:"packed",len:9,dec:2,value: 7.7 },
                  { type:"packed",len:12,dec:5,value: 820.12345 },
                  { type:"packed",len:15,dec:5,pass:false,ret:true } );

if (result.error) alert("Call Failed");
else
{
  var strp4   = result.returnParms[4];         /* Should be 831.82345 as string */
  var intp4   = parseInt(strp4,10) + 42;       /* Should be 831 + 42 = 873 */
  var floatp4 = parseFloat(strp4) + 123.75868; /* Should be 955.58213 as float */
  alert("strp4=" + strp4 + ", intp4=" + intp4.toString() + ", floatp4=" +
floatp4.toString());
}
```

Result

Note that the returned parameter (strP4) is a always a string. By using the standard JavaScript functions parseInt() and parseFloat() it can easily be converted to a number for further manipulation by JavaScript code.



strp4=831.82345, intp4=873, floatp4=955.58213

Example 5 – Value Passed and Returned in same Parm

Server CL program

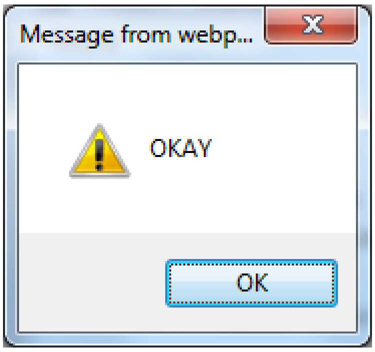
This small CL program named LOTST005 sends a message to the system operator and returns "OKAY":

```
PGM (&P1)
DCL &P1 *CHAR 20
SNDMSG     MSG(&P1) TOMSGQ(QSYSOPR)
CHGVAR &P1 'OKAY'
ENDPGM
```

Button onClick JavaScript coding in Axes

```
var message = "Hello from LOTST005";
var result  = TABLEMANAGER.callProgram("LOTST005","QGPL",
                 {type:"alpha",len:20,pass:true,ret:true,value:message} );

if (result.error) alert("Call Failed");
else alert(result.returnParms[1]);
```

Result



OKAY

Example 6 – Multiple Values Passed and Returned

Server CL program

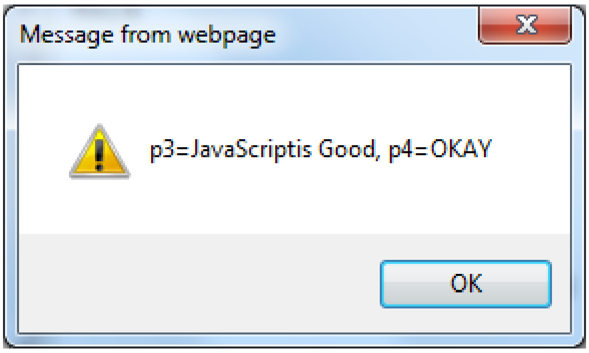
This small CL program named LOTST007 receives and returns multiple parameters:

```
PGM (&P1 &P2 &P3 &P4)
DCL &P1 *CHAR 10
DCL &P2 *CHAR 10
DCL &P3 *CHAR 20
DCL &P4 *CHAR 10
CHGVAR &P3 (&P1 *CAT &P2)
CHGVAR &P4 'OKAY'
ENDPGM
```

Button onClick JavaScript coding in Axes

```
var p1 = "JavaScript";
var p2 = "is Good";
var result  = TABLEMANAGER.callProgram("LOTST007","QGPL",
                 { type:"alpha",len:10,pass:true,value:p1  },
                 { type:"alpha",len:10,pass:true,value:p2  },
                 { type:"alpha",len:20,pass:false,ret:true },
                 { type:"alpha",len:10,pass:false,ret:true } );
if (result.error) alert("Call Failed");
else
{
   var p3 = result.returnParms[3];
   var p4 = result.returnParms[4];
   alert("p3=" + p3 + ", p4=" + p4);
}
```

Result



p3=JavaScriptis Good, p4=OKAY

Example 7 – Multiple Values Returned in One Parameter

Server CL program

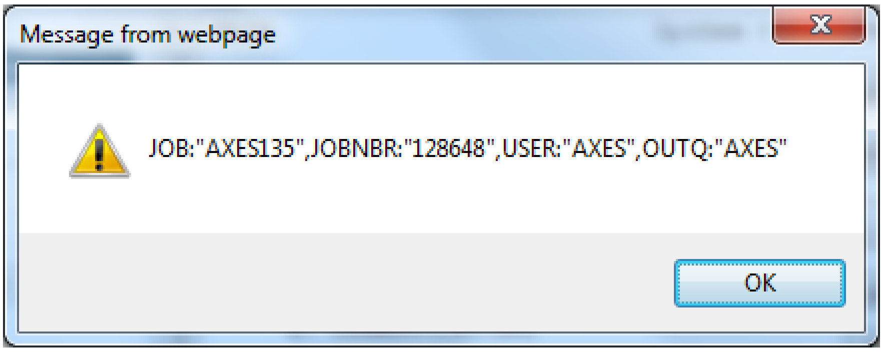
This small CL program named LOTST010 returns a 2000 byte string containing the aXes server's job name, user profile, job number and current output queue. The returned information is formatted as a JSON string. This program is the tip of a very large iceberg of possibility.

```
PGM (&JSON)
   DCL &JSON *CHAR 2000

   DCL &JOB    *CHAR 10
   DCL &USER   *CHAR 10
   DCL &JOBNBR *CHAR 6
   DCL &OUTQ   *CHAR 10

   RTVJOBA JOB(&JOB) USER(&USER) NBR(&JOBNBR) OUTQ(&OUTQ)

   CHGVAR &JSON (' JOB:"' || &JOB |< '"')
   CHGVAR &JSON (&JSON |< ',JOBNBR:"' || &JOBNBR |< '"')
   CHGVAR &JSON (&JSON |< ',USER:"' || &USER |< '"')
   CHGVAR &JSON (&JSON |< ',OUTQ:"' || &OUTQ |< '"')

ENDPGM
```

Button onClick JavaScript coding in Axes – Stage 1

```
var result =
TABLEMANAGER.callProgram("LOTST010","QGPL",{type:"alpha",len:2000,pass:false,ret:true});
alert( result.returnParms[1] );
```

Result – Stage 1



Message from webpage

JOB:"AXES135",JOBNBR:"128648",USER:"AXES",OUTQ:"AXES"

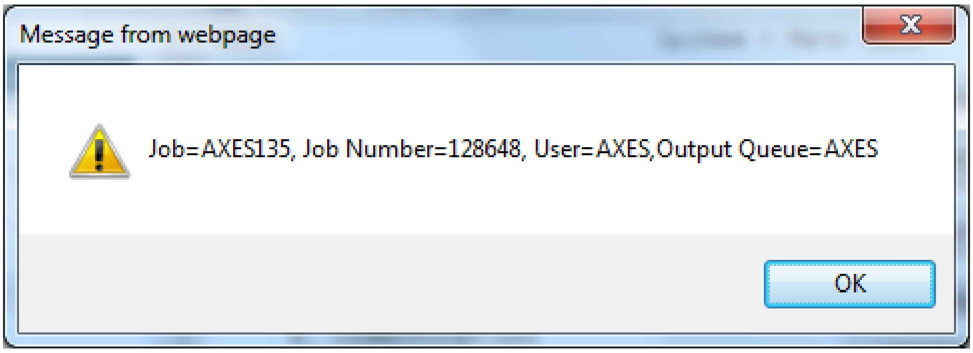
The CL program LOTST010 is executed and it returns a long string with the job name, job number, user and output queue all imbedded in it.

In effect LOTST010 has returned 4 values, but done it using a single program parameter.
Note how the returned string is formatted in much the same way as the program arguments are specified in JavaScript - type:"alpha",len:2000,pass:false,ret:true

Button onClick JavaScript coding in Axes – Stage 2

```
var result =
TABLEMANAGER.callProgram("LOTST010","QGPL",{type:"alpha",len:2000,pass:false,ret:true});
try
{
   var Info = eval( "({" + result.returnParms[1] + "})" )
   window.alert("Job=" + Info.JOB + ", Job Number=" + Info.JOBNBR + ", User=" + Info.USER +
",Output Queue=" + Info.OUTQ);
}
catch (oe)
{
   window.alert( "Error " + oe.description + " detected when loading JSON data " +
result.returnParms[1] );
}
```

Result – Stage 2



Message from webpage

Job=AXES135, Job Number=128648, User=AXES,Output Queue=AXES

The CL program LOTST010 is executed and it returns a long string with the job name, job number, user and output queue all imbedded in it.

The value returned is in JSON string format.

The returned value is executed by using a JavaScript eval operation to create a JavaScript object named Info.

Now the properties Info.JOB, Info.JOBNBR, , Info.USER and Info.OUTQ are all accessible to the JavaScript code. In effect the 4 values returned by CL program LOTST010 are now individually accessible as JavaScript object properties.

The JSON interface format is a driving part of the Web 2.0 / AJAX technologies. Some of the reasons it is very useful include:

- You can alter CL program LOTST010 to return more values at any time. This would not upset any existing JavaScript calls to LOTST010. That would not be true if you were using traditional parameters – every caller would need to change.

- Your JavaScript can easily check whether the version of LOTST010 that it called has actually returned a property by coding, say,

```
var UseJobSize = 42; /* Set the default value this thing */
if (Info.JOBSIZE != null) UseJobSize = Info.JOBSIZE; /* Server value provided */
```

  to decide whether LOTST010 returned a value named JOBSIZE.

- The JSON string you return can be much more complex in nature and include lists, arrays, structures, etc. All of these are instantly accessible to your JavaScript code.

Usage Rules, Guidelines and Tips

The server side program calls are not performed in the same job as the 5250 session. They are initiated from within the aXes server job.

The calls are performed under the user profile that the aXes server jobs run under – not under the user profile who started the 5250 session. However, you can call a CL program that submits jobs for execution under 5250 users profile. See the IBM i SBMJOB command and the concept of job descriptions.

The calls are performed in a multi-threaded process. This means that all resources are shared with all the other active threads - which may also be executing concurrent call operations.

Some of the things that this means include:
You have only 1 QTEMP library that is shared by all threads in the process (ie: IBM i server job).

The IBM i library list concept cannot be practically used. If you change the job's library list then a millisecond later another thread may alter it to something else. You have to use library qualified reference to most IBM i objects. Implement a design such that the client USERENV object knows the library name(s) associated with the 5250 user and pass them to the server program(s) as parameters so that all object references are fully qualified.
Your CL and RPG programs need to be compiled to be thread safe. Generally programs should be compiled using the CRTBNDCL or CRTBNDRPG commands. RPG programs probably need to use the THREAD(*SERIALIZE) control specification option. You should also review all IBM supplied documentation for executing multi-threaded RPG and CL programs before using this aXes feature.
Your CL and RPG programs cannot open a 5250 display file because they are not executing in a context where they are associated with a 5250 device.
Your CL and RPG programs need to be robust. They need to trap errors and handle them gracefully releasing all open or allocated resources.

Your CL and RPG programs need to be symmetrical in resource usage. Basically this means that if something is opened, locked or allocated as the program is executing it always it needs to be symmetrically closed, unlocked or de-allocated as the program is terminating.
Your server programs need to be stateless between each call. This means they must terminate (set on LR in RPG terminology) at the end of every call and cannot remember values between calls in their variables nor leave any resource open or allocated. Where a stateful design is required, typically a unique token or some sort of session id is assigned by the server that can be used to save and restore state on each call.

Your CL and RPG programs need to execute quickly. Anything that would take more than 1 or 2 seconds to execute should be submitted to batch instead.

If your 5250 session locks up you probably have poor error handling in your program. It has failed and is waiting for the system operator to reply to a message. It is recommended that you improve such default IBM i error handling.

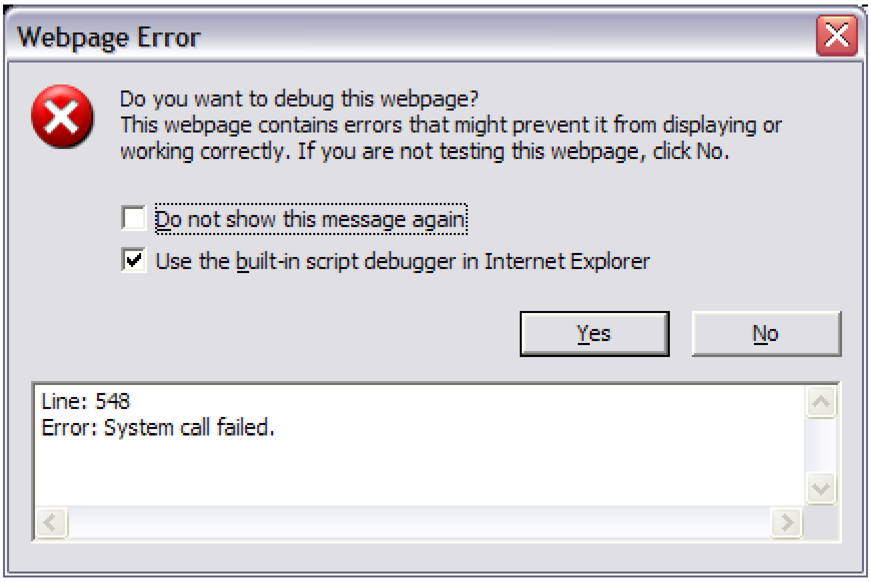
Your RPG or CL program must never ever change the execution characteristics of the IBM i job they are executing. Job characteristics include anything that may impact other programs executing within the same job – including things like library lists, priorities, activation groups, CCSIDs, contents of QTEMP, etc.

FAQ – Developer Mode Issues

aXes Designer has become very sluggish, is there anything I can do?

After using aXes in developer mode for a while, you might feel that it has become slow and heavy. This could be caused by the Internet Explorer memory leaking. Unfortunately there is no solution to this problem and the only way to get around this is by restarting your browser completely when you are experiencing this problem. Please make sure that you completely close your browser window (as opposed to just reloading aXes).

I'm getting an error message "System call failed" while attempting to perform an operation in the designer. What should I do?



Webpage Error

Do you want to debug this webpage?
This webpage contains errors that might prevent it from displaying or working correctly. If you are not testing this webpage, click No.

□ Do not show this message again

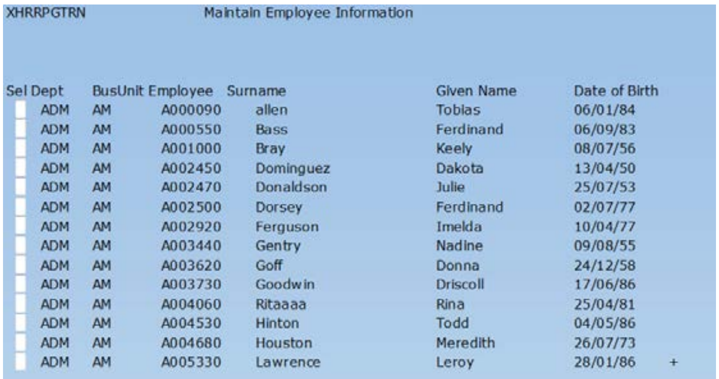
☑ Use the built-in script debugger in Internet Explorer

Line: 548
Error: System call failed.

If you are in the middle of editing, save your work immediately.
Then close your browser window and reopen it.

Example – Using a Hyperlink to select a subfile entry

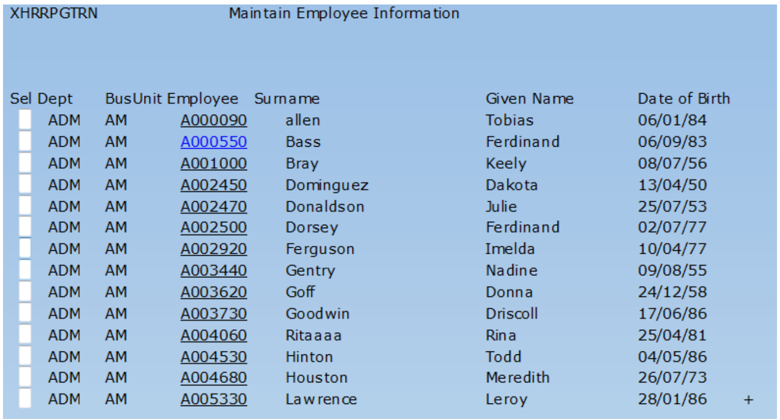
Start Point
The shipped aXes demonstration screen named XHRRPGTRN_Select is used.
The subfile selection column is field named Sel and the employee number column is named Employee:



XHRRPGTRN Maintain Employee Information

Sel	Dept	BusUnit	Employee	Surname	Given Name	Date of Birth
□	ADM	AM	A000090	allen	Tobias	06/01/84
□	ADM	AM	A000550	Bass	Ferdinand	06/09/83
□	ADM	AM	A001000	Bray	Keely	08/07/56
□	ADM	AM	A002450	Dominguez	Dakota	13/04/50
□	ADM	AM	A002470	Donaldson	Julie	25/07/53
□	ADM	AM	A002500	Dorsey	Ferdinand	02/07/77
□	ADM	AM	A002920	Ferguson	Imelda	10/04/77
□	ADM	AM	A003440	Gentry	Nadine	09/08/55
□	ADM	AM	A003620	Goff	Donna	24/12/58
□	ADM	AM	A003730	Goodwin	Driscoll	17/06/86
□	ADM	AM	A004060	Ritaaaa	Rina	25/04/81
□	ADM	AM	A004530	Hinton	Todd	04/05/86
□	ADM	AM	A004680	Houston	Meredith	26/07/73
□	ADM	AM	A005330	Lawrence	Leroy	28/01/86 +

Result
The selection column is still visible, but an employee can be selected and displayed by clicking on the employee number hyperlink:



XHRRPGTRN Maintain Employee Information

Sel	Dept	BusUnit	Employee	Surname	Given Name	Date of Birth	
□	ADM	AM	A000090	allen	Tobias	06/01/84	
□	ADM	AM	A000550	Bass	Ferdinand	06/09/83	
□	ADM	AM	A001000	Bray	Keely	08/07/56	
□	ADM	AM	A002450	Dominguez	Dakota	13/04/50	
□	ADM	AM	A002470	Donaldson	Julie	25/07/53	
□	ADM	AM	A002500	Dorsey	Ferdinand	02/07/77	
□	ADM	AM	A002920	Ferguson	Imelda	10/04/77	
□	ADM	AM	A003440	Gentry	Nadine	09/08/55	
□	ADM	AM	A003620	Goff	Donna	24/12/58	
□	ADM	AM	A003730	Goodwin	Driscoll	17/06/86	
□	ADM	AM	A004060	Ritaaaa	Rina	25/04/81	
□	ADM	AM	A004530	Hinton	Todd	04/05/86	
□	ADM	AM	A004680	Houston	Meredith	26/07/73	
□	ADM	AM	A005330	Lawrence	Leroy	28/01/86	+

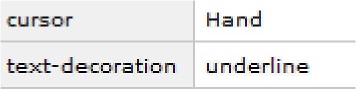
Steps

- ❖ The employee column was selected and changed to be a Hyperlink.
- ❖ These properties were then set in the hyperlink eXtension:

caption (script):

ENV.returnValue = FIELD.getValue();

style



cursor	Hand
text-decoration	underline

onClick (script)

```
/* Find out the subfile index of this field (the one clicked on) */

var iSFLIndex = FIELD.getIndex();

/* Get a reference to the "Sel" field with the same subfile index */

var oSel = FIELDS("Sel",iSFLIndex);

/* If found, set the "Sel" field to "X" and press Enter */

if (oSel != null)
{
   oSel.setValue("X");
   SENDKEY("Enter");
}
```

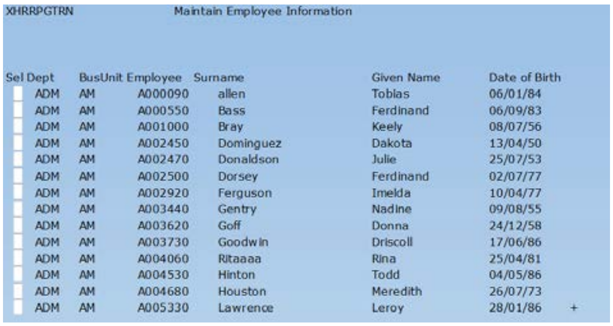
mouseOverColor

blue

Example – Iterating through subfiles entries

Start Point

The shipped aXes demonstration screen named XHRRPGTRN_Select is used.
The subfile selection column field is named Sel, and the employee number, Surname and Given Name columns are named Employee, Surname and Given_Name respectively:

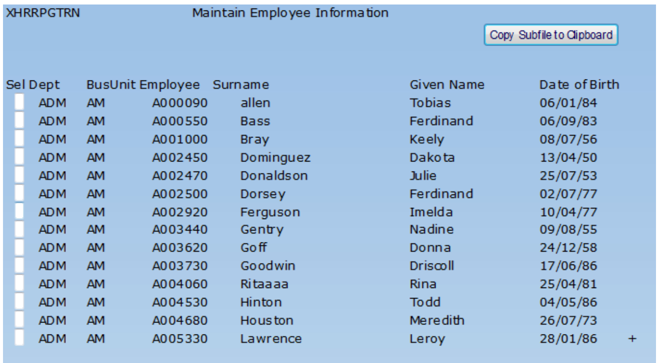


XHRRPGTRN Maintain Employee Information

Sel	Dept	BusUnit	Employee	Surname	Given Name	Date of Birth
□	ADM	AM	A000090	allen	Tobias	06/01/84
□	ADM	AM	A000550	Bass	Ferdinand	06/09/83
□	ADM	AM	A001000	Bray	Keely	08/07/56
□	ADM	AM	A002450	Dominguez	Dakota	13/04/50
□	ADM	AM	A002470	Donaldson	Julie	25/07/53
□	ADM	AM	A002500	Dorsey	Ferdinand	02/07/77
□	ADM	AM	A002920	Ferguson	Imelda	10/04/77
□	ADM	AM	A003440	Gentry	Nadine	09/08/55
□	ADM	AM	A003620	Goff	Donna	24/12/58
□	ADM	AM	A003730	Goodwin	Driscoll	17/06/86
□	ADM	AM	A004060	Ritaaaa	Rina	25/04/81
□	ADM	AM	A004530	Hinton	Todd	04/05/86
□	ADM	AM	A004680	Houston	Meredith	26/07/73
□	ADM	AM	A005330	Lawrence	Leroy	28/01/86 +

Result

A copy to clipboard push button has been added. When clicked all the subfile entries on the current screen are copied to the clipboard:



XHRRPGTRN Maintain Employee Information

Sel	Dept	BusUnit	Employee	Surname	Given Name	Date of Birth
□	ADM	AM	A000090	allen	Tobias	06/01/84
□	ADM	AM	A000550	Bass	Ferdinand	06/09/83
□	ADM	AM	A001000	Bray	Keely	08/07/56
□	ADM	AM	A002450	Dominguez	Dakota	13/04/50
□	ADM	AM	A002470	Donaldson	Julie	25/07/53
□	ADM	AM	A002500	Dorsey	Ferdinand	02/07/77
□	ADM	AM	A002920	Ferguson	Imelda	10/04/77
□	ADM	AM	A003440	Gentry	Nadine	09/08/55
□	ADM	AM	A003620	Goff	Donna	24/12/58
□	ADM	AM	A003730	Goodwin	Driscoll	17/06/86
□	ADM	AM	A004060	Ritaaaa	Rina	25/04/81
□	ADM	AM	A004530	Hinton	Todd	04/05/86
□	ADM	AM	A004680	Houston	Meredith	26/07/73
□	ADM	AM	A005330	Lawrence	Leroy	28/01/86 +

Steps

- A new element was added to the screen as a push button.
- These properties were then set in the push button eXtension:

caption (simple text):

Copy Subfile to Clipboard

onClick (script)

```
var sClipData = "";
var sTab      = "\t";

var iSFLIndex = 1;
var oEmployee = FIELDS("Employee",iSFLIndex);
while (oEmployee != null)
{
    var oSurname   = FIELDS("Surname",iSFLIndex);
    var oGivenName = FIELDS("Given_Name",iSFLIndex);
    sClipData += oEmployee.getValue() + sTab;

    if (oSurname != null) sClipData += oSurname.getValue()   + sTab;
    else sClipData += "UNKNOWN" + sTab;

    if (oGivenName != null) sClipData += oGivenName.getValue() + sTab;
    else            sClipData += "UNKNOWN" + sTab;

    sClipData += "\n";
    iSFLIndex += 1;
    oEmployee = FIELDS("Employee",iSFLIndex);
}
window.clipboardData.clearData();
window.clipboardData.setData("Text",sClipData);
```

Observations

The employee number field was chosen to drive the subfile iteration loop because it exists for every subfile entry.

The subfile iteration loop is structured like this:

```
var iSFLIndex = 1;
var oEmployee = FIELDS("Employee",iSFLIndex);
while (oEmployee != null)
{

    << Process employee >>

    iSFLIndex += 1;
    oEmployee = FIELDS("Employee",iSFLIndex);
}
```

The loop starts at entry one and proceeds until an employee number cannot be found in the subfile.

When iterating a subfile the possibility exists that blank outfield entry will not actually exist on the 5250 display. That is why the logic

```
    var oSurname = FIELDS("Surname",iSFLIndex);

    if (oSurname != null) sClipData += oSurname.getValue()   + sTab;
    else                  sClipData += "UNKNOWN" + sTab;
```

is used. It caters for the fact that an employee's surname may be blank and therefore the entry does not exist in the subfile.

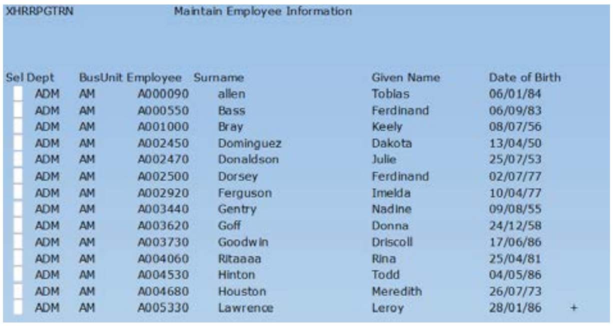
The clipboard data is tab delimited so it can be pasted as columns to, for example, MS-Excel.

You would probably not code logic like this in a real application. It would be better to create a generic subfile clipboard function in your USERENV object and simply call it from the onClick routine.

Example – USERENV and Generic Coding

Start Point

The shipped aXes demonstration screen named XHRRPGTRN_Select is used.
The subfile selection column field is named Sel, and the employee number, Surname, Given Name and Date of Birth columns are named Employee, Surname and Given_Name and Date_of_Birth respectively:



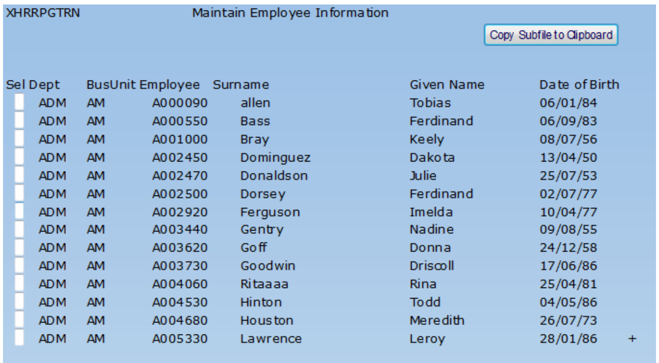
XHRRPGTRN Maintain Employee Information

Sel	Dept	BusUnit	Employee	Surname	Given Name	Date of Birth
	ADM	AM	A000090	allen	Tobias	06/01/84
	ADM	AM	A000550	Bass	Ferdinand	06/09/83
	ADM	AM	A001000	Bray	Keely	08/07/56
	ADM	AM	A002450	Dominguez	Dakota	13/04/50
	ADM	AM	A002470	Donaldson	Julie	25/07/53
	ADM	AM	A002500	Dorsey	Ferdinand	02/07/77
	ADM	AM	A002920	Ferguson	Imelda	10/04/77
	ADM	AM	A003440	Gentry	Nadine	09/08/55
	ADM	AM	A003620	Goff	Donna	24/12/58
	ADM	AM	A003730	Goodwin	Driscoll	17/06/86
	ADM	AM	A004060	Ritaaaa	Rina	25/04/81
	ADM	AM	A004530	Hinton	Todd	04/05/86
	ADM	AM	A004680	Houston	Meredith	26/07/73
	ADM	AM	A005330	Lawrence	Leroy	28/01/86

+

Result

A copy to clipboard push button has been added. When clicked all the subfile entries on the current screen are copied to the clipboard. The copy is achieved by invoking a generic function defined in the USERENV object:



XHRRPGTRN Maintain Employee Information

Sel	Dept	BusUnit	Employee	Surname	Given Name	Date of Birth
	ADM	AM	A000090	allen	Tobias	06/01/84
	ADM	AM	A000550	Bass	Ferdinand	06/09/83
	ADM	AM	A001000	Bray	Keely	08/07/56
	ADM	AM	A002450	Dominguez	Dakota	13/04/50
	ADM	AM	A002470	Donaldson	Julie	25/07/53
	ADM	AM	A002500	Dorsey	Ferdinand	02/07/77
	ADM	AM	A002920	Ferguson	Imelda	10/04/77
	ADM	AM	A003440	Gentry	Nadine	09/08/55
	ADM	AM	A003620	Goff	Donna	24/12/58
	ADM	AM	A003730	Goodwin	Driscoll	17/06/86
	ADM	AM	A004060	Ritaaaa	Rina	25/04/81
	ADM	AM	A004530	Hinton	Todd	04/05/86
	ADM	AM	A004680	Houston	Meredith	26/07/73
	ADM	AM	A005330	Lawrence	Leroy	28/01/86

+

Steps

- This generic function was added to the USERENV object (this code is a generic version of the code using in the preceding subfile and clipboard example):

```
sendSubfiletoClipBoard : function(oP)
{
   var iSFLIndex  = 0;
   var iLimit     = oP.sendFields.length;
   var sClipData  = "";
   var sTab       = "\t";
   var sNL        = "\n";
   var oiterField = null;

   iSFLIndex = 1;
   oiterField = oP.ENV.FIELDS(oP.iterField,iSFLIndex);
   while (oiterField != null)
   {
      for (var i = 0; i < iLimit; i++)
      {
         var osendField = oP.ENV.FIELDS(oP.sendFields[i],iSFLIndex);
         if (osendField != null) sClipData += osendField.getValue()   + sTab;
         else                    sClipData += oP.notFound + sTab;
      }
      sClipData += sNL;
      iSFLIndex += 1;
      oiterField  = oP.ENV.FIELDS(oP.iterField,iSFLIndex);
   }

   window.clipboardData.clearData();
   if (sClipData != "") window.clipboardData.setData("Text",sClipData);

}, /* <= Note the comma */
```

- A new aXes developer browser session was started to ensure that the updated USERENV code is loaded.
- A new element was added to the XHRRPGTRN_Select screen as a push button.
- These properties were then set in the push button eXtension:

  caption (simple text):

  Copy Subfile to Clipboard

  onClick (script)

```
var oP = { ENV       : ENV,
           iterField : "Employee",
           notFound  : "Not found",
           sendFields: ["Employee","Surname","Given_Name","Date_of_Birth"] };

USERENV.sendSubfiletoClipBoard(oP);
```

Observations

The USERENV function sendSubfiletoClipBoard receives an object as a parameter.

The properties within the object parameter are:

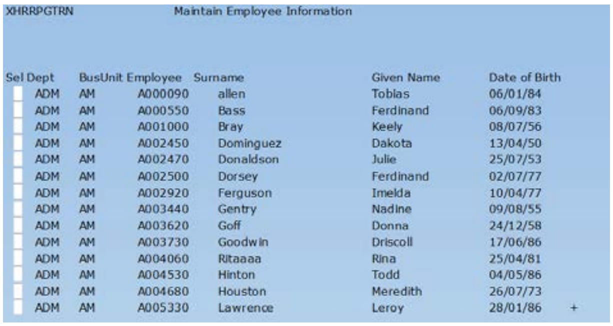
- ENV: a reference to the calling scripts ENV execution environment. This allows the generic code to access the callers standard functions, etc.
- iterField: The name of the subfile fields to use to iterate the subfile.
- notFound: The text to be output to the clipboard for a field not found in the subfile.
- sendFields: An array of the names of the subfile fields to be copied to the clipboard.

Using a generic routine like this in USERENV means that it is easier to add a "Copy to Clipboard" button to lots of screens and avoid repeating JavaScript code.

Example – Selectively Iterating through subfiles entries

Start Point

The shipped aXes demonstration screen named XHRRPGTRN_Select is used.
The subfile selection column field is named Sel, and the employee number, Surname and Given Name columns are named Employee, Surname and Given_Name respectively:

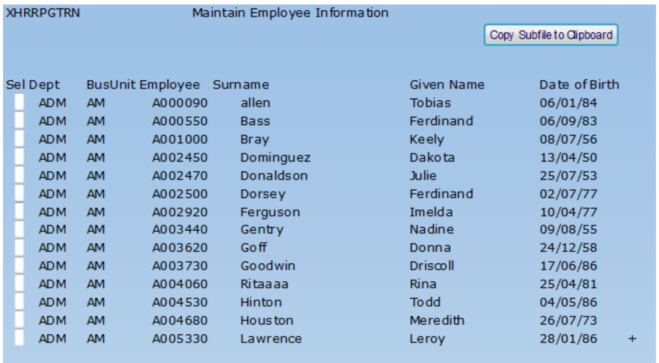


XHRRPGTRN Maintain Employee Information

Sel	Dept	BusUnit	Employee	Surname	Given Name	Date of Birth
	ADM	AM	A000090	allen	Tobias	06/01/84
	ADM	AM	A000550	Bass	Ferdinand	06/09/83
	ADM	AM	A001000	Bray	Keely	08/07/56
	ADM	AM	A002450	Dominguez	Dakota	13/04/50
	ADM	AM	A002470	Donaldson	Julie	25/07/53
	ADM	AM	A002500	Dorsey	Ferdinand	02/07/77
	ADM	AM	A002920	Ferguson	Imelda	10/04/77
	ADM	AM	A003440	Gentry	Nadine	09/08/55
	ADM	AM	A003620	Goff	Donna	24/12/58
	ADM	AM	A003730	Goodwin	Driscoll	17/06/86
	ADM	AM	A004060	Ritaaaa	Rina	25/04/81
	ADM	AM	A004530	Hinton	Todd	04/05/86
	ADM	AM	A004680	Houston	Meredith	26/07/73
	ADM	AM	A005330	Lawrence	Leroy	28/01/86 +

Result

A copy to clipboard push button has been added. When clicked subfile entries selected with a "C" on the current screen are copied to the clipboard:



XHRRPGTRN Maintain Employee Information

Sel	Dept	BusUnit	Employee	Surname	Given Name	Date of Birth
	ADM	AM	A000090	allen	Tobias	06/01/84
	ADM	AM	A000550	Bass	Ferdinand	06/09/83
	ADM	AM	A001000	Bray	Keely	08/07/56
	ADM	AM	A002450	Dominguez	Dakota	13/04/50
	ADM	AM	A002470	Donaldson	Julie	25/07/53
	ADM	AM	A002500	Dorsey	Ferdinand	02/07/77
	ADM	AM	A002920	Ferguson	Imelda	10/04/77
	ADM	AM	A003440	Gentry	Nadine	09/08/55
	ADM	AM	A003620	Goff	Donna	24/12/58
	ADM	AM	A003730	Goodwin	Driscoll	17/06/86
	ADM	AM	A004060	Ritaaaa	Rina	25/04/81
	ADM	AM	A004530	Hinton	Todd	04/05/86
	ADM	AM	A004680	Houston	Meredith	26/07/73
	ADM	AM	A005330	Lawrence	Leroy	28/01/86 +

Steps

- ❖ A new element was added to the screen as a push button.
- ❖ These properties were then set in the push button eXtension:

caption (simple text):

Copy Subfile to Clipboard

onClick (script)

```
var sClipData = "";
var sTab      = "\t";

var iSFLIndex = 1;
var oSel = FIELDS("Sel",iSFLIndex);
while (oSel != null)
{
    if (oSel.getValue() == "C")
    {
       var oEmployee  = FIELDS("Employee",iSFLIndex);
       var oSurname   = FIELDS("Surname",iSFLIndex);
       var oGivenName = FIELDS("Given_Name",iSFLIndex);
       oSel.setValue(" ");
       sClipData += oEmployee.getValue() + sTab;
       if (oSurname != null) sClipData += oSurname.getValue()   + sTab;
       else sClipData += "UNKNOWN" + sTab;
       if (oGivenName != null) sClipData += oGivenName.getValue() + sTab;
       else            sClipData += "UNKNOWN" + sTab;
       sClipData += "\n";
    }

    iSFLIndex += 1;
    oSel = FIELDS("Sel",iSFLIndex);
}

if (sClipData == "")
{
  window.alert("No employees were selected to copy. Put a C beside the employees to copy.");
}
else
{
   window.clipboardData.clearData();
   window.clipboardData.setData("Text",sClipData);
}
```

Observations

The selection ("Sel") field was chosen to drive the subfile iteration loop because it exists for every subfile entry as it is an input field. The subfile iteration loop is structured like this:

```
var iSFLIndex = 1;
var oSel = FIELDS("Sel",iSFLIndex);
while (oSel != null)
{
    if (oSel.getValue() == "C")
    {
        << Process selected subfile Entry >>
    }

    iSFLIndex += 1;
    oSel = FIELDS("Sel",iSFLIndex);
}
```

The loop starts at entry one and proceeds until no selection field can be found. Only entries selected with a "C" are processed.

This line removes the "C" from a selected line:

```
oSel.setValue(" ");
```

When iterating a subfile, the possibility exists that a blank outfield entry will not actually exist on the 5250 display. That is why the logic

```
    var oSurname = FIELDS("Surname",iSFLIndex);

    if (oSurname != null) sClipData += oSurname.getValue()   + sTab;
    else                  sClipData += "UNKNOWN" + sTab;
```

is used. It caters for the fact that an employee's surname may be blank and therefore the entry does not exist in the subfile.

The clipboard data is tab delimited. This makes it nice to paste into MS-Excel.

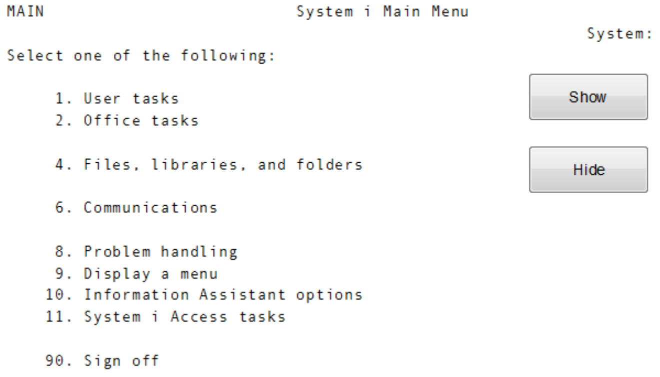
You would probably not code logic like this in a real application. It would be better to create a generic subfile clipboard function in your USERENV object and simply call it from the onClick routine.

Example - Hiding and Showing Fields on a Customized Screens

The starting point for this example is the System i Main Menu, where:

- The screen is named MAIN

- The selection or command field is named CommandLine.
- Two push button eXtensions, captioned Show and Hide, have been added.



```
MAIN                          System i Main Menu
                                                        System:
Select one of the following:

     1. User tasks
     2. Office tasks

     4. Files, libraries, and folders

     6. Communications

     8. Problem handling
     9. Display a menu
    10. Information Assistant options
    11. System i Access tasks

    90. Sign off
```

The *onClick* property of the Show push button is set to this:

```
var cmdfld= FIELDS("CommandLine");

cmdfld.setProperty("visible",true);

cmdfld.refresh();
```

The *onClick* property of the Hide push button is set to this:

```
var cmdfld= FIELDS("CommandLine");

cmdfld.setProperty("visible",false);

cmdfld.refresh();
```

When clicked they cause the selection or command field (CommandLine) to appear and disappear.

You can minimize this type of scripting by adding two generic functions like this to your USERENV object:

```
showField : function(env,name,index)
{
  var oField = env.FIELDS(name,index);
  if (oField != null)
   {
     oField.setProperty("visible",true);
     oField.refresh();
   }
},

hideField : function(env,name,index)
{
   var oField = env.FIELDS(name,index);
   if (oField != null)
   {
     oField.setProperty("visible",false);
     oField.refresh();
   }
},
```

Now the *onClick* property of the Show push button may be reduced to this:

```
USERENV.showField(ENV,"CommandLine");
```

and the *onClick* property of the Hide push button reduced to this:

```
USERENV.hideField(ENV,"CommandLine");
```

You now have generic hide and show functions you can use on any field.

Some other notes about this example:

- The ENV parameter passed into the hide and show functions as a way that it can operate in your scripts context/environment. For example, by passing ENV to the functions they can use the ENV.FIELDS() function.

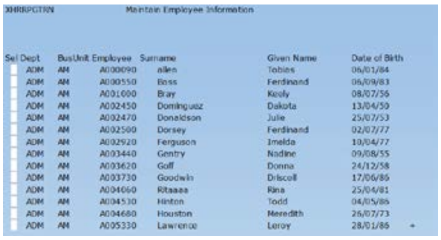
- The optional index parameter used by the show and hide functions is for when you are dealing with subfile fields. Pass it as the index of the subfile field.

- Don't forget to close and restart your aXes development session to pick up USERENV object definition changes.

Example – Dynamic Styling

Start Point

The shipped aXes demonstration screen named XHRRPGTRN_Select is used.

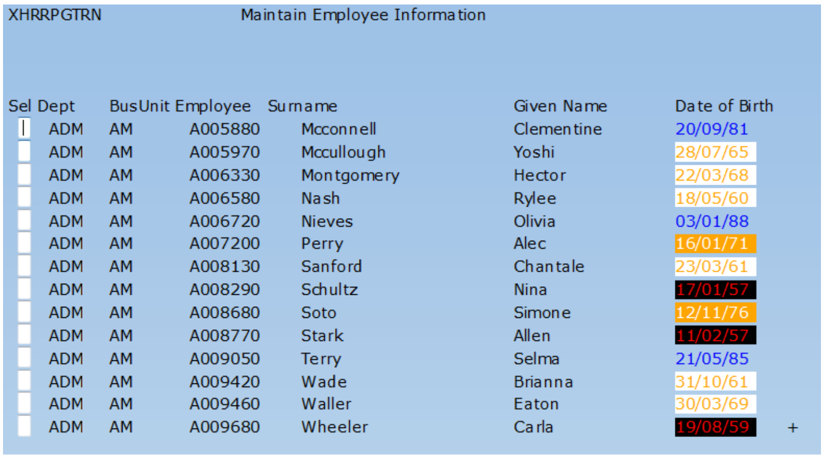


XHRRPGTRN Maintain Employee Information

Sel	Dept	BusUnit	Employee	Surname	Given Name	Date of Birth
☐	ADM	AM	A000090	allen	Tobias	06/01/84
☐	ADM	AM	A000550	Boss	Ferdinand	06/09/83
☐	ADM	AM	A001000	Bray	Keely	08/07/56
☐	ADM	AM	A002450	Dominguez	Dakota	13/04/50
☐	ADM	AM	A002470	Donaldson	Julie	25/07/53
☐	ADM	AM	A002500	Dorsey	Ferdinand	02/07/77
☐	ADM	AM	A002920	Ferguson	Imelda	10/04/77
☐	ADM	AM	A003440	Gentry	Nadine	09/08/55
☐	ADM	AM	A003620	Goff	Donna	24/12/58
☐	ADM	AM	A003730	Goodwin	Driscoll	17/06/86
☐	ADM	AM	A004060	Ritaaaa	Rina	25/04/81
☐	ADM	AM	A004530	Hinton	Todd	04/05/86
☐	ADM	AM	A004680	Houston	Meredith	26/07/73
☐	ADM	AM	A005330	Lawrence	Leroy	28/01/86 +

Result

The style of the Date of Birth Column is changed dynamically to highlight people born in the 50s, 60s, and 70s. Dynamic styling is often used to draw attention to special situations:



XHRRPGTRN Maintain Employee Information

Sel	Dept	BusUnit	Employee	Surname	Given Name	Date of Birth
☐	ADM	AM	A005880	Mcconnell	Clementine	20/09/81
☐	ADM	AM	A005970	Mccullough	Yoshi	28/07/65
☐	ADM	AM	A006330	Montgomery	Hector	22/03/68
☐	ADM	AM	A006580	Nash	Rylee	18/05/60
☐	ADM	AM	A006720	Nieves	Olivia	03/01/88
☐	ADM	AM	A007200	Perry	Alec	16/01/71
☐	ADM	AM	A008130	Sanford	Chantale	23/03/61
☐	ADM	AM	A008290	Schultz	Nina	17/01/57
☐	ADM	AM	A008680	Soto	Simone	12/11/76
☐	ADM	AM	A008770	Stark	Allen	11/02/57
☐	ADM	AM	A009050	Terry	Selma	21/05/85
☐	ADM	AM	A009420	Wade	Brianna	31/10/61
☐	ADM	AM	A009460	Waller	Eaton	30/03/69
☐	ADM	AM	A009680	Wheeler	Carla	19/08/59 +

Note: The color scheme here is only being used to demonstrate a technique.

Steps

❖ The Date of Birth element was selected and these properties were set in its *Default Visualization* eXtension:

style (as script)

```
var sDecade = FIELD.getValue().charAt(6);

var oStyle  = null;

switch (sDecade)
{
   case "5": oStyle = {"color":"red",    "background":"black"};  break;
   case "6": oStyle = {"color":"orange","background":"white"};  break;
   case "7": oStyle = {"color":"white", "background":"orange"}; break;
   default : oStyle = {"color":"blue"}; break;
}

ENV.returnValue = oStyle;
```

Observations

Dynamically evaluated styles need to return a JavaScript object containing the style property name and its value.

The preceding code does this by dynamically manufacturing an object containing the required style properties.

If this code was added to the USERENV object:

```
/* ---------------------------------------------------------------------
/* The USERENV object contains all customer defined shared scripts and
/* ---------------------------------------------------------------------

var USERENV =
{
   o50DecadeStyle  : {"color":"red",    "background":"black"},
   o60DecadeStyle  : {"color":"orange","background":"white"},
   o70DecadeStyle  : {"color":"white", "background":"orange"},
   oDftDecadeStyle : {"color":"blue"},
```

and the preceding dynamic styling code was changed to this:

```
var sDecade = FIELD.getValue().charAt(6);
switch (sDecade)
{
   case "5": ENV.returnValue = USERENV.o50DecadeStyle; break;
   case "6": ENV.returnValue = USERENV.o60DecadeStyle; break;
   case "7": ENV.returnValue = USERENV.o70DecadeStyle; break;
   default : ENV.returnValue = USERENV.oDftDecadeStyle; break;;
}
```

then the result would be improved for two reasons:

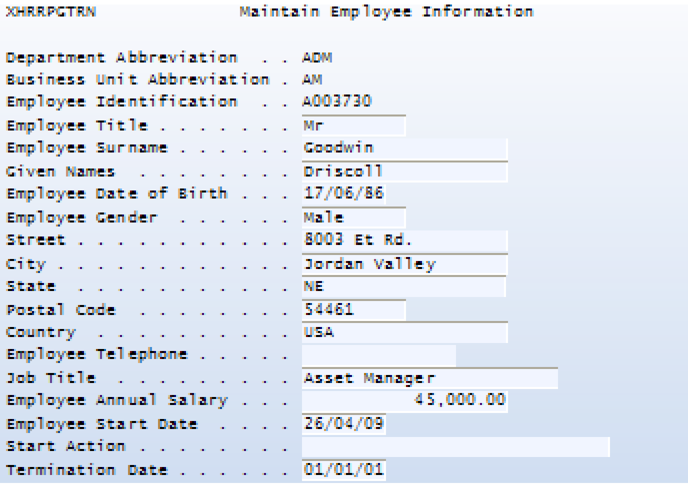
- The styles have been externalized for a single point of change.
- The code is considerably more efficient because the returned styles objects do not have to be repeatedly created. Only the 4 decade styles in USERENV ever have to be created.

Note: If you try this approach out - remember that you have to signoff, then close and reopen the browser window for any changes to USERENV to be loaded. USERENV is only loaded as the aXes terminal session is started.

Example – Dynamically refreshing a drop down without server interaction

Start Point

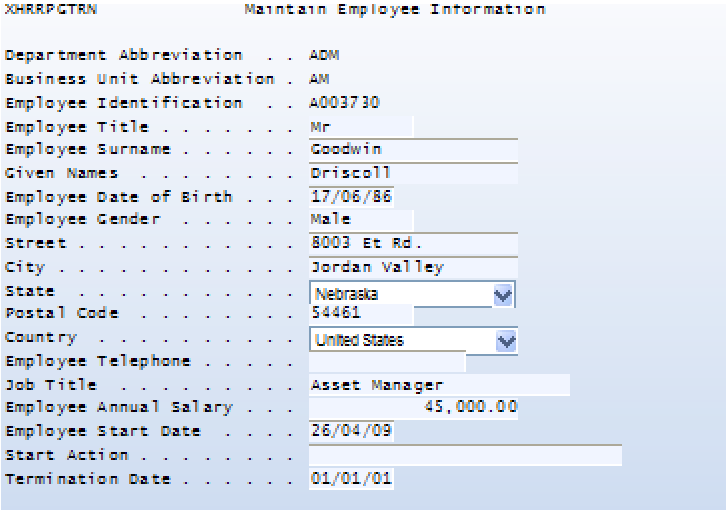
The shipped aXes demonstration screen named XHRRPGTRN_Maint is used.
The *State* field is named Employee_State and the *Country* field is Employee_Country:



```
XHRRPGTRN                Maintain Employee Information

Department Abbreviation  . .  ADM
Business Unit Abbreviation .  AM
Employee Identification  . .  A003730
Employee Title . . . . . . .  Mr
Employee Surname . . . . . .  Goodwin
Given Names  . . . . . . . .  Driscoll
Employee Date of Birth . . .  17/06/86
Employee Gender  . . . . . .  Male
Street . . . . . . . . . . .  8003 Et Rd.
City . . . . . . . . . . . .  Jordan Valley
State  . . . . . . . . . . .  NE
Postal Code  . . . . . . . .  54461
Country  . . . . . . . . . .  USA
Employee Telephone . . . . .
Job Title  . . . . . . . . .  Asset Manager
Employee Annual Salary . . .            45,000.00
Employee Start Date  . . . .  26/04/09
Start Action . . . . . . . .
Termination Date . . . . . .  01/01/01
```

Result

Employee_State and Employee_Country are replaced with drop downs that source its data from static tables. Employee_State is sensitive to the value of Employee_Country so that it will show the states for the selected country without server interaction:



```
XHRRPGTRN                Maintain Employee Information

Department Abbreviation  . .  ADM
Business Unit Abbreviation .  AM
Employee Identification  . .  A003730
Employee Title . . . . . . .  Mr
Employee Surname . . . . . .  Goodwin
Given Names  . . . . . . . .  Driscoll
Employee Date of Birth . . .  17/06/86
Employee Gender  . . . . . .  Male
Street . . . . . . . . . . .  8003 Et Rd.
City . . . . . . . . . . . .  Jordan Valley
State  . . . . . . . . . . .  Nebraska
Postal Code  . . . . . . . .  54461
Country  . . . . . . . . . .  United States
Employee Telephone . . . . .
Job Title  . . . . . . . . .  Asset Manager
Employee Annual Salary . . .            45,000.00
Employee Start Date  . . . .  26/04/09
Start Action . . . . . . . .
Termination Date . . . . . .  01/01/01
```

Steps

Edit tables_static.txt and add the following table of US and Australian states:

```
DefineObjectInstance {

  className = "StaticTable",
  name    = "US_OZ_State",
  source  = "inline",
  rows = {
        {value="AK",text="Alaska", countryCode="USA"},
        {value="AL",text="Alabama", countryCode="USA"},
        {value="AR",text="Arkansas", countryCode="USA"},
        {value="AS",text="American Samoa", countryCode="USA"},
        {value="AZ",text="Arizona", countryCode="USA"},
        {value="CA",text="California", countryCode="USA"},
        {value="CO",text="Colorado", countryCode="USA"},
        {value="CT",text="Connecticut", countryCode="USA"},
        {value="DC",text="District of Columbia", countryCode="USA"},
        {value="DE",text="Delaware", countryCode="USA"},
        {value="FL",text="Florida", countryCode="USA"},
        {value="GA",text="Georgia", countryCode="USA"},
        {value="GU",text="Guam", countryCode="USA"},
        {value="HI",text="Hawaii", countryCode="USA"},
        {value="IA",text="Iowa", countryCode="USA"},
        {value="ID",text="Idaho", countryCode="USA"},
        {value="IL",text="Illinois", countryCode="USA"},
        {value="IN",text="Indiana", countryCode="USA"},
        {value="KS",text="Kansas", countryCode="USA"},
        {value="KY",text="Kentucky", countryCode="USA"},
        {value="LA",text="Louisiana", countryCode="USA"},
        {value="MA",text="Massachusetts", countryCode="USA"},
        {value="MD",text="Maryland", countryCode="USA"},
        {value="ME",text="Maine", countryCode="USA"},
        {value="MI",text="Michigan", countryCode="USA"},
        {value="MN",text="Minnesota", countryCode="USA"},
        {value="MO",text="Missouri", countryCode="USA"},
        {value="MP",text="Northern Mariana Islands", countryCode="USA"},
        {value="MS",text="Mississippi", countryCode="USA"},
        {value="MT",text="Montana", countryCode="USA"},
        {value="NC",text="North Carolina", countryCode="USA"},
        {value="ND",text="North Dakota", countryCode="USA"},
        {value="NE",text="Nebraska", countryCode="USA"},
        {value="NH",text="New Hampshire", countryCode="USA"},
        {value="NJ",text="New Jersey", countryCode="USA"},
        {value="NM",text="New Mexico", countryCode="USA"},
        {value="NV",text="Nevada", countryCode="USA"},
        {value="NY",text="New York", countryCode="USA"},
        {value="OH",text="Ohio", countryCode="USA"},
        {value="OK",text="Oklahoma", countryCode="USA"},
        {value="OR",text="Oregon", countryCode="USA"},
        {value="PA",text="Pennsylvania", countryCode="USA"},
        {value="PR",text="Puerto Rico", countryCode="USA"},
        {value="RI",text="Rhode Island", countryCode="USA"},
        {value="SC",text="South Carolina", countryCode="USA"},
        {value="SD",text="South Dakota", countryCode="USA"},
        {value="TN",text="Tennessee", countryCode="USA"},
        {value="TX",text="Texas", countryCode="USA"},
        {value="UM",text="United States Minor Outlying Islands", countryCode="USA"},
        {value="UT",text="Utah", countryCode="USA"},
        {value="VA",text="Virginia", countryCode="USA"},
        {value="VI",text="Virgin Islands", countryCode="USA"},
        {value="VT",text="Vermont", countryCode="USA"},
        {value="WA",text="Washington", countryCode="USA"},
        {value="WI",text="Wisconsin", countryCode="USA"},
        {value="WV",text="West Virginia", countryCode="USA"},
        {value="WY",text="Wyoming", countryCode="USA"},
        {value="ACT",text="Canberra", countryCode="AUS"},
        {value="NSW",text="New South Wales", countryCode="AUS"},
        {value="QLD",text="Queensland", countryCode="AUS"},
        {value="NT",text="Northern Territory", countryCode="AUS"},
        {value="SA",text="South Australia", countryCode="AUS"},
        {value="VIC",text="Victoria", countryCode="AUS"},
        {value="WA",text="Western Australia", countryCode="AUS"},
        {value="TAS",text="Tasmania", countryCode="AUS"},
     },
};
```

- Employee_State was selected and changed to be a Drop Down.

- These properties were then set in the Drop Down eXtension:

  dataSourceType:

  Static Table

tableName:

US_OZ_State

onFillDropDown (script):

```
/* Get the current country */
var currentCountry = FIELDS("Employee_Country").getValue();

/* Only add entries to the drop down which match the value of the country field */
if (currentCountry == ROW.countryCode)
{
   ENV.returnValue = ROW.text;
}
else
{
   ENV.returnValue = null;
}
```

- Employee_Country was selected and changed to be a Drop Down
- These properties were then set in the Drop Down eXtension:

dataSourceType:

Static Table

tableName:

ISOCountry

onFillDropDown (script):

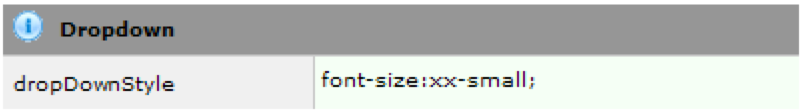
```
/* For the purpose of this exercise, filter the countries so that we only
add Australia and USA */


var country = ROW.value;

if ( (country == "AUS" ) || (country == "USA" ) )
{
  ENV.returnValue = ROW.text;
}
else
{
  ENV.returnValue = null;
}
```

onSelectValueChanged (script):

```
/* Set the country field value first and then refresh the state drop down.
Refreshing the state drop down will cause it to be reloaded from the table
data triggering the onFillDropDown event. Because now the country has
changed, the state drop down will only fill the states according to the
changed value of country */


FIELD.setValue(ROW.value);
FIELDS("Employee_State").refresh();
```

Remarks:

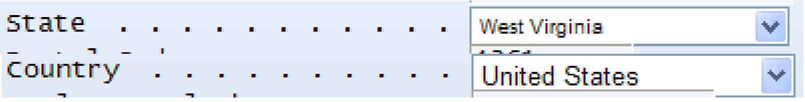
To avoid the Drop Down overlapping part of the input fields you can set the font size in the Drop Down's style property. For example, you can set it to **xx-small:**



Dropdown

dropDownStyle	font-size:xx-small;

In this picture we changed the size of the States but not the Country:

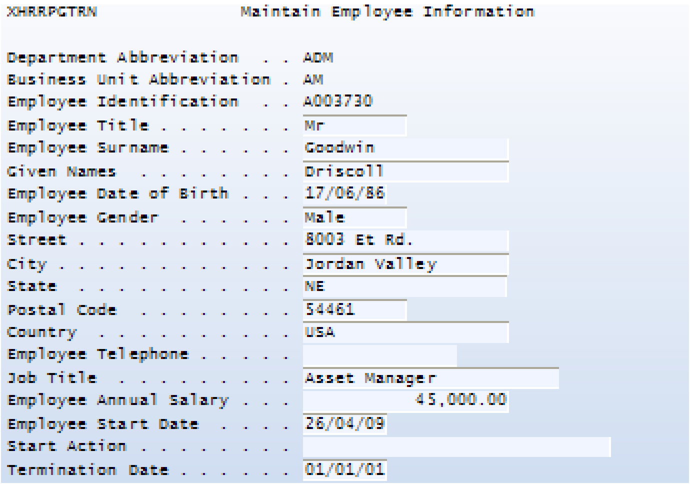


State . . . . West Virginia

Country . . . . United States

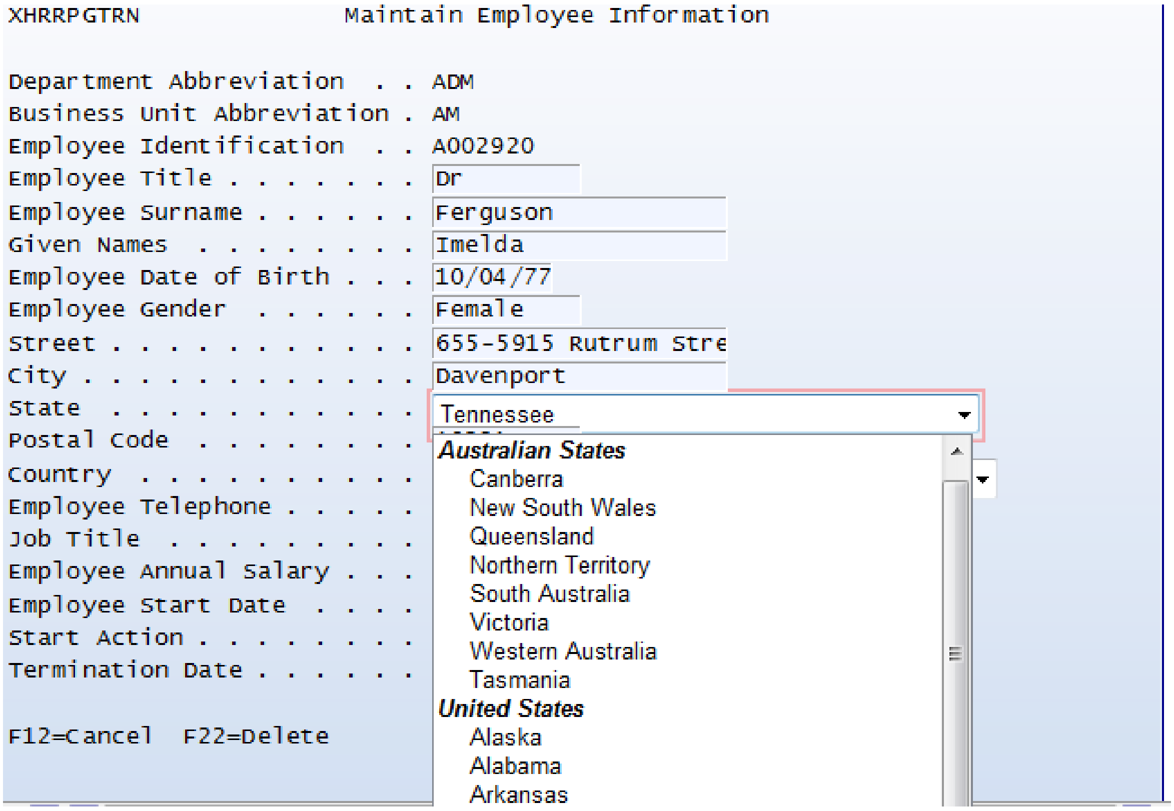
Example – Two level drop down

Start Point
The shipped aXes demonstration screen named XHRRPGTRN_Maint is used.
The *State* field is Employee_State:



```
XHRRPGTRN                Maintain Employee Information

Department Abbreviation  . . ADM
Business Unit Abbreviation . AM
Employee Identification  . . A003730
Employee Title . . . . . . . Mr
Employee Surname . . . . . . Goodwin
Given Names  . . . . . . . . Driscoll
Employee Date of Birth . . . 17/06/86
Employee Gender  . . . . . . Male
Street . . . . . . . . . . . 8003 Et Rd.
City . . . . . . . . . . . . Jordan Valley
State  . . . . . . . . . . . NE
Postal Code  . . . . . . . . 54461
Country  . . . . . . . . . . USA
Employee Telephone . . . . .
Job Title  . . . . . . . . . Asset Manager
Employee Annual Salary . . .            45,000.00
Employee Start Date  . . . . 26/04/09
Start Action . . . . . . . .
Termination Date . . . . . . 01/01/01
```

Result
States appear as children of the Country:



```
XHRRPGTRN                Maintain Employee Information

Department Abbreviation  . . ADM
Business Unit Abbreviation . AM
Employee Identification  . . A002920
Employee Title . . . . . . . Dr
Employee Surname . . . . . . Ferguson
Given Names  . . . . . . . . Imelda
Employee Date of Birth . . . 10/04/77
Employee Gender  . . . . . . Female
Street . . . . . . . . . . . 655-5915 Rutrum Stre
City . . . . . . . . . . . . Davenport
State  . . . . . . . . . . . Tennessee
Postal Code  . . . . . . . .
Country  . . . . . . . . . .
Employee Telephone . . . . .
Job Title  . . . . . . . . .
Employee Annual Salary . . .
Employee Start Date  . . . .
Start Action . . . . . . . .
Termination Date . . . . . .

F12=Cancel  F22=Delete
```

Steps

Use the same table as the one used for the ***Dynamically refreshing a drop down without server interaction*** example.
Edit static_tables.txt and add two rows:

One before the first row containing the Australian States and another one before the first row of US States:

```
        {optgroup="Australian States"},
        {value="ACT",text="Canberra", countryCode="AUS"},
        {value="NSW",text="New South Wales", countryCode="AUS"},
        {value="QLD",text="Queensland", countryCode="AUS"},
        {value="NT",text="Northern Territory", countryCode="AUS"},
        {value="SA",text="South Australia", countryCode="AUS"},
        {value="VIC",text="Victoria", countryCode="AUS"},
        {value="WA",text="Western Australia", countryCode="AUS"},
        {value="TAS",text="Tasmania", countryCode="AUS"},
        {optgroup="United States"},
        {value="AK",text="Alaska", countryCode="USA"},
        {value="AL",text="Alabama", countryCode="USA"},
        {value="AR",text="Arkansas", countryCode="USA"},
        {value="AS",text="American Samoa", countryCode="USA"},
        {value="AZ",text="Arizona", countryCode="USA"},
        {value="CA",text="California", countryCode="USA"},
        {value="CO",text="Colorado", countryCode="USA"},
        {value="CT",text="Connecticut", countryCode="USA"},
        {value="DC",text="District of Columbia", countryCode="USA"},
        {value="DE",text="Delaware", countryCode="USA"},
        {value="FL",text="Florida", countryCode="USA"},
        {value="GA",text="Georgia", countryCode="USA"},
        {value="GU",text="Guam", countryCode="USA"},
        {value="HI",text="Hawaii", countryCode="USA"},
        {value="IA",text="Iowa", countryCode="USA"},
        {value="ID",text="Idaho", countryCode="USA"},
        {value="IL",text="Illinois", countryCode="USA"},
        {value="IN",text="Indiana", countryCode="USA"},
        {value="KS",text="Kansas", countryCode="USA"},
        {value="KY",text="Kentucky", countryCode="USA"},
        {value="LA",text="Louisiana", countryCode="USA"},
        {value="MA",text="Massachusetts", countryCode="USA"},
        {value="MD",text="Maryland", countryCode="USA"},
        {value="ME",text="Maine", countryCode="USA"},
        {value="MI",text="Michigan", countryCode="USA"},
        {value="MN",text="Minnesota", countryCode="USA"},
        {value="MO",text="Missouri", countryCode="USA"},
        {value="MP",text="Northern Mariana Islands", countryCode="USA"},
        {value="MS",text="Mississippi", countryCode="USA"},
        {value="MT",text="Montana", countryCode="USA"},
        {value="NC",text="North Carolina", countryCode="USA"},
        {value="ND",text="North Dakota", countryCode="USA"},
        {value="NE",text="Nebraska", countryCode="USA"},
        {value="NH",text="New Hampshire", countryCode="USA"},
        {value="NJ",text="New Jersey", countryCode="USA"},
        {value="NM",text="New Mexico", countryCode="USA"},
        {value="NV",text="Nevada", countryCode="USA"},
        {value="NY",text="New York", countryCode="USA"},
        {value="OH",text="Ohio", countryCode="USA"},
        {value="OK",text="Oklahoma", countryCode="USA"},
        {value="OR",text="Oregon", countryCode="USA"},
        {value="PA",text="Pennsylvania", countryCode="USA"},
        {value="PR",text="Puerto Rico", countryCode="USA"},
        {value="RI",text="Rhode Island", countryCode="USA"},
        {value="SC",text="South Carolina", countryCode="USA"},
        {value="SD",text="South Dakota", countryCode="USA"},
        {value="TN",text="Tennessee", countryCode="USA"},
        {value="TX",text="Texas", countryCode="USA"},
        {value="UM",text="United States Minor Outlying Islands", countryCode="USA"},
        {value="UT",text="Utah", countryCode="USA"},
        {value="VA",text="Virginia", countryCode="USA"},
        {value="VI",text="Virgin Islands", countryCode="USA"},
        {value="VT",text="Vermont", countryCode="USA"},
        {value="WA",text="Washington", countryCode="USA"},
        {value="WI",text="Wisconsin", countryCode="USA"},
        {value="WV",text="West Virginia", countryCode="USA"},
        {value="WY",text="Wyoming", countryCode="USA"},
    },
};
```

- Employee_State was selected and changed to be a Drop Down.

- These properties were then set in the Drop Down eXtension:

dataSourceType:

Static Table

tableName:

US_OZ_State

onFillDropDown (script):

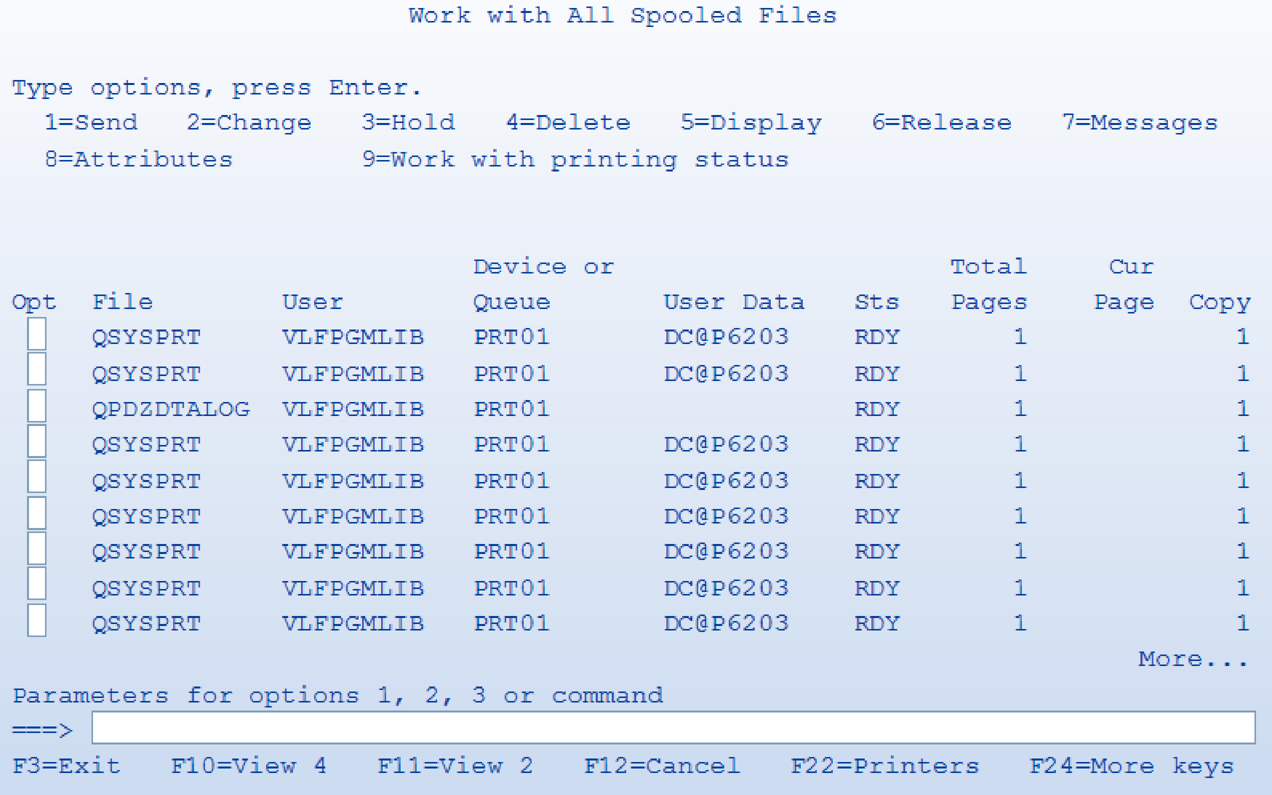
```
if (ROW.optgroup != null)
{
   ENV.returnValue = null;
}
else
{
   ENV.returnValue = ROW.text;
}
```

An example of achieving the same but with the data sourced from an xml file:

```
<?xml version="1.0" encoding="iso-8859-1"?>

<states>
   <row>
      <optgroup>US States</optgroup>
   </row>
   <row>
    <code>AK</code>
    <desc >Alaska</desc>
   </row>
   <row>
    <code>NE</code>
    <desc>Nebraska</desc>
   </row>
   <row>
      <optgroup>Australian States</optgroup>
   </row>
   <row>
    <code>ACT</code>
    <desc>Canberra</desc>
   </row>
   <row>
    <code>QLD</code>
    <desc>Queensland</desc>
   </row>
</states>
```

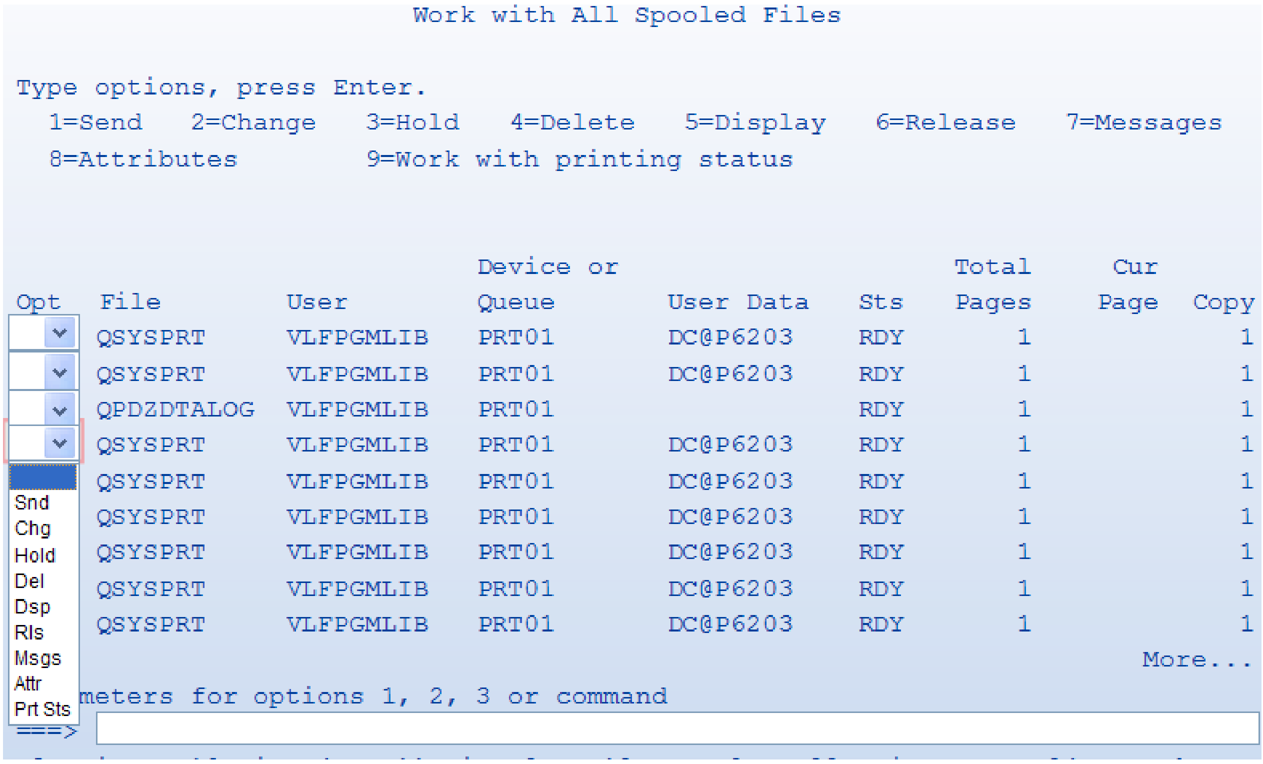
Example – Using a drop down as subfile option field

Start Point
A screen like Work With Spool Files:



```
                        Work with All Spooled Files

Type options, press Enter.
  1=Send    2=Change    3=Hold    4=Delete    5=Display    6=Release    7=Messages
  8=Attributes          9=Work with printing status


                             Device or                        Total     Cur
Opt  File         User         Queue        User Data    Sts    Pages    Page   Copy
 □   QSYSPRT      VLFPGMLIB    PRT01        DC@P6203     RDY        1              1
 □   QSYSPRT      VLFPGMLIB    PRT01        DC@P6203     RDY        1              1
 □   QPDZDTALOG   VLFPGMLIB    PRT01                     RDY        1              1
 □   QSYSPRT      VLFPGMLIB    PRT01        DC@P6203     RDY        1              1
 □   QSYSPRT      VLFPGMLIB    PRT01        DC@P6203     RDY        1              1
 □   QSYSPRT      VLFPGMLIB    PRT01        DC@P6203     RDY        1              1
 □   QSYSPRT      VLFPGMLIB    PRT01        DC@P6203     RDY        1              1
 □   QSYSPRT      VLFPGMLIB    PRT01        DC@P6203     RDY        1              1
 □   QSYSPRT      VLFPGMLIB    PRT01        DC@P6203     RDY        1              1
                                                                               More...
Parameters for options 1, 2, 3 or command
===> ____________________________________________________________________________
F3=Exit   F10=View 4   F11=View 2   F12=Cancel   F22=Printers   F24=More keys
```

Result
The option fields are turned into drop downs to make it simpler to select.



```
                        Work with All Spooled Files

Type options, press Enter.
  1=Send    2=Change    3=Hold    4=Delete    5=Display    6=Release    7=Messages
  8=Attributes          9=Work with printing status


                             Device or                       Total     Cur
Opt  File         User         Queue        User Data    Sts   Pages     Page    Copy
     QSYSPRT      VLFPGMLIB    PRT01        DC@P6203     RDY       1                1
     QSYSPRT      VLFPGMLIB    PRT01        DC@P6203     RDY       1                1
     QPDZDTALOG   VLFPGMLIB    PRT01                     RDY       1                1
     QSYSPRT      VLFPGMLIB    PRT01        DC@P6203     RDY       1                1
     QSYSPRT      VLFPGMLIB    PRT01        DC@P6203     RDY       1                1
     QSYSPRT      VLFPGMLIB    PRT01        DC@P6203     RDY       1                1
     QSYSPRT      VLFPGMLIB    PRT01        DC@P6203     RDY       1                1
     QSYSPRT      VLFPGMLIB    PRT01        DC@P6203     RDY       1                1
     QSYSPRT      VLFPGMLIB    PRT01        DC@P6203     RDY       1                1
                                                                            More...
    meters for options 1, 2, 3 or command
===>
```

Steps

1) Add this table definition to your tables_static.txt:

```
DefineObjectInstance {

    className = "StaticTable",
    name     = "WrkSplF_Options",
    source   = "inline",
    rows = {
              {value="",text=""},
              {value="1",text="Snd"},
              {value="2",text="Chg"},
              {value="3",text="Hold"},
              {value="4",text="Del"},
              {value="5",text="Dsp"},
              {value="6",text="Rls"},
              {value="7",text="Msgs"},
              {value="8",text="Attr"},
              {value="9",text="Prt Sts"},
           },

    };
```

2) Repeat this step for each of the Opt fields **ONLY IN THE FIRST PAGE** of the screen:

- Make the field a Drop Down and size it appropiately
- Set the *dataSourceType* to Static Table and the *tableName* to WrkSplF_Options
- Set *onSelectValueChanged* to

  if (ROW.value != "") {FIELD.setValue(ROW.value); SENDKEY("Enter");

3) Save the screen.

You can now use the drop down options to work with a file.

Note that in a real application you will need to hide the instructions on the top of the display because they are no longer valid.

Example – Dynamic Google Chart 1

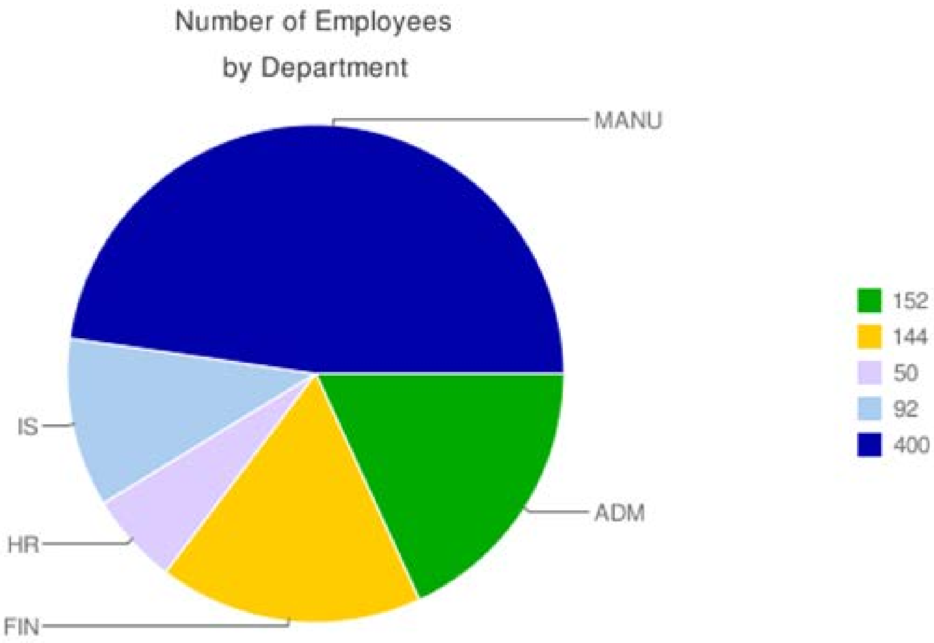
By using this extension you will be subject to the Google Terms of Service as outlined in http://code.google.com/apis/chart/tems.html

Start Point

You can start with any screen because charts are not associated with fields. For the sake of this example we will start with a blank screen.

Result

A Pie Google Chart showing the number of employees by department. They use data from static tables that were created by direct scripting (instead of being loaded from the server):



Steps

Add this table definition to your tables_static.txt:

```
DefineObjectInstance {
       className          = "StaticTable",
       name               = "XHREmployee",
       source             = "sql",
       selectSQLcommand   = "XHRDEPCDE from AXESDEMO.XHREMPTN order by XHRDEPCDE",
       resultColumnNames  = {"deptCode"},
};
```

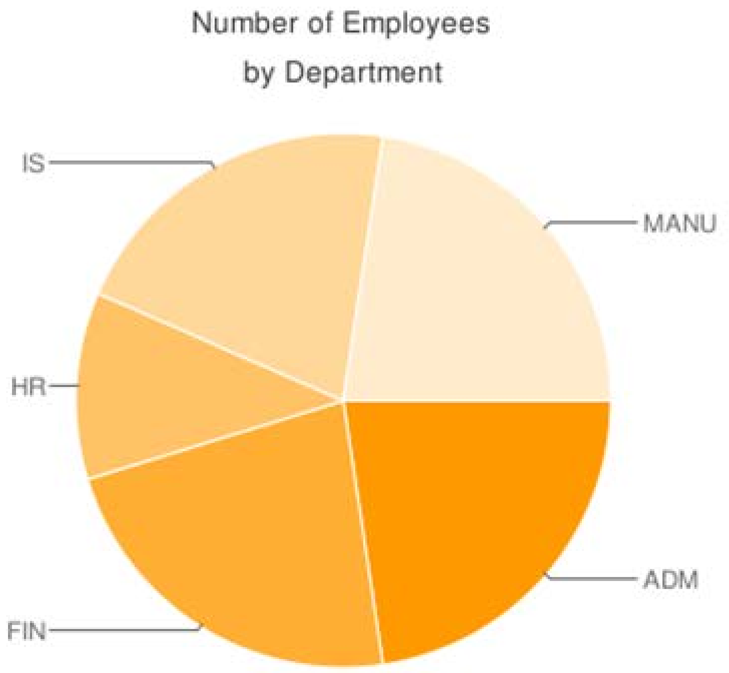
Add a new element, set its extension type as Google Chart and size it.
Now set these extension properties:

tableName: Employees_in_Dept
onLoadChart – intermediate version:

```
/* Load the static tables file */
TABLEMANAGER.loadStaticTables(USERENV.staticTablesFile);

/* The department table returned by the selectSQL filled with departments */
var deptTable = TABLEMANAGER.getTable("XHREmployee");

/* The table to be filled manually based on the department table */
var chartTable = TABLEMANAGER.getTable("Employees_in_Dept");

/* Get the total number of table rows which is the same as the total number of employee */
var iRows = deptTable.childCount();

var savDept = "";
var iEmployeeInDept = 0;
var oChild;

/* Traverse the static table, count the number of employees for each department and insert a
child row into the table nominated in the tableName property whenever a change of department
is detected. Note the order by in the selectSQL statement makes sure the records are ordered
by the department code */

for (var i = 0; i < iRows; i++)
{
 oChild = deptTable.child(i);
 if (oChild.deptCode != savDept)
 {
   if (savDept != "")
   {
     chartTable.insertChild({chartData:iEmployeeInDept.toString(),chartLabel:savDept});
   }
   savDept = oChild.deptCode;
   iEmployeeInDept = 1;
 }
 else
 {
    iEmployeeInDept++;
 }
}
```

Title: Number+of+Employees|by+Department

INTERMEDIATE RESULT 1:



To add colors you could add the RGB hex color codes to the colors property. In this case because there are 5 departments you would add 5 color codes like this:

colors: 00AA00,FFCC00,DDCCFF,ABCDEF,0000AA

However, since we don't really know how many departments until the query is executed, the proper way to do it is inside the for loop when the department changes. To do this we need an array of colors to select from, a positional index, a variable to store the entire color string and a variable for the comma separator:

```
/* array of colors to set to the different departments. It should contain enough colors so
that no color is repeated.  */
var arrayColors = ["00AA00", "FFCC00" , "DDCCFF", "ABCDEF", "0000AA", "84871C", "FF9900"];
var iColorIndex = 0;
var chco = "";
var colorSep = "";
```

Now we modify the logic inside the `for` loop to start building the color string:

```
/* Load the static tables file */
TABLEMANAGER.loadStaticTables(USERENV.staticTablesFile);

/* The department table returned by the selectSQL filled with departments */
var deptTable = TABLEMANAGER.getTable("XHREmployee");

/* The table to be filled manually based on the department table */
var chartTable = TABLEMANAGER.getTable("Employees_in_Dept");

/* Get the total number of table rows which is the same as the total number of employee */
var iRows = deptTable.childCount();

var savDept = "";
var iEmployeeInDept = 0;
var oChild;

/* Traverse the static table, count the number of employees for each department and insert a
child row into the table nominated in the tableName property whenever a change of department
is detected. Note the order by in the selectSQL statement makes sure the records are ordered
by the department code */

for (var i = 0; i < iRows; i++)
{
 oChild = deptTable.child(i);
 if (oChild.deptCode != savDept)
 {
   if (savDept != "")
   {
     chartTable.insertChild({chartData:iEmployeeInDept.toString(),chartLabel:savDept});

     /* Make sure we are not passed the last color and if we have start from the first one */
     if (iColorIndex >= arrayColors.length)
     {
       iColorIndex = 0;
     }
     chco += colorSep + arrayColors[iColorIndex];
     iColorIndex++;
     /* Change the color separator so from now one it puts a comma before the next color */
     colorSep = ",";
   }
   savDept = oChild.deptCode;
   iEmployeeInDept = 1;
 }
 else
 {
    iEmployeeInDept++;
 }
}
```

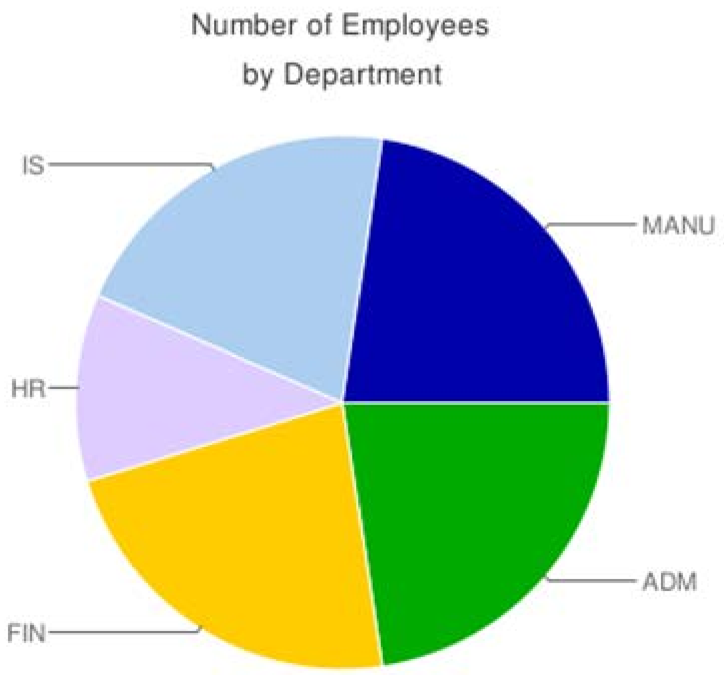
Then we must set the **colors** property:

```
FIELD.setProperty("colors", chco);
```

The entire script should now look something like this:

```
/* Load the static tables file */
TABLEMANAGER.loadStaticTables(USERENV.staticTablesFile);

/* The department table returned by the selectSQL filled with departments */
var deptTable = TABLEMANAGER.getTable("XHREmployee");

/* The table to be filled manually based on the department table */
var chartTable = TABLEMANAGER.getTable("Employees_in_Dept");

/* Get the total number of table rows which is the same as the total number of employee */
var iRows = deptTable.childCount();

var savDept = "";
var iEmployeeInDept = 0;
var oChild;

/* array of colors to set to the different departments. It should contain enough colors so
that no color is repeated.  */
var arrayColors = ["00AA00", "FFCC00" , "DDCCFF", "ABCDEF", "0000AA", "84871C", "FF9900"];
var iColorIndex = 0;
var chco = "";
var colorSep = "";

/* Traverse the static table, count the number of employees for each department and insert a
child row into the table nominated in the tableName property whenever a change of department
is detected. Note the order by in the selectSQL statement makes sure the records are ordered
by the department code */

for (var i = 0; i < iRows; i++)
{
  oChild = deptTable.child(i);
  if (oChild.deptCode != savDept)
  {
    if (savDept != "")
    {
      chartTable.insertChild({chartData:iEmployeeInDept.toString(),chartLabel:savDept});
      /* Make sure we are not passed the last color and if we have start from the first one
*/
      if (iColorIndex >= arrayColors.length)
      {
        iColorIndex = 0;
      }
      chco += colorSep + arrayColors[iColorIndex];
      iColorIndex++;
      /* Change the color separator so from now one it puts a comma before the next color */
      colorSep = ",";
    }
    savDept = oChild.deptCode;
    iEmployeeInDept = 1;
   }
   else
   {
     iEmployeeInDept++;
   }
}

FIELD.setProperty("colors", chco);
```

INTERMEDIATE RESULT 2:



The next modification is to make the chart show chart **legends** to show the number of employees in each department. The logic is very similar to the one used for the colors ...

```
var chdl = "";
var chdlSep = "";
```

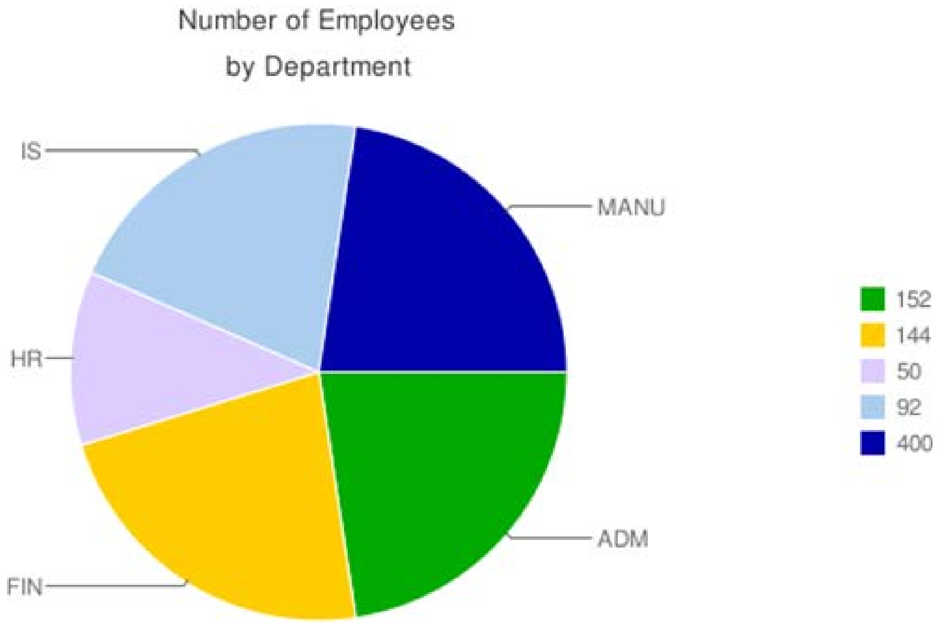
... store the number of employees in the for loop ...

```
for (var i = 0; i < iRows; i++)
{
 oChild = deptTable.child(i);
 if (oChild.deptCode != savDept)
 {
   if (savDept != "")
   {
     chartTable.insertChild({chartData:iEmployeeInDept.toString(),chartLabel:savDept});

     /* Make sure we are not passed the last color and if we have start from the first one */
     if (iColorIndex >= arrayColors.length)
     {
       iColorIndex = 0;
     }
     chco += colorSep + arrayColors[iColorIndex];
     iColorIndex++;
     /* Change the color separator so from now one it puts a comma before the next color */
     colorSep = ",";
     /* Legends string with the number of employees. The separator here is the pipe char */
     chdl += chdlSep + iEmployeeInDept.toString();
     chdlSep = "|";

   }
   savDept = oChild.deptCode;
   iEmployeeInDept = 1;
 }
 else
 {
    iEmployeeInDept++;
 }
}
```

... and set the property

```
FIELD.setProperty("legends", chdl);
```

INTERMEDIATE RESULT 3:



Note that Google Chart doesn't scale the data by default. Hence any value greater than 100 appears as 100. In the above chart there is no visual clue that for example the MANU department has more than four times the number of employees than IS (400 vs. 92). The final step then is to make the chart reflect the real proportions. First add a variable:

```
var percent;
```

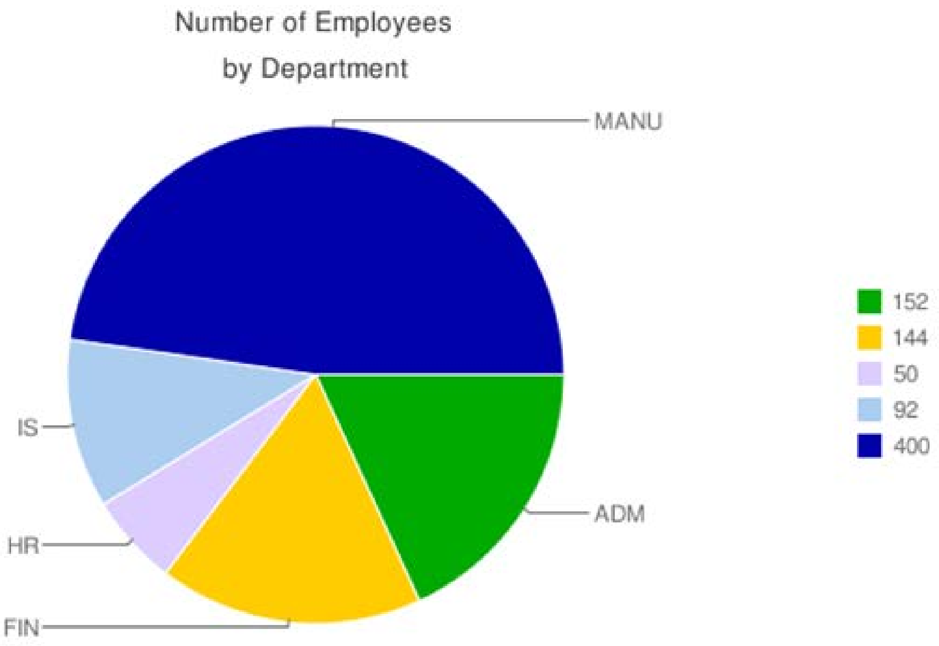
Then make the calculation and modify the `insertChild()` call to use the calculated value:

```
percent = (iEmployeeInDept * 100) / iRows;
chartTable.insertChild({chartData:percent.toString(),chartLabel:savDept});
```

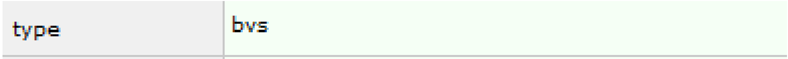
The final version of the script should look something like this:

```
/* Load the static tables file */
TABLEMANAGER.loadStaticTables(USERENV.staticTablesFile);

/* The department table returned by the selectSQL filled with departments */
var deptTable = TABLEMANAGER.getTable("XHREmployee");

/* The table to be filled manually based on the department table */
var chartTable = TABLEMANAGER.getTable("Employees_in_Dept");

/* Get the total number of table rows which is the same as the total number of employee */
var iRows = deptTable.childCount();

var savDept = "";
var iEmployeeInDept = 0;
var oChild;

/* array of colors to set to the different departments. It should contain enough colors so
that no color is repeated.  */
var arrayColors = ["00AA00", "FFCC00" , "DDCCFF", "ABCDEF", "0000AA", "84871C", "FF9900"];
var iColorIndex = 0;
var chco = "";
var colorSep = "";
var chdl = "";
var chdlSep = "";
var percent;
/* Traverse the static table, count the number of employees for each department and insert a
child row into the table nominated in the tableName property whenever a change of department
is detected. Note the order by in the selectSQL statement makes sure the records are ordered
by the department code */

for (var i = 0; i < iRows; i++)
{
 oChild = deptTable.child(i);
 if (oChild.deptCode != savDept)
 {
   if (savDept != "")
   {
     percent = (iEmployeeInDept * 100) / iRows;
     chartTable.insertChild({chartData:percent.toString(),chartLabel:savDept});
     /* Make sure we are not passed the last color and if we have start from the first one */
     if (iColorIndex >= arrayColors.length)
     {
       iColorIndex = 0;
     }
     chco += colorSep + arrayColors[iColorIndex];
     iColorIndex++;
     /* Change the color separator so from now one it puts a comma before the next color */
     colorSep = ",";
     chdl += chdlSep + iEmployeeInDept.toString();
     chdlSep = "|";
   }
   savDept = oChild.deptCode;
   iEmployeeInDept = 1;
 }
 else
 {
    iEmployeeInDept++;
 }
}

FIELD.setProperty("colors", chco);
FIELD.setProperty("legends", chdl);
```

FINAL RESULT:

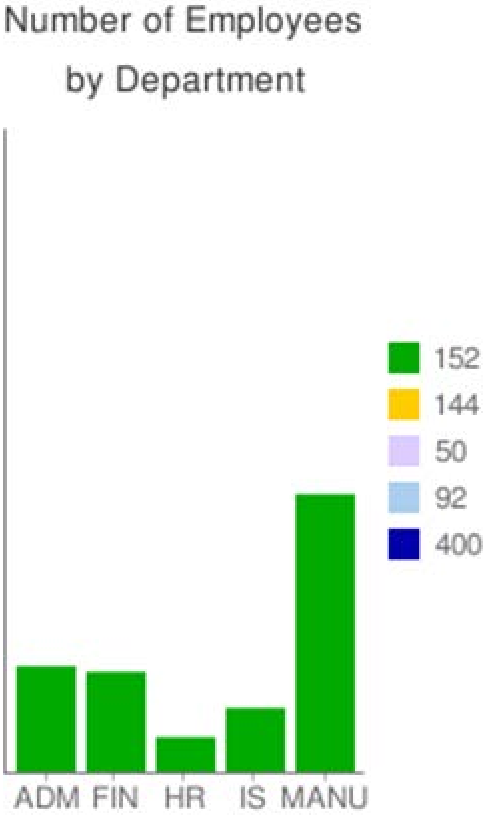


To visualize the PIE chart as a BAR one, change the **type** property from **p** to **bvs:**



type	bvs

The chart should now look something like this:



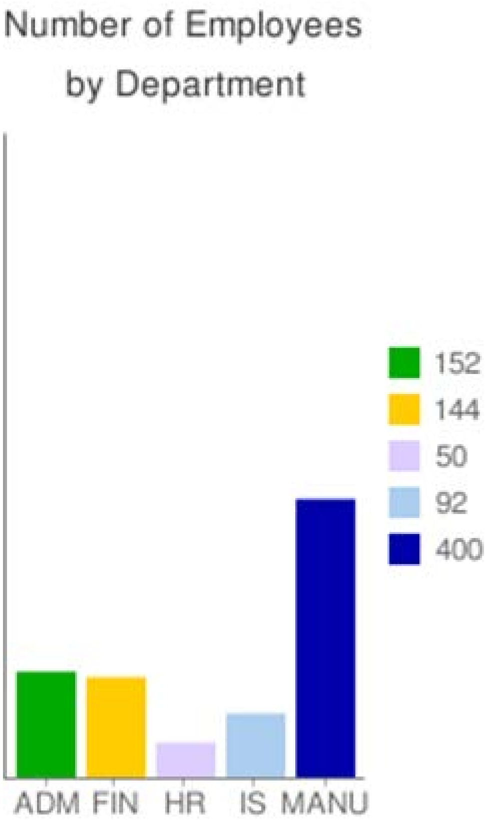
The colors are all the same because for this type of chart, the character used to separate the colors inside the color string (**chco=**) is a pipe char instead of a comma.

In the **onLoadChart** script modify this line

```
colorSep = ",";
```

To this line

```
colorSep = "|";
```



Observations

To make charting more flexible and reusable it is strongly recommended to create a `function` in USERENV.js so that you are able to parameterize variables according to the chart type. Using the above as an example, you could pass this function a color separator ("," or "|") so that you don't have to change your script.

Example – Dynamic Google Chart 2

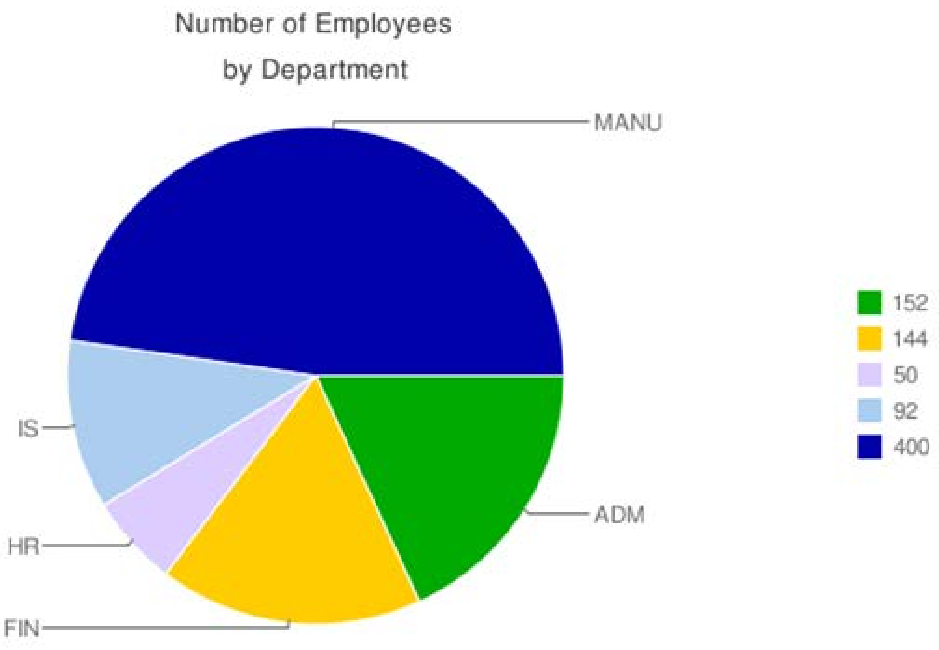
By using this extension you will be subject to the Google Terms of Service as outlined in http://code.google.com/apis/chart/tems.html

Start Point

Please – do the Dynamic Google Chart 1 tutorial before this one. This tutorial assumes that you have done it.

In this tutorial we will generate the same chart by generating the entire URL parameter string. Start with a blank screen:

Result
A Pie Google Chart showing the number of employees by department. They use data from static tables that were created by direct scripting (instead of being loaded from the server):



Steps

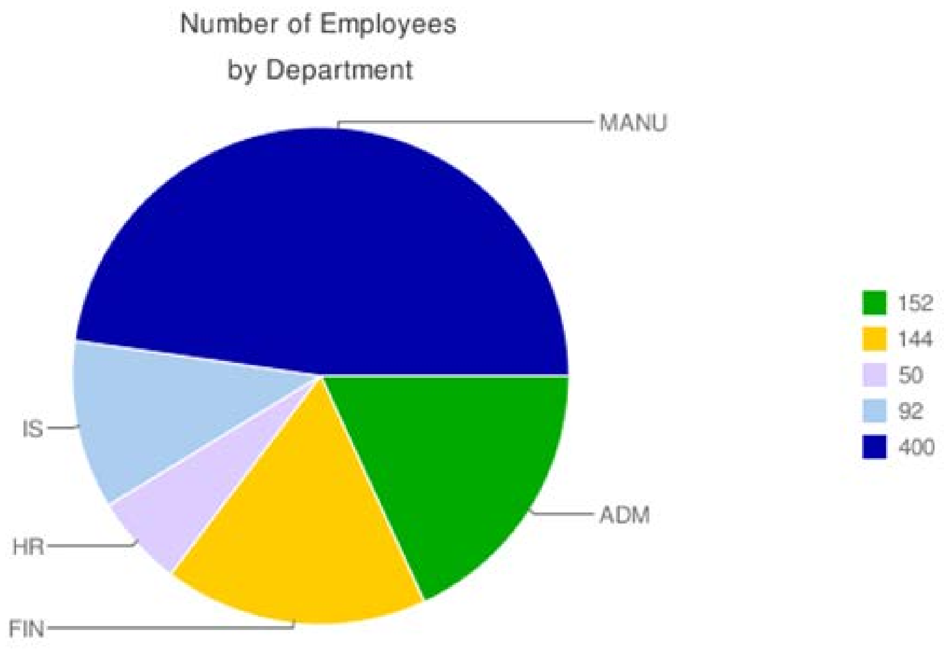
- Add a new element, set its extension type as Google Chart and size it.

Now set these extension properties:

sourceParmString: User String
tableName: Employees_in_Dept
onLoadChart:

```
/* Load the static tables file */
TABLEMANAGER.loadStaticTables(USERENV.staticTablesFile);

/* The department table returned by the selectSQL filled with departments */
var deptTable = TABLEMANAGER.getTable("XHREmployee");

/* Get the total number of table rows which is the same as the total number of employee */
var iRows = deptTable.childCount();
var oChild;
var savDept = "";
var iEmployeeInDept = 0;

/* array of colors to set to the different departments. It should contain enough colors so
that no color is repeated.  */
var arrayColors = ["00AA00", "FFCC00" , "DDCCFF", "ABCDEF", "0000AA", "84871C", "FF9900"];
var iColorIndex = 0;
var chco = "chco=";
var colorSep = "";
/* Variable to store the legends */
var chdl = "chdl=";
var chdlSep = "";
/* Variable to store the labels */
var chl = "chl=";
var chlSep = "";
/* Variable to store the data */
var chd = "chd=t:";
var chdSep = "";
var sURLParms = "chs=" + FIELD.getSize().width + "x" + FIELD.getSize().height+
"&cht=p&chtt=Number+of+Employees|by+Department";

var percent;
/* Traverse the static table, count the number of employees for each department and insert a
child row into the table nominated in the tableName property whenever a change of department
is detected. Note the order by in the selectSQL statement makes sure the records are ordered
by the department code */

for (var i = 0; i < iRows; i++)
{
 oChild = deptTable.child(i);
 if (oChild.deptCode != savDept)
 {
   if (savDept != "")
   {
     percent = (iEmployeeInDept * 100) / iRows;
     chd += chdSep + percent.toString();
     chdSep = ",";
     chl += chlSep + savDept;
     chlSep = "|";
     /* Make sure we are not passed the last color and if we have start from the first one */
     if (iColorIndex >= arrayColors.length)
     {
       iColorIndex = 0;
     }
     chco += colorSep + arrayColors[iColorIndex];
     iColorIndex++;
     /* Change the color separator so from now one it puts a comma before the next color */
     colorSep = ",";
     chdl += chdlSep + iEmployeeInDept.toString();
     chdlSep = "|";
   }
   savDept = oChild.deptCode;
   iEmployeeInDept = 1;
 }
 else
 {
    iEmployeeInDept++;
 }
}

sURLParms += "&" + chd + "&" + chl + "&" + chco + "&" + chdl;
FIELD.setProperty("userString", sURLParms);
```

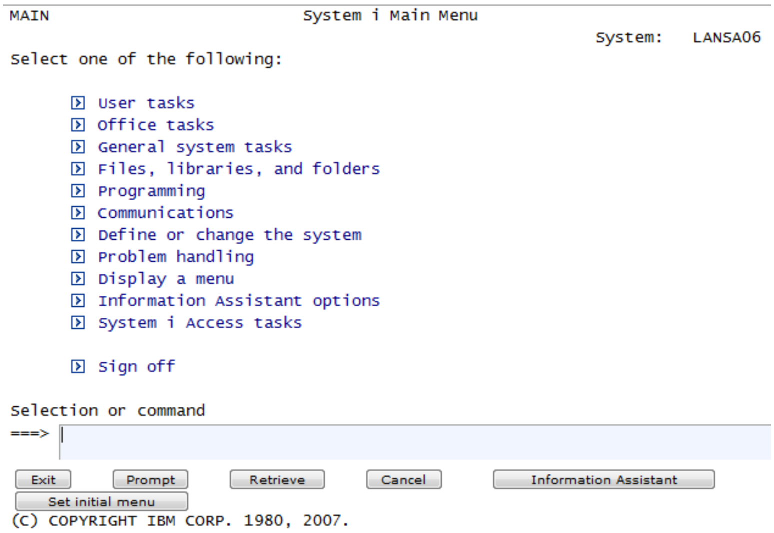
FINAL RESULT:


Number of Employees
by Department
MANU
IS
HR
FIN
ADM
🟩 152
🟨 144
🟪 50
🟦 92
🟦 400

Example – Creating Static Tables by Scripting

Start Point

The System i Main Menu, identified as screen *MAIN*, with the entry field named *CommandLine*:

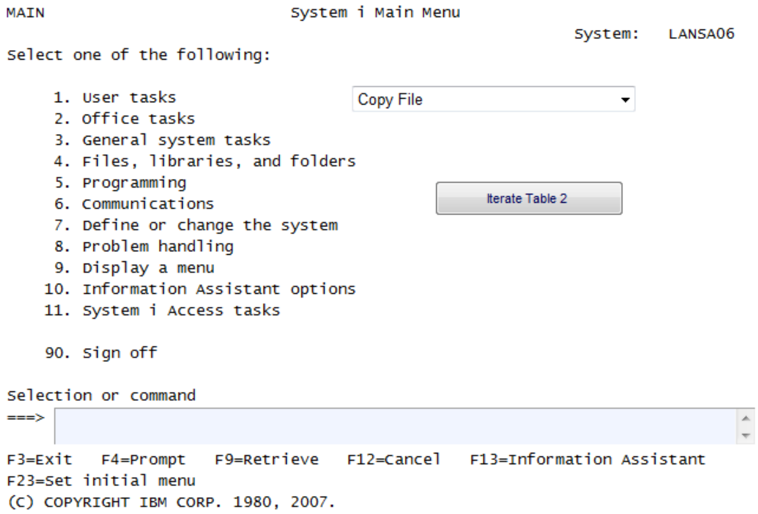


```
MAIN                         System i Main Menu
                                                        System:    LANSA06
Select one of the following:

     ⊡ User tasks
     ⊡ Office tasks
     ⊡ General system tasks
     ⊡ Files, libraries, and folders
     ⊡ Programming
     ⊡ Communications
     ⊡ Define or change the system
     ⊡ Problem handling
     ⊡ Display a menu
     ⊡ Information Assistant options
     ⊡ System i Access tasks

     ⊡ Sign off

Selection or command
===> |
```

```
(C) COPYRIGHT IBM CORP. 1980, 2007.
```

Result

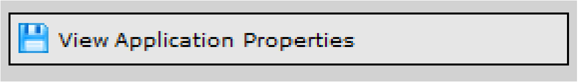
A dropdown and a push button are added to the MAIN screen. They use data from static tables that were created by direct scripting (instead of being loaded from the server):



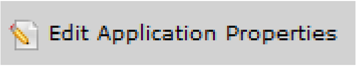
```
MAIN                         System i Main Menu
                                                        System:    LANSA06
Select one of the following:

     1. User tasks                          Copy File
     2. Office tasks
     3. General system tasks
     4. Files, libraries, and folders
     5. Programming                                 Iterate Table 2
     6. Communications
     7. Define or change the system
     8. Problem handling
     9. Display a menu
    10. Information Assistant options
    11. System i Access tasks

    90. Sign off

Selection or command
===>

F3=Exit   F4=Prompt   F9=Retrieve   F12=Cancel   F13=Information Assistant
F23=Set initial menu
(C) COPYRIGHT IBM CORP. 1980, 2007.
```

Steps

- ❖ The application properties are viewed



and then edited



- ❖ These properties were then set in the application's *Basic* properties:

onApplicationStart

```
var t1 = TABLEMANAGER.getTable("TestTable1");

t1.insertChild({text:"Copy File",value:"CPYF"});
t1.insertChild({text:"Call Program",value:"CALL"});
t1.insertChild({text:"This Job",value:"WRKJOB"});
t1.insertChild({text:"System Status",value:"WRKSYSSTS"});

var t2 = TABLEMANAGER.getTable("TestTable2");

t2.insertChild({code:"SAL",desc:"Sales"});
t2.insertChild({code:"MKT",desc:"Sales"});
t2.insertChild({code:"ADM",desc:"Administration"});
```

- The application properties were then saved.

- A *Push Button* is added to the MAIN screen as a new element.
  These properties were set:

  caption

  Iterate Table 2

  onClick

```
var t2 = TABLEMANAGER.getTable("TestTable2");
var iLimit = t2.childCount();
var sMessage = "";
for (var i = 0; i < iLimit; i++)
{
   var row = t2.child(i);
   sMessage += row.code + " - " + row.desc + "\r";
}
window.alert(sMessage);
```

- A *Drop Down* is added to the MAIN screen as a new element.
  These properties were set:

  dataSourceType

  Static Table

  tableName

  TestTable1

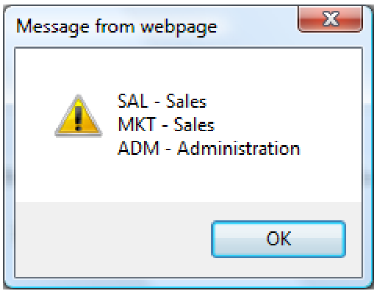
  onSelectedValueChanged

```
var field = FIELDS("CommandLine");
field.setValue(ROW.value);
```

Observations

The drop down is filled from a static table named *TestTable1*. When an entry is selected it sets the *CommandLine* field on the MAIN screen to the selected table row's *value* property/column.

The push button reads all the entries (children) in the static table named *TestTable2*. It formats a string from each rows *code* and *desc* values - presenting the result like this:



Message from webpage

SAL - Sales
MKT - Sales
ADM - Administration

The crux of this example is the application start up logic that creates the static tables *TestTable1* and *TestTable2* by executing script. The annotated logic is:

Get a reference to a table named *TestTable1*, creating it if it does not exist:

```
var t1 = TABLEMANAGER.getTable("TestTable1");
```

Create four rows (children) in the table where each child has properties/columns named *text* and *value* :

```
t1.insertChild({text:"Copy File",value:"CPYF"});
t1.insertChild({text:"Call Program",value:"CALL"});
t1.insertChild({text:"This Job",value:"WRKJOB"});
t1.insertChild({text:"System Status",value:"WRKSYSSTS"});
```

Get a reference to a table named *TestTable2*, creating it if it does not exist:

```
    var t2 = TABLEMANAGER.getTable("TestTable2");
```

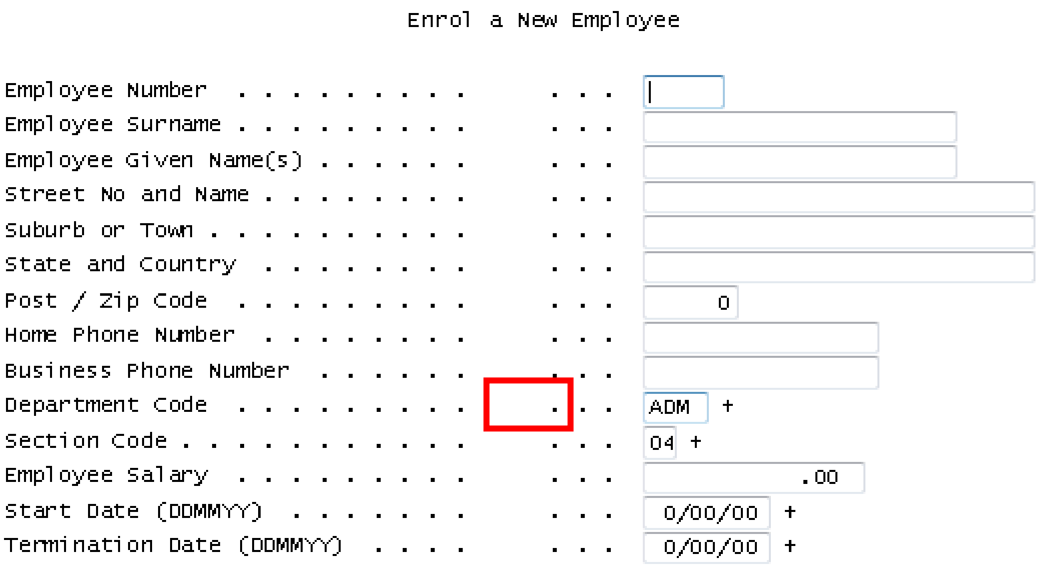
Create three rows (children) in the table - each child has properties/columns named *code* and *desc* :

```
t2.insertChild({code:"SAL",desc:"Sales"});
t2.insertChild({code:"MKT",desc:"Sales"});
t2.insertChild({code:"ADM",desc:"Administration"});
```

Example – Working with 5250 Cursor

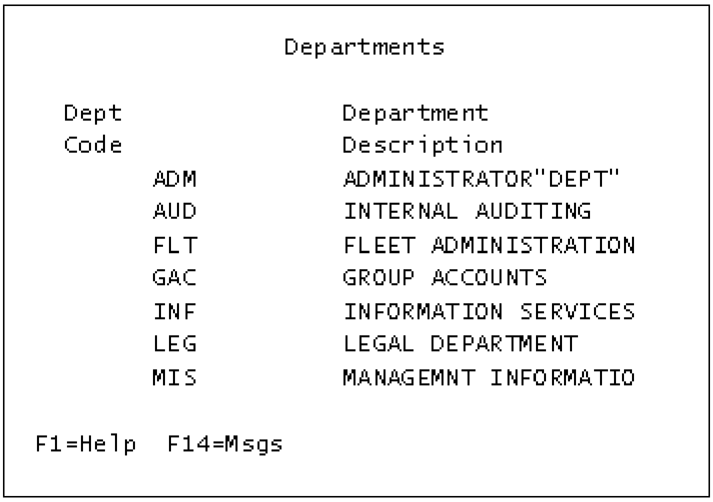
Since many 5250 applications are heavily depending on the cursor for its navigation, a mechanism to control the position the cursor in the back-end 5250 program is required.

Supposing that you have an ***enrol employee*** screen, and the department field has a prompt function that displays the list of available departments.



```
                           Enrol a New Employee

Employee Number . . . . . . . . .      . . .  |
Employee Surname . . . . . . . . .     . . .
Employee Given Name(s) . . . . . .     . . .
Street No and Name . . . . . . . .     . . .
Suburb or Town . . . . . . . . . .     . . .
State and Country . . . . . . . .      . . .
Post / Zip Code . . . . . . . . .      . . .        0
Home Phone Number . . . . . . . .      . . .
Business Phone Number . . . . . .      . . .
Department Code . . . . . . . . .      . . .  ADM  +
Section Code . . . . . . . . . . .     . . .  04 +
Employee Salary . . . . . . . . .      . . .              .00
Start Date (DDMMYY) . . . . . . .      . . .   0/00/00  +
Termination Date (DDMMYY) . . . .      . . .   0/00/00  +
```

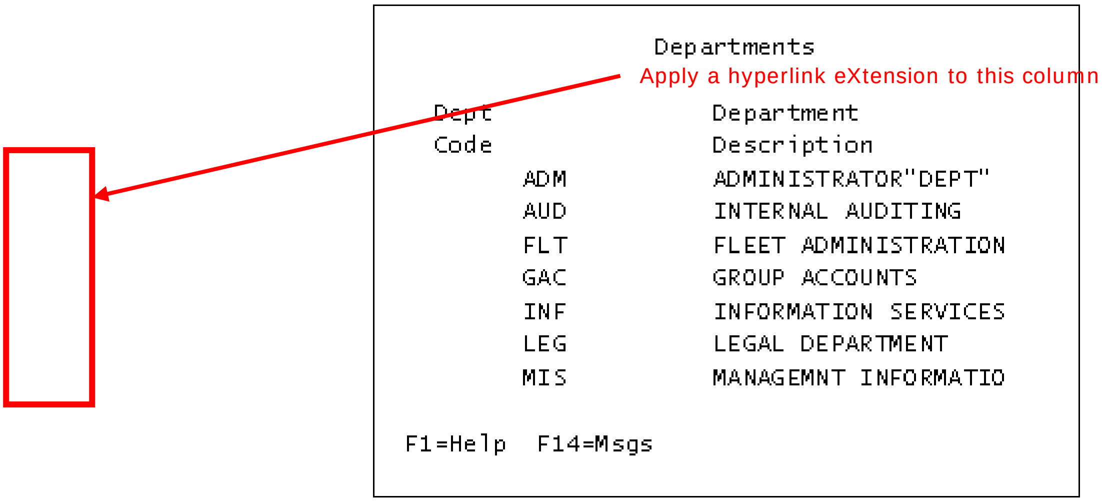
When the prompt function is executed on the **Department Code** field, a screen showing the department list will appear:



```
                  Departments

  Dept               Department
  Code               Description
        ADM          ADMINISTRATOR"DEPT"
        AUD          INTERNAL AUDITING
        FLT          FLEET ADMINISTRATION
        GAC          GROUP ACCOUNTS
        INF          INFORMATION SERVICES
        LEG          LEGAL DEPARTMENT
        MIS          MANAGEMNT INFORMATIO

F1=Help  F14=Msgs
```

To select a department in a 5250 terminal, the user will have to position the cursor on the correct line to select a department. However this will not work for a web-based application, so we need to programmatically instruct the program on the back-end to move the cursor to a specific location when the user clicks on a department in order to let the program know which department is selected.

The steps below show one way to do this.



```
                  Departments

   Dept                       Department
   Code                       Description
         ADM                  ADMINISTRATOR"DEPT"
         AUD                  INTERNAL AUDITING
         FLT                  FLEET ADMINISTRATION
         GAC                  GROUP ACCOUNTS
         INF                  INFORMATION SERVICES
         LEG                  LEGAL DEPARTMENT
         MIS                  MANAGEMNT INFORMATIO

 F1=Help  F14=Msgs
```

Apply a hyperlink eXtension to this column

Switch the **caption** property of the hyperlink to scripting mode by clicking on the pencil icon.
Type the following script into the script editor:

```
FIELD.getValue()
```

This instructs the hyperlink to display the original 5250 text as the link's text.

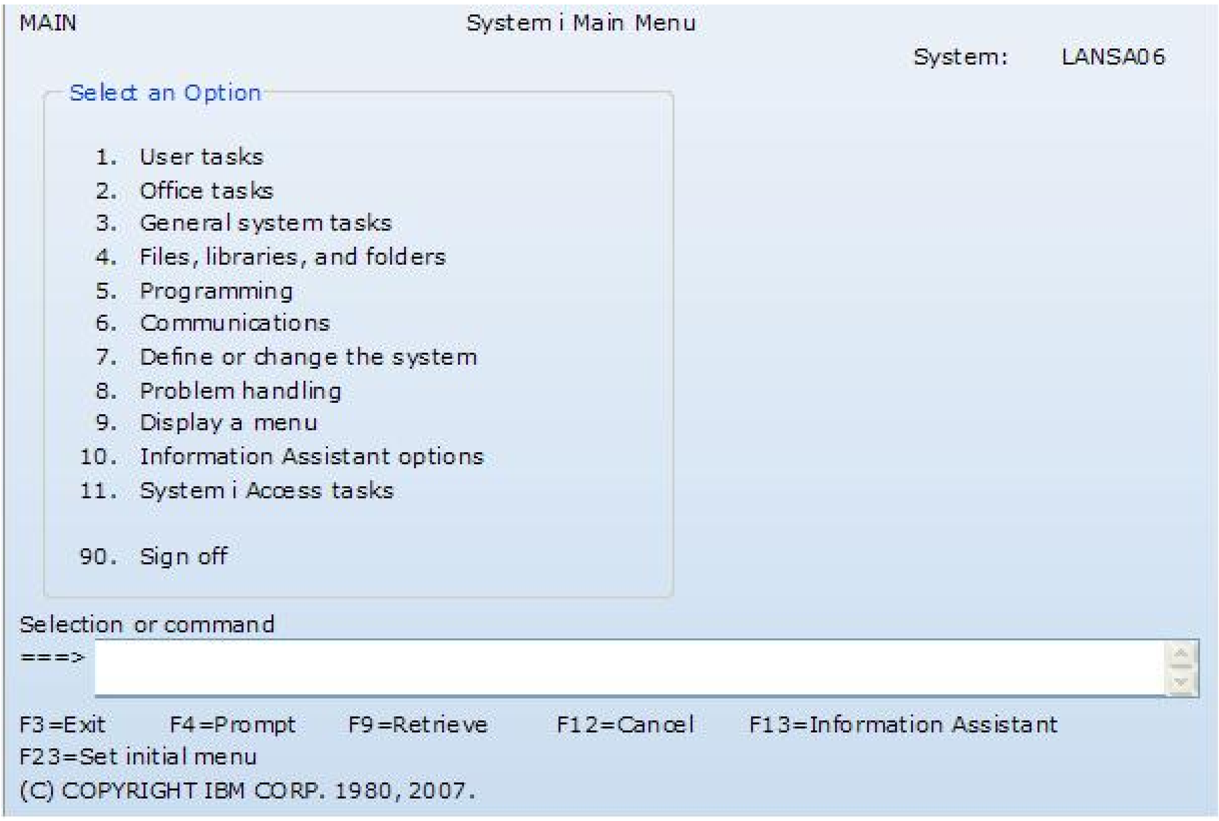
Then type in the following script in the **onClick** event of the hyperlink:

```
FIELD.set5250Cursor();
SENDKEY("Enter");
```

FIELD.set5250Cursor method will move the logical cursor to the element that is clicked. It then simulates an ENTER keystroke.

Example – Multi-lingual Text

Start Point
The System i Main Menu, identified as screen *MAIN*, with the entry field named *CommandLine*:



MAIN System i Main Menu

System: LANSA06

Select an Option

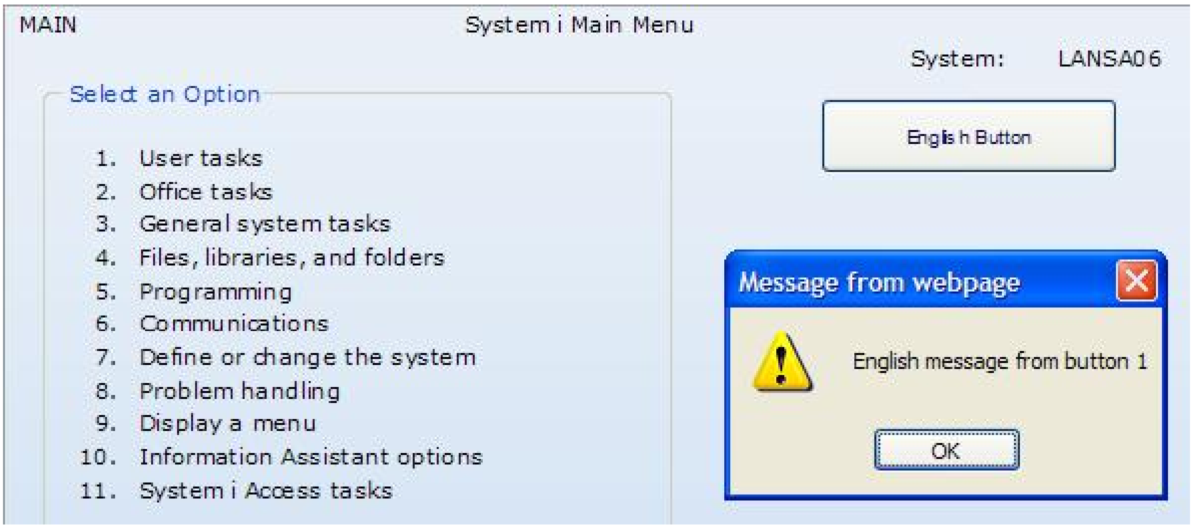
1. User tasks
2. Office tasks
3. General system tasks
4. Files, libraries, and folders
5. Programming
6. Communications
7. Define or change the system
8. Problem handling
9. Display a menu
10. Information Assistant options
11. System i Access tasks

90. Sign off

Selection or command
===>

F3=Exit F4=Prompt F9=Retrieve F12=Cancel F13=Information Assistant
F23=Set initial menu
(C) COPYRIGHT IBM CORP. 1980, 2007.

Result
A push button is added to the MAIN screen. When axes is run in the default language or with &lang=en added to the url, the button caption and the alert issued by clicking the button use english captions.



MAIN System i Main Menu

System: LANSA06

Select an Option

1. User tasks
2. Office tasks
3. General system tasks
4. Files, libraries, and folders
5. Programming
6. Communications
7. Define or change the system
8. Problem handling
9. Display a menu
10. Information Assistant options
11. System i Access tasks

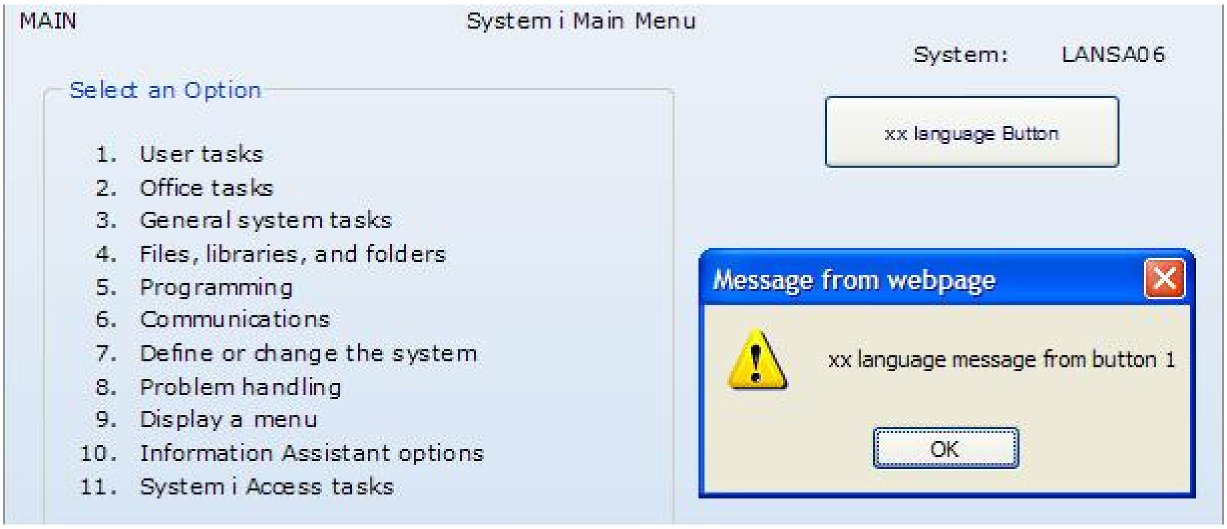
Englis h Button

Message from webpage

English message from button 1

OK

When &lang=xx is added to the url, the xx language caption is shown for the button and in the alert:



MAIN System i Main Menu

System: LANSA06

Select an Option

1. User tasks
2. Office tasks
3. General system tasks
4. Files, libraries, and folders
5. Programming
6. Communications
7. Define or change the system
8. Problem handling
9. Display a menu
10. Information Assistant options
11. System i Access tasks

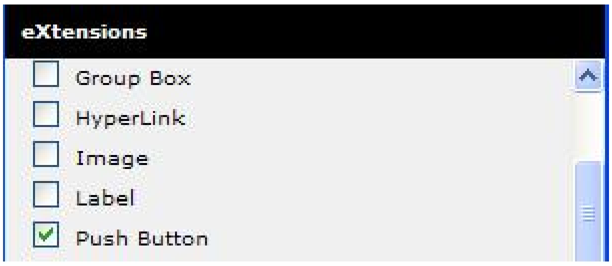
Message from webpage

xx language message from button 1

Steps

- ❖ Edit the screen properties and add a new element

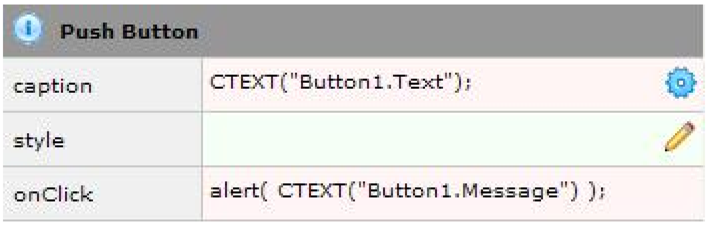
- ❖ Set the element's visualization to push button



eXtensions

□ Group Box
□ HyperLink
□ Image
□ Label
☑ Push Button

- ❖ Set its properties as shown:



Push Button

caption	CTEXT("Button1.Text");
style	
onClick	alert(CTEXT("Button1.Message"));

(note that the caption property is a script, not a value)

- ❖ Edit the file in axes/ts/lang called Custs_Text_en.txt

Add the following entries just *before* the last line:

```
"Button1.Text" : "English Button",
"Button1.Message" : "English message from button 1",
```

- ❖ Create a new file in axes/ts/lang called Custs_Text_xx.txt (Copy it from Custs_Text_en.txt)

Edit it and modify the entries you added in the previous step to be:

```
"Button1.Text" : "xx language Button",
```

"Button1.Message" : "xx language message from button 1",

- From a command line on the iSeries, check the authority to the file, using wrklnk 'axes/ts/lang/*', and then option 9 against the entry for Custs_Text_xx.txt.

  Ensure that the authority for *PUBLIC is *R (and *only* *R).

- Ensure all your changes are saved, and restart axes in run mode, and sign on.

  On the main screen, the button should show the english caption: English Button

  And when the button is clicked, the alert should show: "English message from button 1"

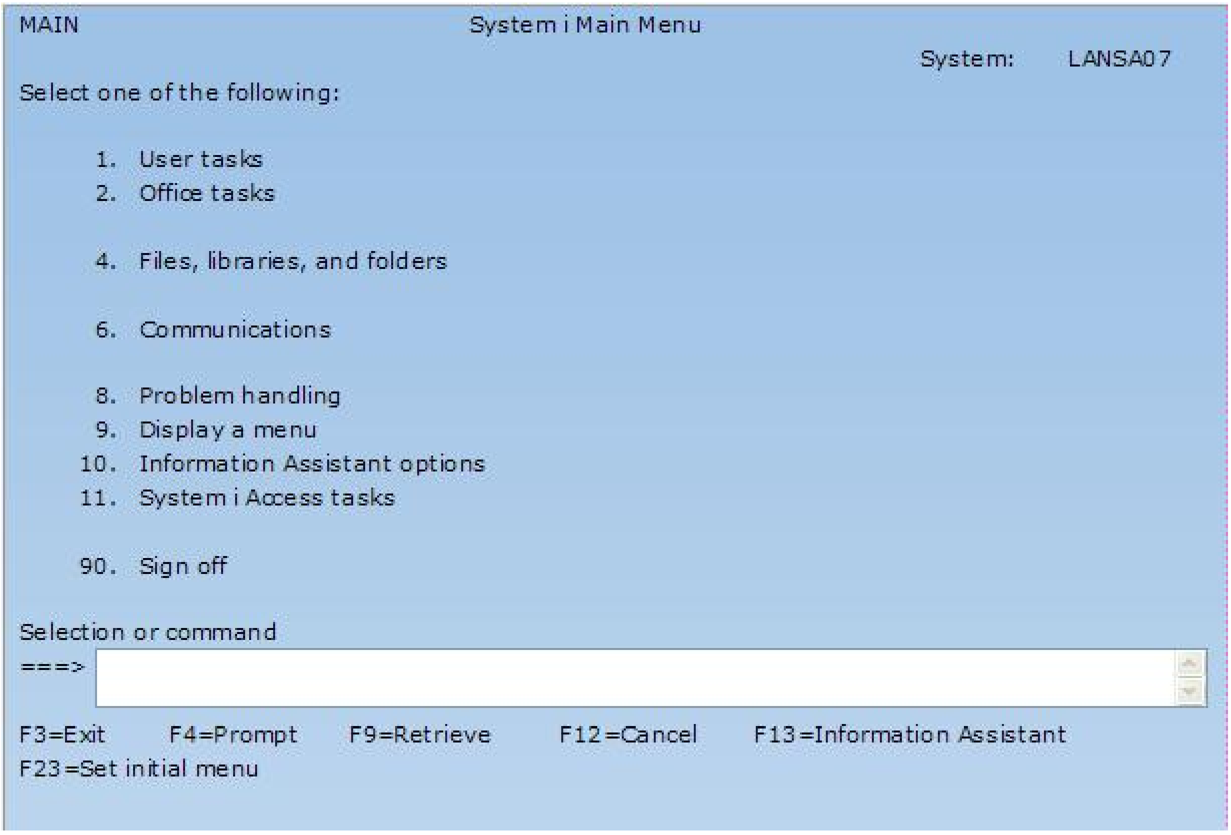
- Restart axes in run mode, but add &lang=xx to the url

  On the main screen, the button should show the xx language caption: xx language Button

  And when the button is clicked, the alert should show: "xx language message from button 1"

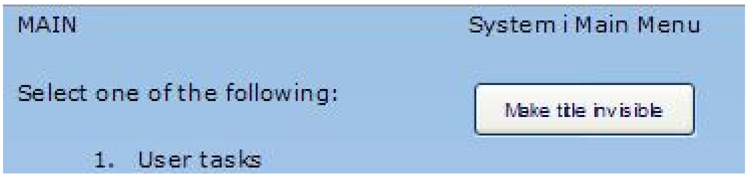
Example – Visibility control (over a screen element - dynamic)

Start Point
The System i Main Menu, identified as screen *MAIN*



```
MAIN                              System i Main Menu
                                                          System:    LANSA07
Select one of the following:

     1.  User tasks
     2.  Office tasks

     4.  Files, libraries, and folders

     6.  Communications

     8.  Problem handling
     9.  Display a menu
    10.  Information Assistant options
    11.  System i Access tasks

    90.  Sign off

Selection or command
===>

F3=Exit    F4=Prompt    F9=Retrieve    F12=Cancel    F13=Information Assistant
F23=Set initial menu
```

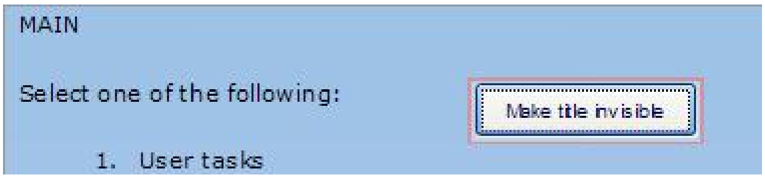
Result
A push button is added to the MAIN screen. When the button is clicked it makes the screen title element ("System i Main Menu") disappear.



MAIN System i Main Menu

Select one of the following:

1. User tasks



MAIN

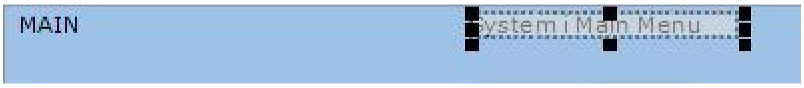
Select one of the following:

1. User tasks

Steps

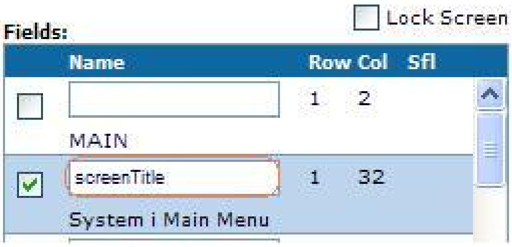
- Start axes in developer mode, and go to the first screen.

- Click on the "System i Main Menu" element on the screen. (The screen title element)



MAIN System i Main Menu

- Name the element "screenTitle". (Now that the element is named, it can be referenced by other elements on the screen) Save your changes.



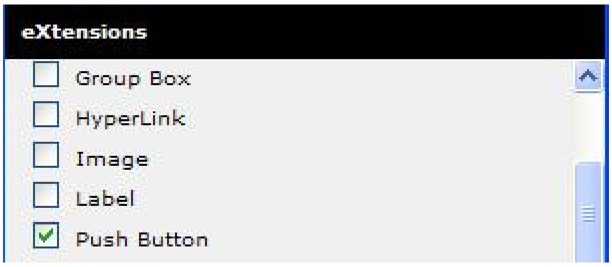
□ Lock Screen

Fields:

	Name	Row	Col	Sfl
□	MAIN	1	2	
☑	screenTitle System i Main Menu	1	32	

- Click on the Edit Screen button and Add a new element

- Set the new element's visualization to push button



eXtensions

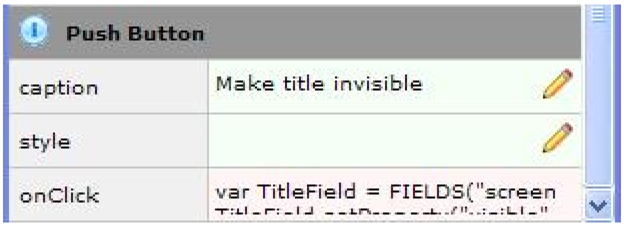
□ Group Box
□ HyperLink
□ Image
□ Label
☑ Push Button

- Set the push button's caption property to:

  Make title invisible

- Set the onClick property to:

```
var TitleField = FIELDS("screenTitle");
TitleField.setProperty("visible", false);
TitleField.refresh();
```



Push Button	
caption	Make title invisible
style	
onClick	var TitleField = FIELDS("screen

- Save your changes. You should now be able to click the button and make the screen title disappear

Observations

In the onClick code:

```
var TitleField = FIELDS("screenTitle");
```

is allowing the button element to work with the screen title element, by accessing the FIELDS collection, using the name "screenTitle" as the key.

```
TitleField.setProperty("visible", false);
```

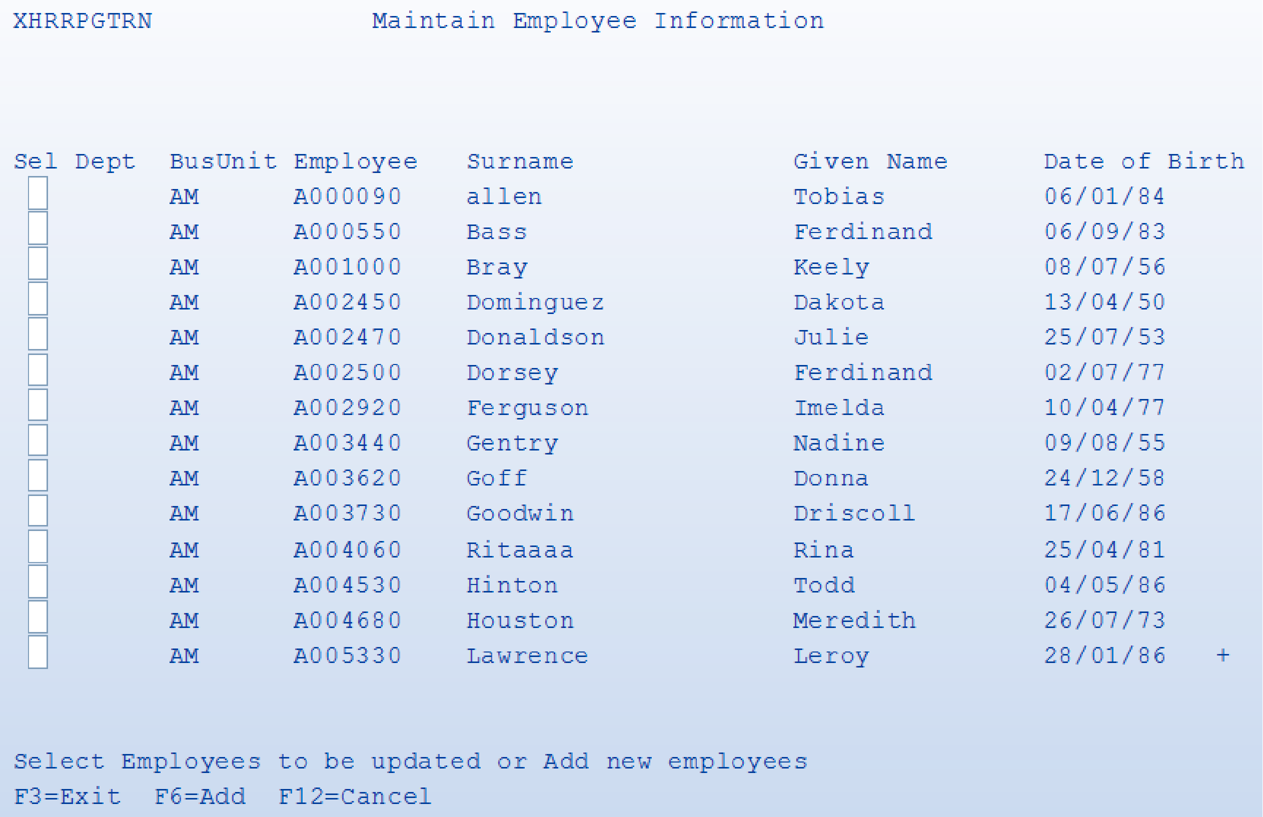
All elements have a visible property, so set the screen title's visible property to false, using this method.

```
TitleField.refresh();
```

When you change a property of an element that changes its appearance, you usually need to tell the element to re-draw itself. This is done with the .refresh() method.

Example – Using a date format not available in the Date eXtension

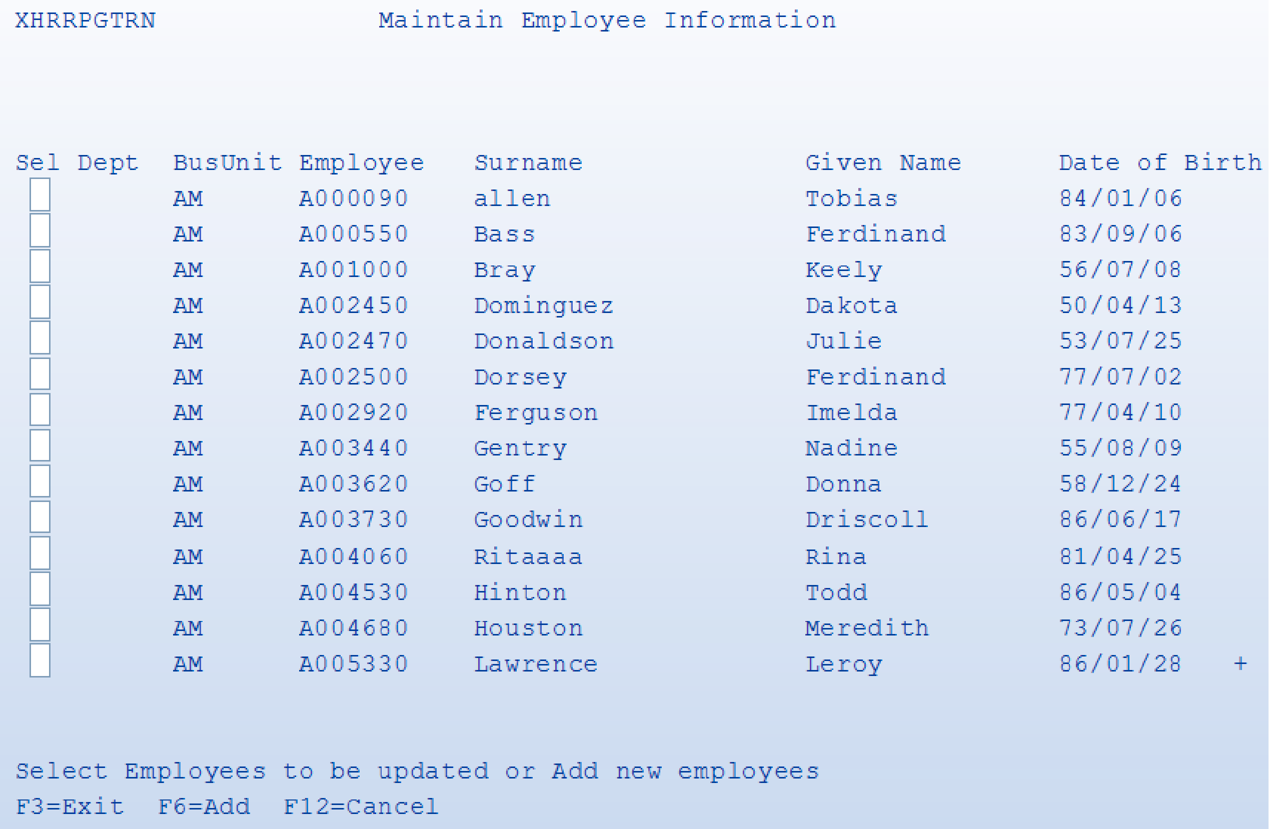
Start Point
The Maintain Employee Information subfile identified as screen *XHRRPGTRN_Select.*



```
XHRRPGTRN               Maintain Employee Information



Sel Dept  BusUnit Employee    Surname                 Given Name       Date of Birth
 □         AM      A000090     allen                   Tobias           06/01/84
 □         AM      A000550     Bass                    Ferdinand        06/09/83
 □         AM      A001000     Bray                    Keely            08/07/56
 □         AM      A002450     Dominguez               Dakota           13/04/50
 □         AM      A002470     Donaldson               Julie            25/07/53
 □         AM      A002500     Dorsey                  Ferdinand        02/07/77
 □         AM      A002920     Ferguson                Imelda           10/04/77
 □         AM      A003440     Gentry                  Nadine           09/08/55
 □         AM      A003620     Goff                    Donna            24/12/58
 □         AM      A003730     Goodwin                 Driscoll         17/06/86
 □         AM      A004060     Ritaaaa                 Rina             25/04/81
 □         AM      A004530     Hinton                  Todd             04/05/86
 □         AM      A004680     Houston                 Meredith         26/07/73
 □         AM      A005330     Lawrence                Leroy            28/01/86    +


Select Employees to be updated or Add new employees
F3=Exit  F6=Add  F12=Cancel
```

Result

Date of Birth displaying in format YY/MM/DD not available in the list of extension formats:



```
XHRRPGTRN               Maintain Employee Information



Sel Dept  BusUnit Employee    Surname                 Given Name       Date of Birth
 □         AM      A000090     allen                   Tobias           84/01/06
 □         AM      A000550     Bass                    Ferdinand        83/09/06
 □         AM      A001000     Bray                    Keely            56/07/08
 □         AM      A002450     Dominguez               Dakota           50/04/13
 □         AM      A002470     Donaldson               Julie            53/07/25
 □         AM      A002500     Dorsey                  Ferdinand        77/07/02
 □         AM      A002920     Ferguson                Imelda           77/04/10
 □         AM      A003440     Gentry                  Nadine           55/08/09
 □         AM      A003620     Goff                    Donna            58/12/24
 □         AM      A003730     Goodwin                 Driscoll         86/06/17
 □         AM      A004060     Ritaaaa                 Rina             81/04/25
 □         AM      A004530     Hinton                  Todd             86/05/04
 □         AM      A004680     Houston                 Meredith         73/07/26
 □         AM      A005330     Lawrence                Leroy            86/01/28    +


Select Employees to be updated or Add new employees
F3=Exit  F6=Add  F12=Cancel
```

Steps

- Edit the language defaults file /ts/lang/Texts_Cust_xx.txt where xx is the language code.
- Find the dateFormatDisplay setting. Change the shipped default from:

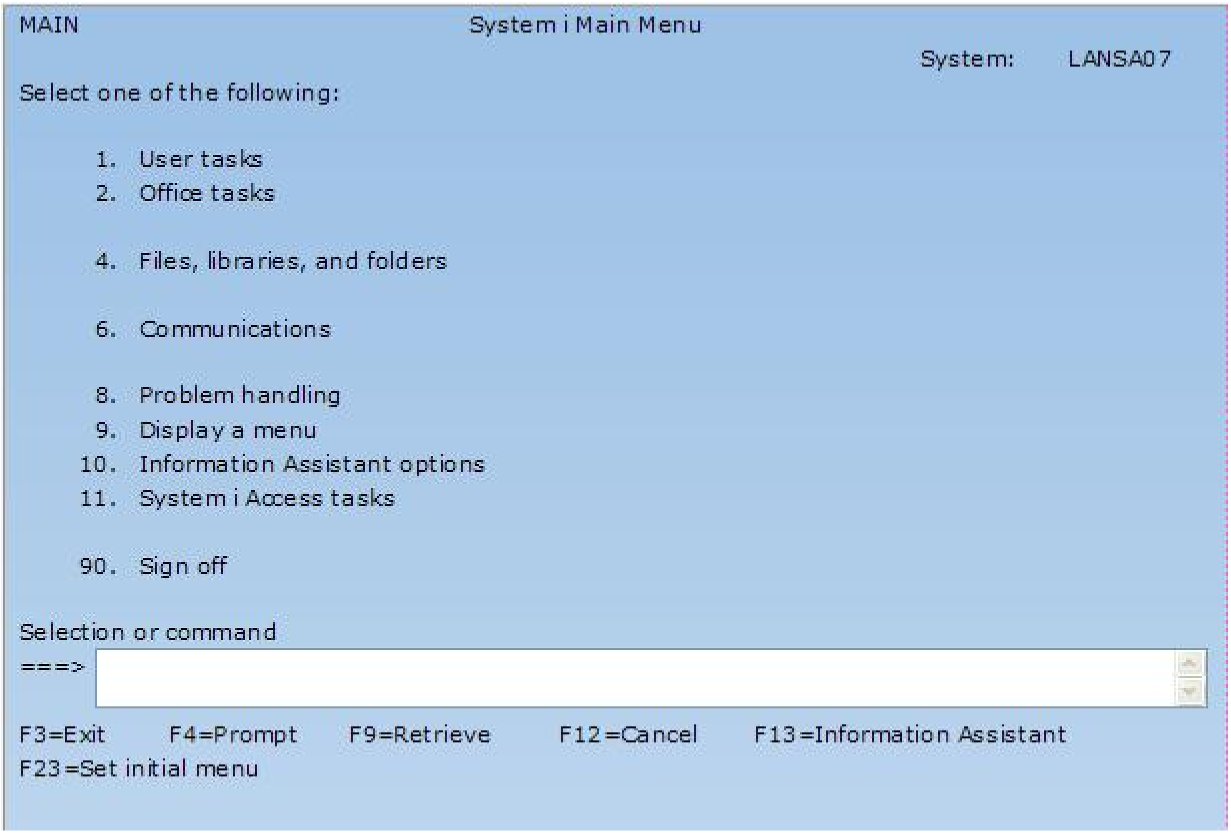
  "dateFormatDisplay" : "**d / mm / yy**",

  to

  "dateFormatDisplay" : "**y / mm / dd**",
- Save and close the file.
- Clear your browser cache to pick up the new file version.
- Start axes in developer mode go to the Maintain Employee screen, and edit the screen.
- Select one of the date fields in the subfile and change it to use the Date eXtension.
- Save the screen. YY/MM/DD is now your language default dateFormatDisplay.
- Likewise, if most of your programs use YY/MM/DD to store dates but DD/MM/YY to display them, leave the dateFormatDisplay with its shipped default value and change the dateFormatServer value.

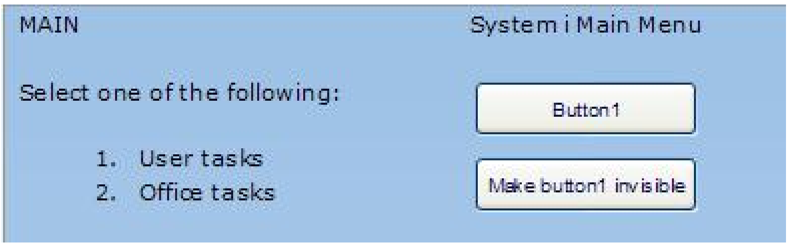
Example – Visibility control over a new element - dynamic

Start Point
The System i Main Menu, identified as screen *MAIN*



```
MAIN                        System i Main Menu
                                                      System:      LANSA07
Select one of the following:

      1.  User tasks
      2.  Office tasks

      4.  Files, libraries, and folders

      6.  Communications

      8.  Problem handling
      9.  Display a menu
     10.  Information Assistant options
     11.  System i Access tasks

     90.  Sign off

Selection or command
===>

F3=Exit      F4=Prompt      F9=Retrieve      F12=Cancel      F13=Information Assistant
F23=Set initial menu
```

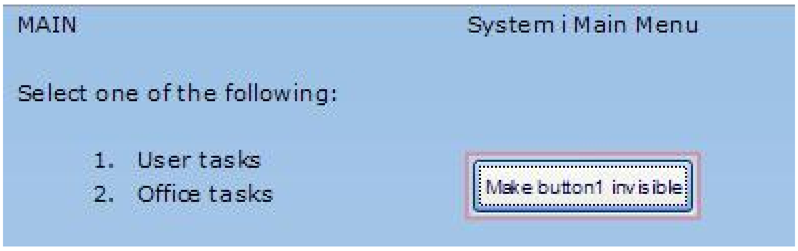
Result
Two push buttons are added to the MAIN screen. When the second button is clicked it makes the first button disappear.



MAIN System i Main Menu

Select one of the following:

1. User tasks
2. Office tasks



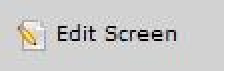
MAIN System i Main Menu

Select one of the following:

1. User tasks
2. Office tasks

Steps

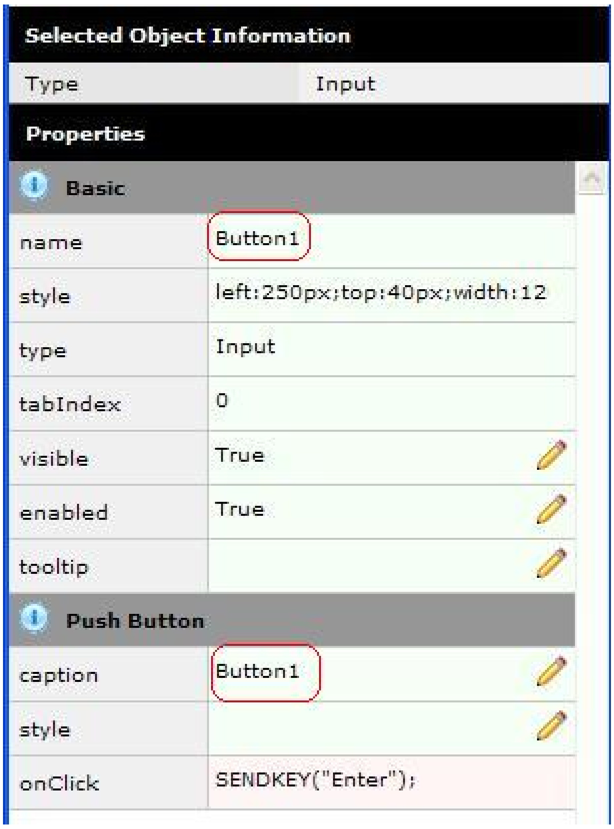
- ❖ Start axes in developer mode, and go to the first screen, and edit the screen.



- ❖ Add a new element and make it a pushbutton

Make its caption property "Button1"

Make its name property "Button1" (this allows other elements on the screen to set its properties)



Selected Object Information

Type	Input

Properties

Basic

name	Button1
style	left:250px;top:40px;width:12
type	Input
tabIndex	0
visible	True
enabled	True
tooltip	

Push Button

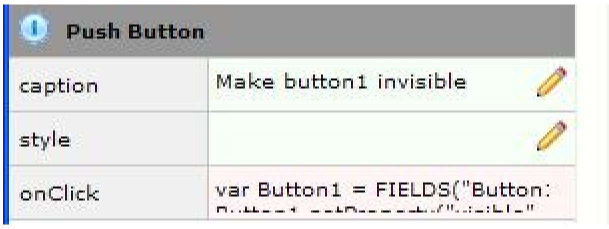
caption	Button1
style	
onClick	SENDKEY("Enter");

- ❖ Add another element, and make it a pushbutton.

Make its caption property "Make Button1 invisible"

Make its onClick property:

```
var Button1 = FIELDS("Button1");
Button1.setProperty("visible", false);
Button1.refresh();
```



Push Button

caption	Make button1 invisible
style	
onClick	var Button1 = FIELDS("Button:

- Save your changes. You should now be able to click the second button and make the first button disappear

Example – Dynamic Tables using a condition that evaluates a numeric value

Choose any screen and add:

1. A New Element type Label. Set its caption to "Salaries greater than"

2. A New Element. Leave it with the Default Visualization. Set its name to **Request_Salary**

3. A New Element type DropDown.

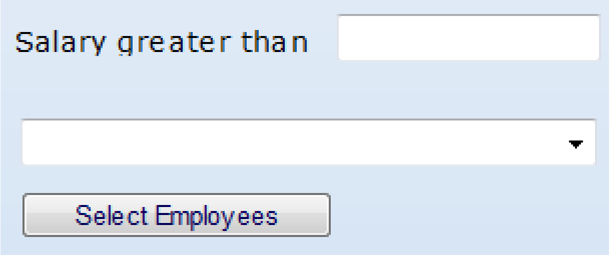
Set its dataSourceType to Dynamic.
Set its onFillDropDown property to: **ROW.lastName + " (" + ROW.salary + ")";**
Set its SQL Query Name to **EmployeeSalaries**
Set its SQL Variables property to:

```
var salaryValue = parseFloat(FIELDS("Request_Salary").getValue());

if ( (isNaN(salaryValue)) || ( typeof(salaryValue) != "number") )
{
  ENV.SQL.SQLVariableSalary = 0;
}
else
{
  ENV.SQL.SQLVariableSalary = salaryValue;
}
```

4. A New Element type Push Button:

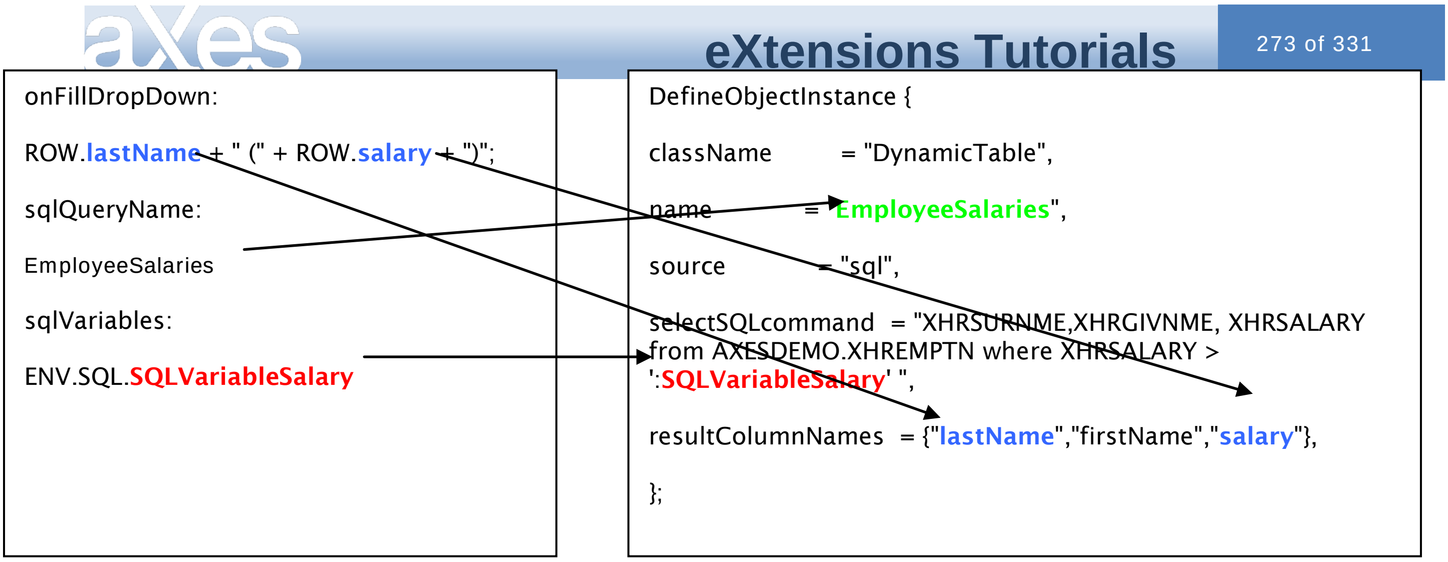
Set its caption to **Select Employees**
Set its onClick property to **FIELDS("Employees").refresh();**

The screen with the New Elements should look something like this:



5. Edit your Dynamic Tables file and add this table:

```
    DefineObjectInstance {
      className         = "DynamicTable",
      name              = "EmployeeSalaries",
      source            = "sql",
      selectSQLcommand  = "XHRSURNME,XHRGIVNME, XHRSALARY from AXESDEMO.XHREMPTN where
XHRSALARY > ':SQLVariableSalary' ",
      resultColumnNames  = { "lastName", "firstName", "salary"},
    };
```



```
onFillDropDown:

ROW.lastName + " (" + ROW.salary + ")";

sqlQueryName:

EmployeeSalaries

sqlVariables:

ENV.SQL.SQLVariableSalary
```

```
DefineObjectInstance {

className        = "DynamicTable",

name             = "EmployeeSalaries",

source           = "sql",

selectSQLcommand  = "XHRSURNME,XHRGIVNME, XHRSALARY
from AXESDEMO.XHREMPTN where XHRSALARY >
':SQLVariableSalary' ",

resultColumnNames  = {"lastName","firstName","salary"},

};
```

aXes® Documentation Library

eXtensions Tutorial 12 - Smart Phone Applications

Prerequisites for Completing this Tutorial

To follow this tutorial you must have:

1. Completed aXes tutorials 1 through 4.

2. The ability to use the IBM i operating system, 5250 DDS and perform RPG programming.

Overview and Objectives

The following material provides the foundations for developing a simple generic framework for building Smart Phone applications.

To do this it uses a generic approach and generic implementation techniques.

The objective of this tutorial is to help you understand the basics of a simple generic framework for Smart Phone applications so that you can replace, evolve, extend or customize it to create your own specific framework.

Getting Started - Check List of Objects You Need

You need to have the AXESDEMO library installed on your system.

Library AXESDEMO must contain these objects:

Name	Type / Attribute	Description	Check
QTUTORIAL	*FILE / PF	RPG, DDS and CL source code for examples	
TU4DISPLAY	*FILE / DSPF	Main Display File	
TU4DRIVER	*PGM / RPGLE	Main Driver Program	
TU4LOGON	*PGM / CLP	Direct logon program – calls TU4DRIVER	
TU4CUSTM2P	*PGM / RPGLE	Customer Inquiry – Method 2 example program	
TU4CUSTM3D	*FILE / DSPF	Customer Inquiry – Method 3 example display file	
TU4CUSTM3P	*PGM / RPGLE	Customer Inquiry – Method 3 example program	
AXMBCUST	*FILE / PF	Customer File - DDS is in QDDSSRC	
AXMBPART	*FILE / PF	Spare Parts File – DDS is in QDDSSRC	

Also complete this checklist:

Check to be performed	Check Passed
aXes development environment is installed and operational on development PC	
aXes tutorials 1 through 4 completed	
aXes version shows as 1.35.005 (or later). Check the bottom right of aXes-TS or aXes-TS2 login screen shows this (or a higher version number): Version 1.35.005 - Copyright © 2009-2010 LANSA Group. All Rights Reserved.	
Safari or Google Chrome browser installed on development PC Safari – http://www.apple.com/safari/download/ Google - http://www.google.com/chrome	

Implementing a Smart Phone Application

About This Tutorial

The following tutorial steps are designed to help you understand a variety of techniques that may be useful when implementing a smart phone application.

These tutorial steps do not produce a holistic result – a real smart phone application that you could use in a production environment – instead they produce a set of results that show you how to do various things that you would want to do in a real application. They should provide you with raw material that you can consolidate and extrapolate into a real application of your own design.

Create a new aXes project

To get started on this tutorial, log on to aXes as a developer and create a new project.
All the following tutorial steps should be performed within that project.

Setting up shared code in the USERENV.JS file (USERENV object)

The USERENV.JS file and the USERENV JavaScript object are important concepts in reusing logic and in creating single maintenance points for design and layout changes.

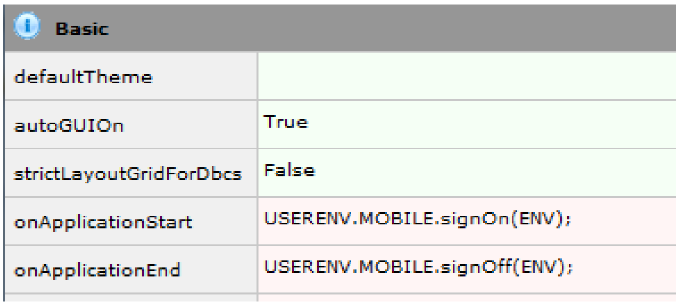
In this tutorial create a sub-object of USERENV called USERENV.MOBILE by copying and pasting this starter code inside your project's USERENV object:

```
   /* ===================================================================== */
   /* Smart Phone Manager Object - all references are USERENV.MOBILE.xxxxxx */
   /* ===================================================================== */

   MOBILE :
   {
             /* _ prefix denote private objects that should not be referenced by eXtension
scripts */

             _isiPhone         : false,
             _screenSendEnabled : false,
             _ScreenFields :
{NEXTACTION:"ifld10d",INFO_1:"ifld202",INFO_2:"ifld302",INFO_3:"ifld402",INFO_4:"ifld502",INF
O_5:"ifld602",

INFO_6:"ifld702",INFO_7:"ifld802",INFO_8:"ifld902",INFO_9:"iflda02",INFO_10:"ifldb02"},

             /* Exposed properties and methods that may be referenced by eXtension scripts */

             currentCUSTOMER   : null,

             deviceWidth       : function(env) { env.returnValue = 320; },
             deviceHeight      : function(env) { env.returnValue = 430;  },
             deviceZoom        : function(env) { if (USERENV.MOBILE._isiPhone)
env.returnValue = -1; else env.returnValue = 100; },
             lockZoom          : function(env) { env.returnValue = true; },

             signOn : function(env)
             {
                USERENV.MOBILE._isiPhone = (navigator.userAgent.match(/iPhone|iPod/) !=
null);
                env.SHOWAXESMENUBAR(false);
                env.SHOWAXESSTATUSBAR(false);
             },

             signOff : function(env)
             {
                env.SHOWAXESMENUBAR(true);
                env.SHOWAXESSTATUSBAR(true);
                USERENV.MOBILE._screenSendEnabled = false;
             },

             onArrive : function(env)
             {
                USERENV.MOBILE.screenSendEnabled = false;
             },

             onLeave : function(env)
             {
                env.returnValue = USERENV.MOBILE._screenSendEnabled;
             },

             setINFO : function(env,index,value)
             {
                var id = USERENV.MOBILE._ScreenFields["INFO_" + index.toString()];
                if (id == null) { alert("USERENV.MOBILE.setINFO encountered an invalid index
value."); return; };
                var element = AXES.currentForm.getElementById(id);
                if (element == null) {alert("USERENV.MOBILE.setINFO cannot find specified
INFO_n field element on screen."); return; };
                element.setValue(value.toString());
             },

             getINFO : function(env,index,value)
             {
                var id = USERENV.MOBILE._ScreenFields["INFO_" + index.toString()];
                if (id == null) {alert("USERENV.MOBILE.getINFO encountered an invalid index
value."); return(""); };
                var element = AXES.currentForm.getElementById(id);
                if (element == null) {alert("USERENV.MOBILE.getINFO cannot find specified
INFO_n field element on screen."); return(""); };
                return(element.getValue());
             },

             gotoScreen : function(env,nextscreen)
             {
                if (nextscreen == null) nextscreen = "HOME";
                var nextaction =
AXES.currentForm.getElementById(USERENV.MOBILE._ScreenFields.NEXTACTION);
                if (nextaction == null) {alert("USERENV.MOBILE.gotoScreen cannot find
NEXTACTION field."); return };
                nextaction.setValue(nextscreen);
               for (var i = 2; i < arguments.length; i++) { USERENV.MOBILE.setINFO(env,(i-
1),arguments[i]); }
                USERENV.MOBILE._screenSendEnabled = true;
                env.SENDKEY("Enter");
             }
   }, /* <--- Note the comma */

   /* ================================= */
   /* End of Smart Phone Manager Object */
   /* ================================= */
```

Using centralized signOn() and signOff() logic

Your USERENV.MOBILE object contains functions signOn() and signOff().

Plug them into your project by editing your application properties so that they are called at the appropriate times - like this:



Basic	
defaultTheme	
autoGUIOn	True
strictLayoutGridForDbcs	False
onApplicationStart	USERENV.MOBILE.signOn(ENV);
onApplicationEnd	USERENV.MOBILE.signOff(ENV);

The calls are

USERENV.MOBILE.signOn(ENV);

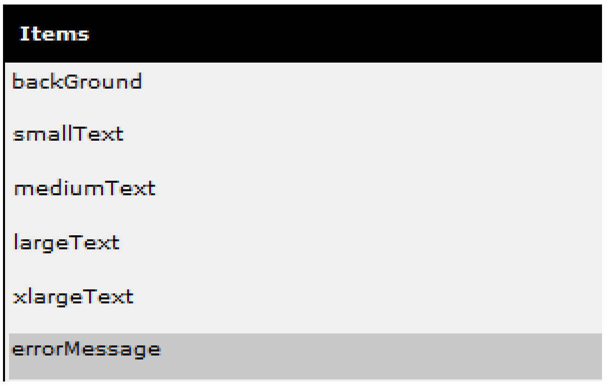
and

USERENV.MOBILE.signOff(ENV);

Save your project changes.

Set up Some Styles

Next you need to set up some base styles as part of your application.
Set them into your project by editing your application properties.
In this tutorial we are going to use these styles:



Items

backGround

smallText

mediumText

largeText

xlargeText

errorMessage

Which have these style property values:

Style Item Name	Property	Value to use (double check your values)
background	Background	#27282d
smallText	Font-Family Font-Size Color	Verdana 10pt White
mediumText	Font-Family Font-Size Color	Verdana 10pt White

largeText	Font-Family Font-Size Color	Verdana 10pt White
xlargeText	Font-Family Font-Size Color	Verdana 10pt White
errorMessage	Font-Family Font-Size Color	Verdana 10pt Red

Define these base styles into your project and save your changes.

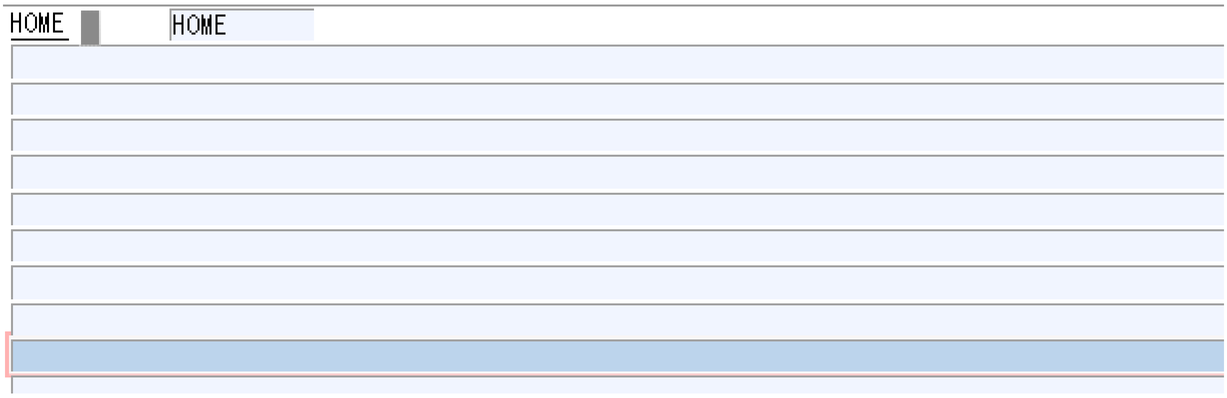
Setting up a HOME Screen

Log on as an aXes developer.

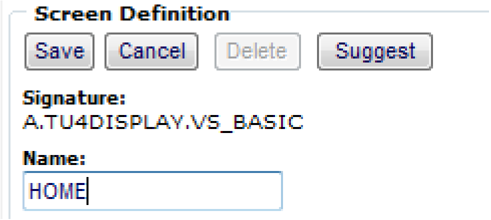
Add library AXESDEMO to your library list.

Call program TU4DRIVER – the tutorial 4 driver program written in free format ILE RPG. The source code for this program is in source file QTUTORIAL in library AXESDEMO. It has no parameters.

The resulting 5250 display will look like this - which may seem a bit strange at first.



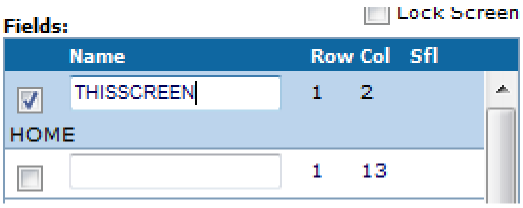
Identify this screen with the name HOME - the Suggest button will do this automatically or you can type the name in:



Signature:
A.TU4DISPLAY.VS_BASIC

Name:
HOME

Now click on the first field on the screen (an output field at line 1 position 2 containing the word "HOME") to select it – name the screen field THISSCREEN.



Fields: □ Lock Screen

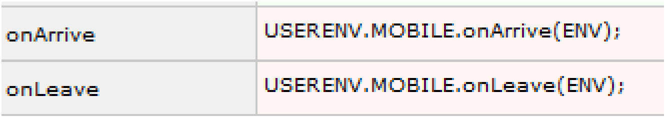
	Name	Row	Col	Sfl
☑	THISSCREEN HOME	1	2	
□		1	13	

Make sure to check the identification box for this field so that it is used as part of the screen signature to uniquely identify it.

Save your screen identification changes.

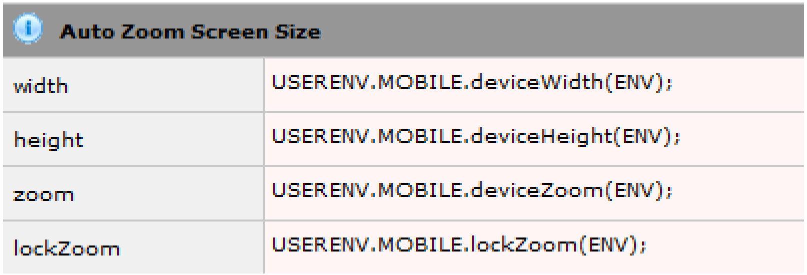
Now start customizing this screen.

Hook up the USERENV.MOBILE.onArrive() and USERENV.MOBILE.onLeave() functions like this:



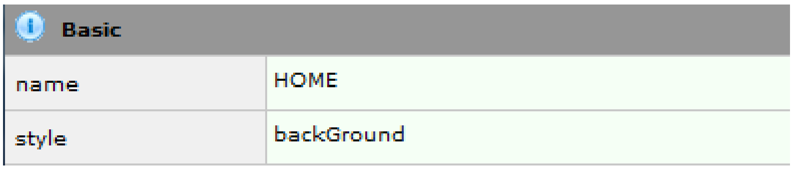
onArrive	USERENV.MOBILE.onArrive(ENV);
onLeave	USERENV.MOBILE.onLeave(ENV);

Then add an Auto Zoom Screen Size extension to the screen and hook up all the USERENV.MOBILE sizing functions - like this:



Auto Zoom Screen Size	
width	USERENV.MOBILE.deviceWidth(ENV);
height	USERENV.MOBILE.deviceHeight(ENV);
zoom	USERENV.MOBILE.deviceZoom(ENV);
lockZoom	USERENV.MOBILE.lockZoom(ENV);

Now apply the background style created earlier to the whole screen:

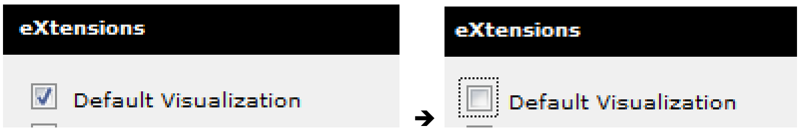


Basic	
name	HOME
style	backGround

Save your changes.

Next you need to hide every 5250 field on the screen.

You do this by selecting each field and un-checking its default visualization extension - like this:



When you save your changes you should see a completely blank screen.

This is a blank "virtual screen" canvas for you to start painting on:

Make sure that your screen is this dark colour.

In the following steps we are going to insert white text onto it.

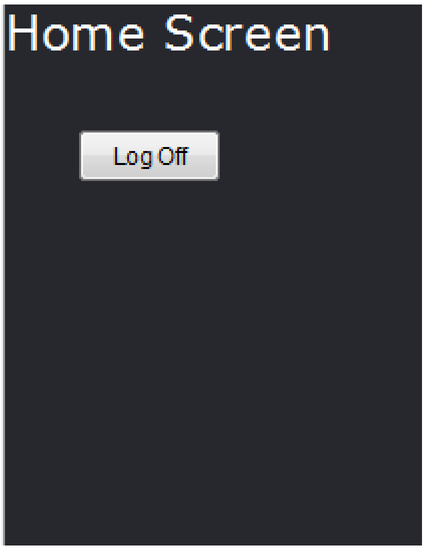
If you start inserting white text onto a white background things can become very confusing!

Now add a label eXtension to your screen captioned *Home Screen.*

Use the *xlargeText* base style for the label to make it use the large **white** font.

Then add a push button eXtension captioned *Log Off* to your screen.

The screen should now look something like this:



Change the ***onClick*** property of the push button to execute this script:

```
USERENV.MOBILE.gotoScreen(ENV,"EXIT");
```

The gotoScreen() function puts the string "EXIT" into the 5250 screen and sends the enter key. This tells the driver program TU4DRIVER to end.

Save your changes and then click the Log Off button – you should be returned to where you called the program TU4DRIVER from.

Note: In a real application a program like TU4DRIVER would be called automatically when the user signs on - so when TU4DRIVER ends the user would be logged off from the system. More on this later.

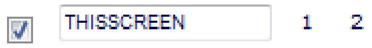
You should now be able to call up the TU4DRIVER program, which will display your HOME page, and then end it again by clicking the Log Off button. Please ensure you can do this easily by repeating the process several times before proceeding with this tutorial.

You have now created a Home Page for your application.

It is pretty sparse at the moment – however the following tutorial steps will show you how to add functionality to it.

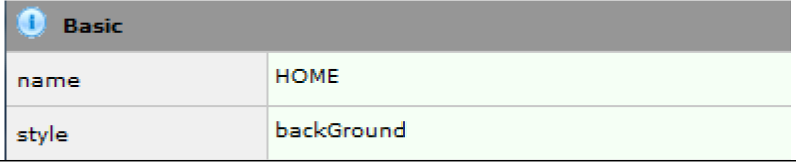
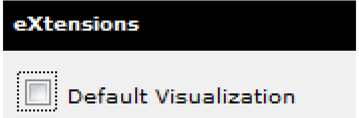
New Screen Checklist

For every new screen you define in this tutorial you should complete this check list:

Step	Action	Checked
1	The screen is named HOME, MESSAGE or VS_xxxxxxxx according to the name displayed in output field at line 1 position 2 on screen.	
2	Output field at line 1 position 2 named THISSCREEN has been checked as an identification field  ☑ THISSCREEN 1 2	
3	The THISSCREEN field at line 1 position 2 displays the same screen name as you input in step 1 (i.e.: you have given the screen the correct name).	
4	Screen identification details have been saved	
5	Screen customization started	

6	Screen onArrive and onLeave events execute correct USERENV.MOBILE functions:
	onArrive USERENV.MOBILE.onArrive(ENV);
	onLeave USERENV.MOBILE.onLeave(ENV);	

7	Auto Zoom Screen Size has been set up like this:
	(the properties need to be set to evaluate script mode)
	width USERENV.MOBILE.deviceWidth(ENV);
	height USERENV.MOBILE.deviceHeight(ENV);
	zoom USERENV.MOBILE.deviceZoom(ENV);
	lockZoom USERENV.MOBILE.lockZoom(ENV);	

8	The base style *backGround* has been applied to whole screen:  Basic name HOME style backGround	
9	All unwanted 5250 fields on the screen have been hidden from view by un-checking their default visualization property:  eXtensions ☐ Default Visualization	
10	Screen customization changes saved	

Setting Up a Messages Screen

It's useful to have a generic messages screen in smart phone applications.

---

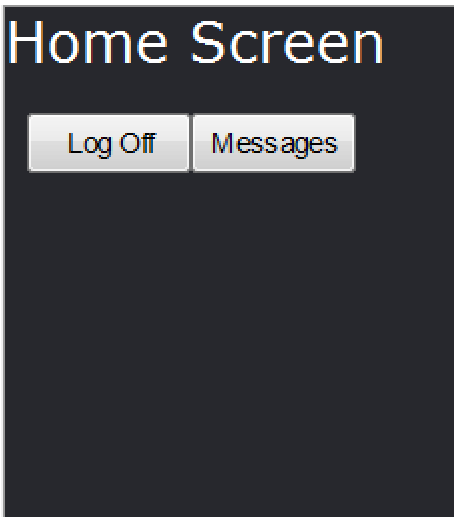
To set up a messages screen do the following:

Call program TU4DRIVER to display your Home Page.

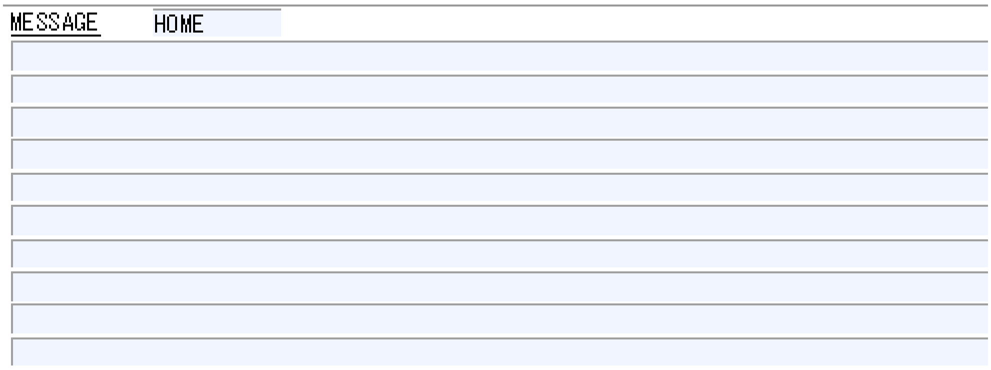
Add a new push button to your Home Page captioned Messages that calls a Message screen when clicked.

```
USERENV.MOBILE.gotoScreen(ENV,"MESSAGE");
```

Save your changes. Your home screen should now look something like this:



Click the Message button. The result should look like this:



This is a brand new blank canvas virtual screen (more about them later).

This screen should be named **MESSAGE**.

Complete the preceding New Screen Checklist to set this screen up for proper smart phone operations – you should end up with a completely blank screen.

Now add a new label eXtension ("Messages") and two new push button eXtensions ("Home" and "Log Off") to your messages screen.

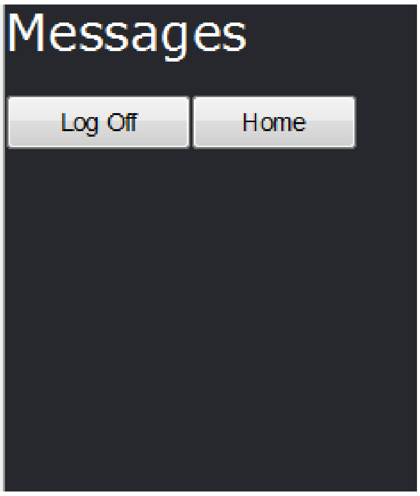
The Home button should do this when clicked or touched:

```
USERENV.MOBILE.gotoScreen(ENV,"HOME");
```

The Log Off button should do this when clicked or touched:

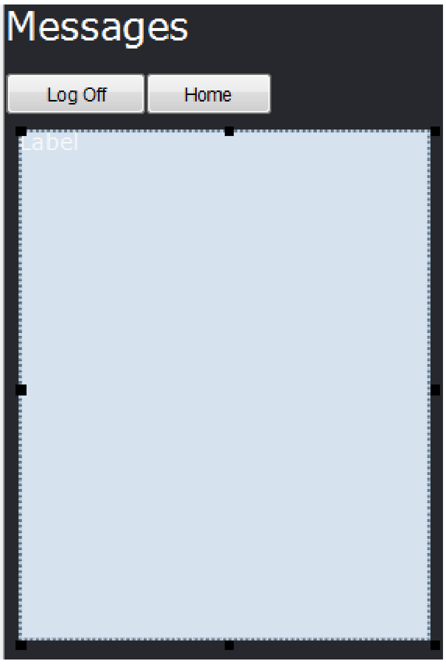
```
USERENV.MOBILE.gotoScreen(ENV,"EXIT");
```

Your MESSAGE screen should now look like this:



Save your changes.

Now add a new label eXtension to the middle of your MESSAGE screen - something like this:



Use base style mediumText so that text is largish and easy to read.

Now change the ***text*** property of the label so that the content it is dynamically created.

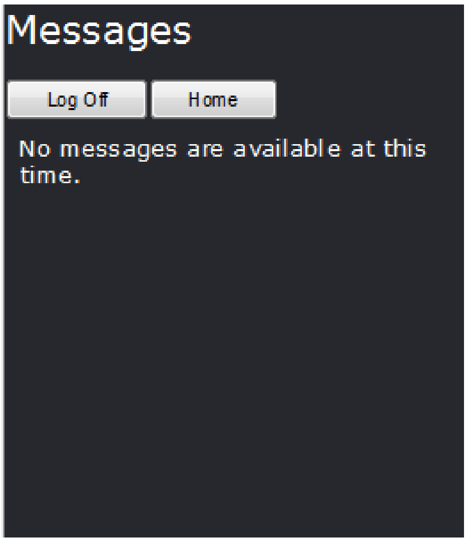
The script you need to use is something like this:

```
var text = "";
for (var i = 1; i <= 10; i++) { text += USERENV.MOBILE.getINFO(ENV,i); }
if (text == "") text = "No messages are available at this time.";
ENV.returnValue = text;
```

What this script does is retrieve the contents of the hidden 5250 fields and assemble them into one long text string. The resulting string is displayed in the label eXtension.

Save your changes.

Your MESSAGE screen should now look like this:



No messages are available at this time.

You should now be able to start up the smart phone application by calling TU4DRIVER to display your HOME screen.
From the HOME screen you can display the MESSAGE screen, and from there you can go back to the HOME screen or EXIT (log off).

Please check you can do all these things before proceeding.

Presenting Information on a Smart Phone

The following tutorial steps are going to implement a *Customer Details Inquiry* using three different methods.

- Method 1: The inquiry is controlled by the client side scripting - using SQL to extract the customer details.
- Method 2: The inquiry is controlled by the client side scripting – but uses a service or subroutine style RPG program to extract the required customer details.
- Method 3: The inquiry is initiated by client side scripting – but the presentation is performed and ultimately managed and validated by a classic 5250 RPG program.

Obviously doing the same thing three different ways is not the objective of this tutorial.

The objective is to help you to understand the three main choices you have for common forms of information presentation from a smart phone.

By completing these tutorial steps you should be able to compare and contrast the advantages and disadvantages of each method – allowing you to make the best choice for real scenarios that you encounter.

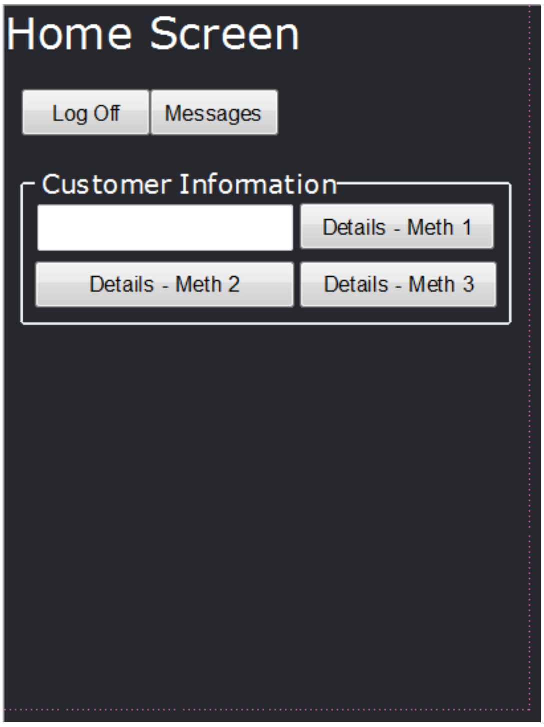
Customer Inquiry

Start aXes as a developer and call program TU4DRIVER to display your HOME screen.

Add to your HOME screen:

- A group box eXtension,
- An input field eXtension (= default visualization)
- Three push buttons eXtensions

Position, style and label them like this:

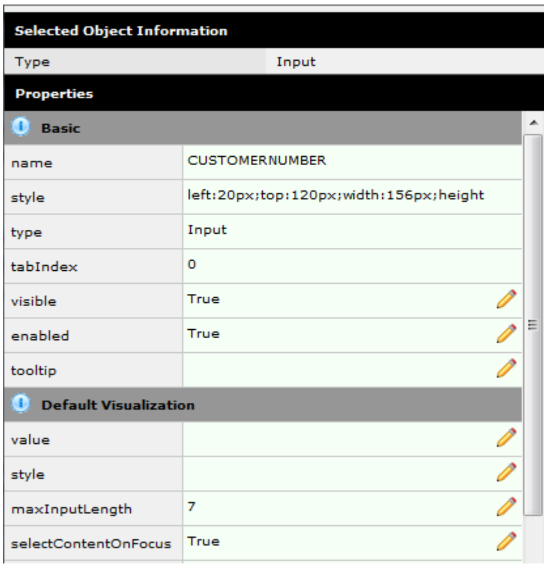


You need to set the style of the group box to use a white font.

Make sure that the input eXtension is:

- named CUSTOMERNUMBER
- has its maximumInputLength property set to 7

Like this:



Selected Object Information

Type	Input

Properties

Basic

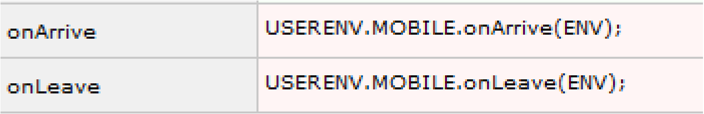
name	CUSTOMERNUMBER
style	left:20px;top:120px;width:156px;height
type	Input
tabIndex	0
visible	True
enabled	True
tooltip	

Default Visualization

value	
style	
maxInputLength	7
selectContentOnFocus	True

Save your customization changes.

Optional Technical Notes:
Try clicking your three new Details push buttons - notice how nothing happens?
This is because your home screen has on onLeave function like this:



onArrive	USERENV.MOBILE.onArrive(ENV);
onLeave	USERENV.MOBILE.onLeave(ENV);

If you look at the onArrive and onLeave functions in USERENV.MOBILE they do this:

```
onArrive : function(env)
{
  USERENV.MOBILE.screenSendEnabled = false;
},

onLeave : function(env)
{
  env.returnValue = USERENV.MOBILE._screenSendEnabled;
},
```

Every time a screen arrives the property USERENV.MOBILE.screenSendEnabled is set to false.

Every time the screen tries to leave (e.g.: when you click one of the new Details buttons) the current value of USERENV.MOBILE._screenSendEnabled is returned back to the aXes driver to indicate whether to proceed or not.

So when you click one of the new Details buttons the value being returned is false – so the request to send details to the server (i.e.: leave this screen) is ignored.

The important matching code is in:

```
gotoScreen : function(env,nextscreen)
{
   <unrelated logic has been omitted>
   USERENV.MOBILE._screenSendEnabled = true;
   env.SENDKEY("Enter");
 }
```

When you invoke USERENV.MOBILE.gotoScreen() it sets the blocking property screenSendEnabled to true – which means that the subsequent SENDKEY("Enter") will be sent back to the server.

In other words - the *only* way to submit a request to the server is by using USERENV.MOBILE.gotoScreen() – any other scripted or user imitated use of the Enter key, function keys (which don't exist on smart phones anyway) are all ignored.

Method 1 - Using Client Logic Only

One of the ways you can define screens for presenting information is to use a *virtual screen*.

A virtual screen is one that does not need a specialized 5250 program on the server to create its visual content or behaviour. The visual content typically comes from execution of scripts and server side SQL requests.

The screen content is assembled and controlled entirely by the client logic.

First, define a dynamic SQL request in your project's Dynamic Tables file. This defines an SQL command to find a specified customer and creates an aXes table named AXMBCUST-INQUIRE:

```
    DefineObjectInstance {
      className          = "DynamicTable",
      name               = "AXMBCUST-INQUIRE",
      source             = "sql",
      selectSQLcommand   = "CUSTNUMBR,CUSTNAME,CUSTADDR,CUSTCITY,CUSTZIP,CUSTPHONE,CUSTEMAIL
from AXESDEMO.AXMBCUST where CUSTNUMBR = :SQLVariable_CUSTOMERNUMBER",
      resultColumnNames  = {
"CUSTNUMBR","CUSTNAME","CUSTADDR","CUSTCITY","CUSTZIP","CUSTPHONE","CUSTEMAIL" },
    };
```

Save your changes and follow any restart instructions.

Now add this code to your *Details – Meth 1* button's click handling:

```
/* Default the things to do next */
var NEXTACTION = "MESSAGE";
var INFO_1 = "";

/* Get the customer number from the current screen as number */

var iCustNo = parseInt(FIELDS("CUSTOMERNUMBER").getValue(),10);

/* If the customer number is not valid goto message screen */

if ((isNaN(iCustNo))||(iCustNo < 1)||(iCustNo > 9999999))
{
   INFO_1 = "Customer number " + iCustNo.toString() + " is not a valid customer number.";
}
else
{

   /* Ask the manager to load the dynamic table, passing the customer number as an SQL
variable */
   TABLEMANAGER.loadDynamicTable("AXMBCUST-
INQUIRE",USERENV.dynamicTablesFile,{SQLVariable_CUSTOMERNUMBER:iCustNo.toString()});

   /* Get the first row from the aXes table produced by executing the SQL command */
   USERENV.MOBILE.currentCUSTOMER = TABLEMANAGER.getTable("AXMBCUST-INQUIRE").child(0);

   /* If not found then no customer with the number exists so go to message screen */
   if (USERENV.MOBILE.currentCUSTOMER == null)
     INFO_1 = "No customer with customer number " + iCustNo.toString() + " can be found on
the server.";

   else
      NEXTACTION = "VS_CUSTM1";
}

/* Proceed on to the next action/screen */

USERENV.MOBILE.gotoScreen(ENV,NEXTACTION,INFO_1);
```

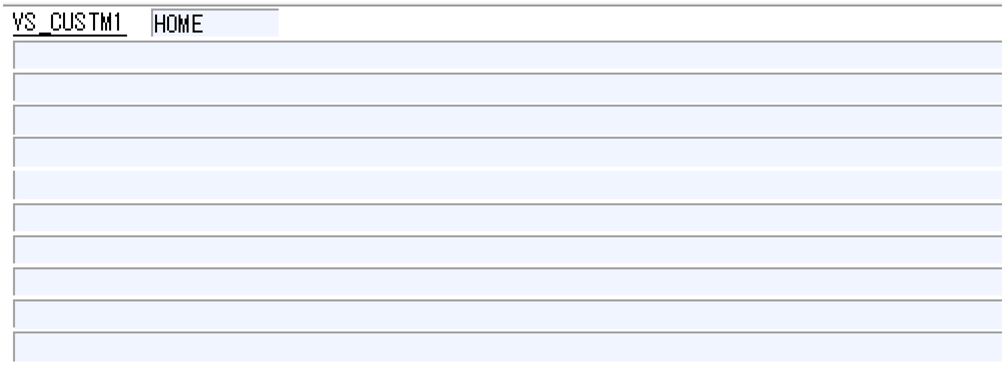
This script:

- Retrieves and validates the CUSTOMERNUMBER entered by the user.
- Executes the SQL command defined in the dynamic SQL table (AXMBCUST-INQUIRE).
- If a row (record) is found it goes to the screen named **VS_CUSTM1**.
- If a row (record) cannot be found it goes to the screen named **MESSAGE**.

Try out your *Details – Meth 1* by entering numbers 1111111 and 999999. They should cause your MESSAGE screen to be displayed.

Now try out a valid number like 1 or 6.

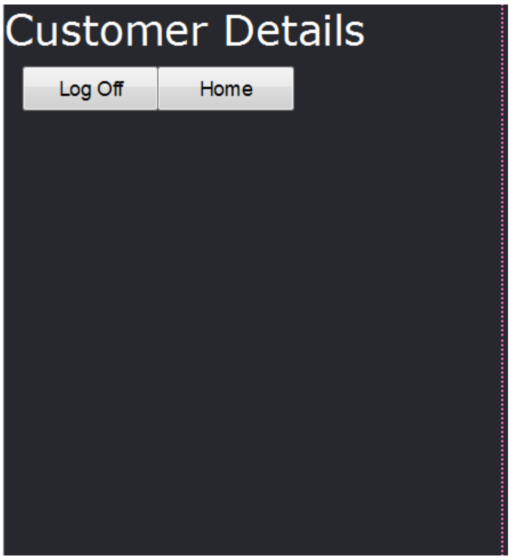
The will cause a brand new screen named VS_CUSTM1 to be displayed, something like this:



You should recognize this type of screen now – it is a "blank" virtual screen that you have decided to name VS_CUSTM1.

Set VS_CUSTM1 up in the normal manner by completing all the steps in the preceding New Screen Checklist table.

Then add a title *Customer Details* and also Home and Log Off buttons to your new VS_CUSTM1 screen so that you have a blank canvas starting point like this:



The Log Off button's script should do this: USERENV.MOBILE.gotoScreen(ENV,"EXIT");
The Home button's script should do this: USERENV.MOBILE.gotoScreen(ENV,"HOME");

Check that your buttons work as expected.

Now *add and style* a label eXtension on the screen that has ***text*** property "Name:".

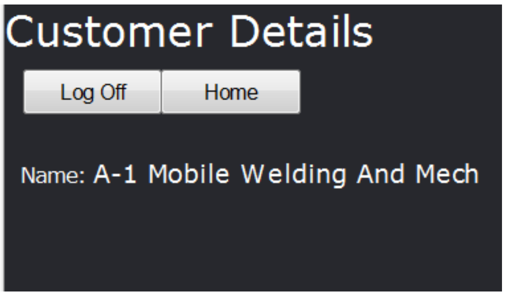
Then add *and style* another label eXtension beside it.

Set its ***text*** property by evaluating this script:

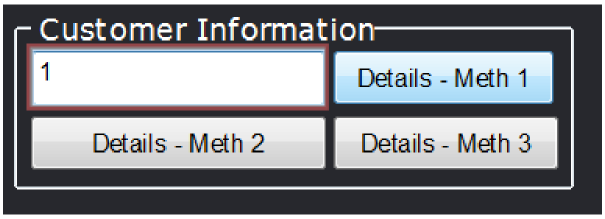
ENV.returnValue = USERENV.MOBILE.currentCUSTOMER.CUSTNAME;

Save your changes.

You should see a customer name appear on the screen like this:



To recap – back on your home screen an SQL command was executed when a customer number was entered and the *Details – Meth 1* button was clicked.



The SQL command produced an aXes table that had row elements named:

```
"CUSTNUMBR","CUSTNAME","CUSTADDR","CUSTCITY","CUSTZIP","CUSTPHONE","CUSTEMAIL"
```

It also set a reference to the first aXes table row into the USERENV.MOBILE object like this:

```
USERENV.MOBILE.currentCUSTOMER = TABLEMANAGER.getTable("AXMBCUST-INQUIRE").child(0);
```

So ….

```
USERENV.MOBILE.currentCUSTOMER.CUSTNAME;
```

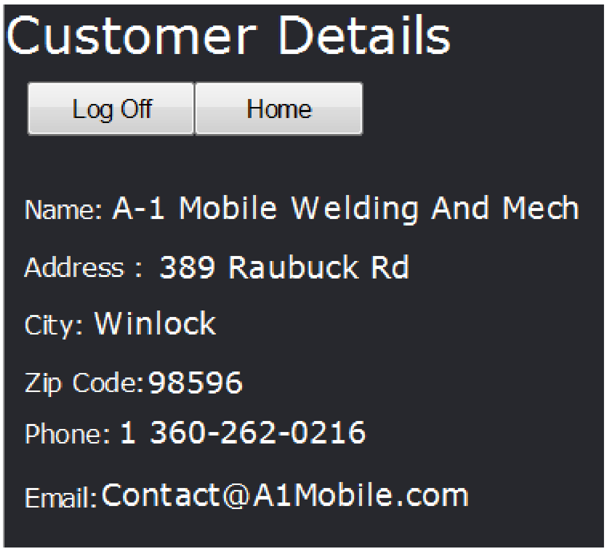
contains the name of the selected customer.

And equally ….

```
USERENV.MOBILE.currentCUSTOMER.CUSTPHONE;
```

contains the customer's phone number.

So by using the information in USERENV.MOBILE.currentCUSTOMER you should be able to produce a screen that looks something like this:



Customer Details

Name: A-1 Mobile Welding And Mech

Address : 389 Raubuck Rd

City: Winlock

Zip Code: 98596

Phone: 1 360-262-0216

Email: Contact@A1Mobile.com

Save all your changes and test your inquiry with a number of different customer numbers.

You have now built a basic customer inquiry.

The key thing to understand is that you did this by using a virtual screen and by client side scripting alone.

You did not need to build any special-purpose server-side RPG programs to do this.

Time to try out the Real Thing via Desktop, Phone or Emulator

Before looking at the other two ways you can do a customer inquiry, it is probably worth going through how you can test your application as the "real thing".

Up until now you have been running driver program TU4DRIVER by calling it from command entry. For application development that is fine – but in a real application you would want to run it as soon as the user logged on.

In the AXESDEMO library there is a CL program called TU4LOGON that does this:

```
PGM

DCL        VAR(&USER) TYPE(*CHAR) LEN(10)

CHGJOB     BRKMSG(*HOLD) STSMSG(*NONE)
MONMSG     MSGID(CPF0000 MCH0000)

RTVJOBA    USER(&USER)
MONMSG     MSGID(CPF0000 MCH0000)

CLRMSGQ    MSGQ(*WRKSTN)
MONMSG     MSGID(CPF0000 MCH0000)

CLRMSGQ    MSGQ(&USER)
MONMSG     MSGID(CPF0000 MCH0000)

ADDLIBLE   LIB(AXESDEMO)
MONMSG     MSGID(CPF0000 MCH0000)

CALL       PGM(TU4DRIVER)
MONMSG     MSGID(CPF0000 MCH0000)

SIGNOFF

ENDPGM
```

TU4LOGON can act as a simple user logon program for purposes of this tutorial.

You will probably need to duplicate program Tu4LOGON from library AXESDEMO into a library that it is in your user profile's library list (e.g.: QGPL).

Next - typing in long URLs on a smart phone is error prone. You can improve this situation by making a simple HTML document in the root of your aXes system.

For example, if you create a file named TUT4.HTML in the root of your aXes system containing this HTML ….

```
<html>
<head>
</head>
<body>
<script type="text/javascript">
window.location.replace("http://<aXesHost>/ts/ts2/mobile.html?definitionSet=<ppppp>&user=<uuu
uu>&program=TU4LOGON");
</script>
</body>
</html>
```

Where:

- <aXesHost> is your aXes Host
- <pppppp> is your project folder's name
- <uuuuuu> is your user profile name.

Note: After creating file TU4.HTML make sure that user *PUBLIC has *R (and only *R) access rights to the file. Use the IBM i WRKLNK command to check and change this.

Now on your phone you can enter the simplified URL instead:

http://<axes Host>/tut4.html

One other thing worth noting is that ultimately it is **mobile.html** that is opened to access aXes – not the default index.html. This causes an aXes mobile session to be started instead of a normal aXes-TS type 5250 session.

This example causes program TU4LOGON to be started by specifying it on the 5250 logon screen. Your IBM i system configuration might not allow this. If so, you should set up a testing user profile that has TU4LOGON specified as its initial program.

Having set up a simple entry point like http://<axes Host>/tut4.html you can now try out your application for real in a number of ways:

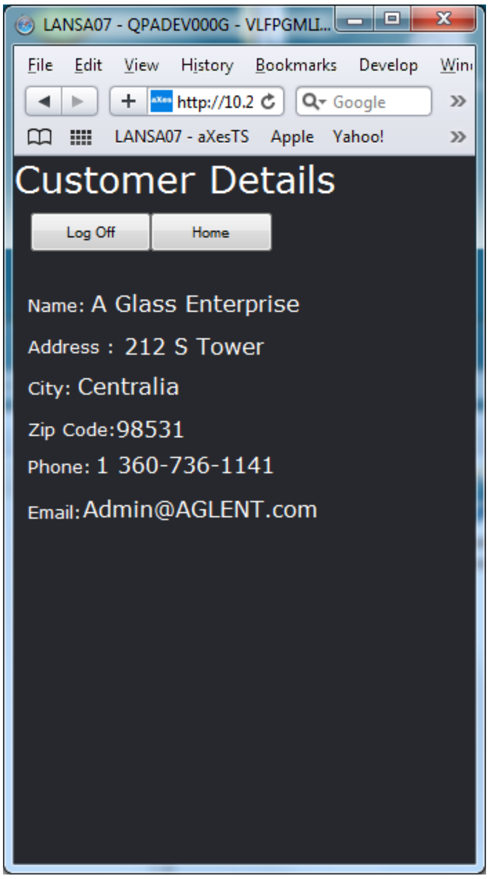
- From a smart phone such an iPhone or an Android 2.1 (or later) phone.

- From a desktop browser – but you must use Safari or Chrome - you cannot use IE.

- In a phone emulator or simulator. These are available for Windows and Mac systems. The complexity of their set up varies and is beyond the scope this tutorial.

For example, you can test your application:

In the Safari browser on Windows desktop:



LANSA07 - QPADEV000G - VLFPGMLI...
File Edit View History Bookmarks Develop Win
http://10.2
Google
LANSA07 - aXesTS Apple Yahoo!

Customer Details

Name: A Glass Enterprise

Address : 212 S Tower

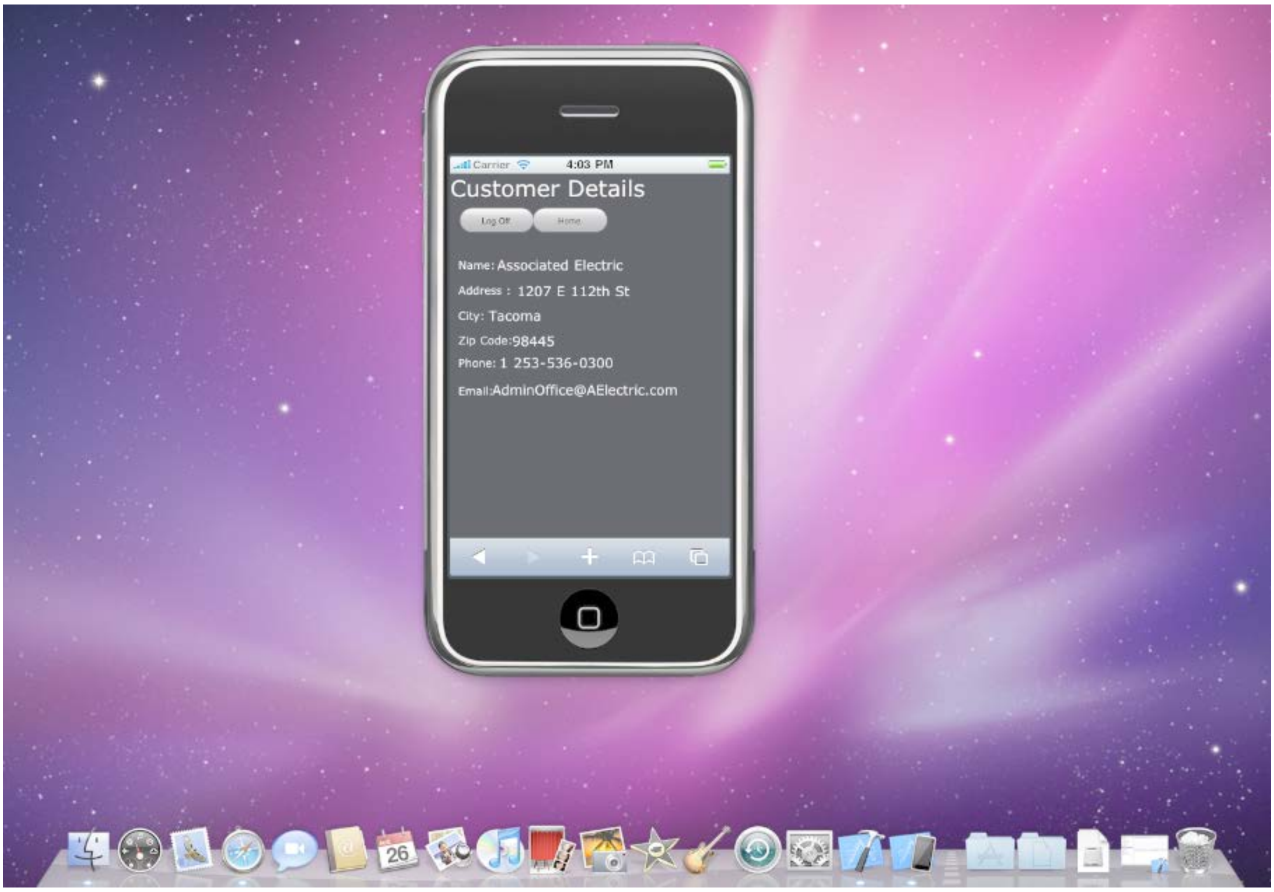
City: Centralia

Zip Code:98531

Phone: 1 360-736-1141

Email: Admin@AGLENT.com

In an iPhone emulator on a Mac system:



Customer Details

Name: Associated Electric

Address : 1207 E 112th St

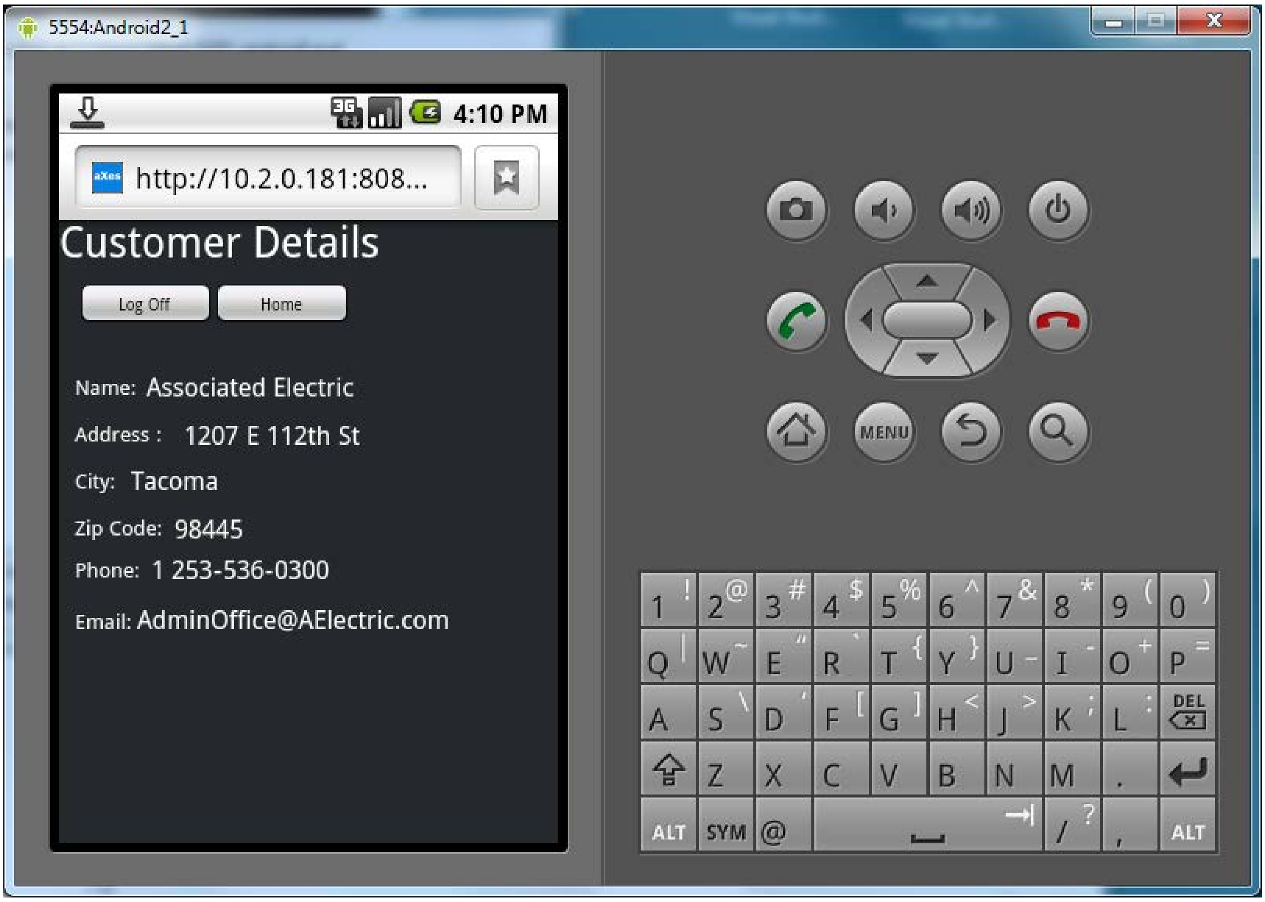
City: Tacoma

Zip Code:98445

Phone: 1 253-536-0300

Email:AdminOffice@AElectric.com

In an Android emulator:



Customer Details

Name: Associated Electric

Address : 1207 E 112th St

City: Tacoma

Zip Code: 98445

Phone: 1 253-536-0300

Email: AdminOffice@AElectric.com

You should try your application out on one of these test platforms.

Remember that ultimately you must try out your application on a real Smart Phone.

Method 2 - Using a Service or Subroutine Program Approach

In the customer inquiry method 1 tutorial you built a virtual screen to display customer details. This was done entirely by the client and no specialized server RPG side RPG program was required.

The second example involves creating an RPG service or subroutine style program (it is called a service or subroutine style program because it provides a service to the smart phone application without having a user interface of its own).

First you need to write and compile your RPG "service" or "subroutine" program.

The shipped example is **TU4CUSTM2P** with the source code in the QTUTORIAL file.

The RPG code looks like this:

```
FAXMBCUST  IF   E           K DISK
 D/copy QTUTORIAL,TU4INCLUDA
 C     *ENTRY        PLIST
 C                   PARM                    StateBlock
 C/free
      CustNumbr = %dec(Info_1:7:0);
      StateBlock = ' ';
      Chain CustNumbr AXMBCUSTR;
      If NOT %Found(AXMBCUST);
         Info_1 = 'No customer with number ' + %char(CustNumbr)
                + ' could be found.';
          NextAction = 'MESSAGE';
      else;
          info_1 = %char(CustNumbr);
          info_2 = CustName;
          info_3 = CustAddr;
          info_4 = CustCity;
          info_5 = CustZip;
          info_6 = CustPhone;
          info_7 = CustEmail;
          NextAction = 'VS_CUSTM2';
     endif;
     *inlr = *on;
     return;
 /end-free
```

This program receives a data structure parameter named StateBlock (defined by the /COPY member TU4INCLUDA).

StateBlock contains sub-fields NEXTACTION and INFO_1 through INFO_10.

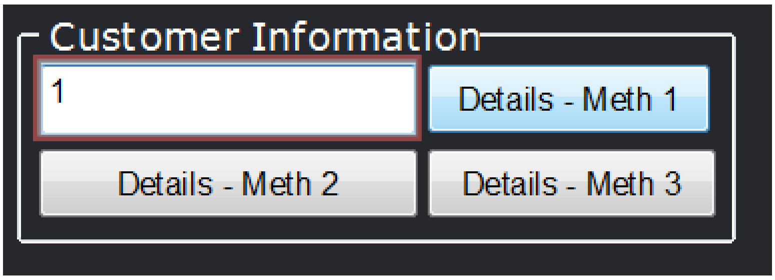
Logically it does this:

- Extracts a customer number from the INFO_1 field passed in.
- Reads the associated customer record (row) from file (table) AXMBCUST.
- If not found, it sets INFO_1 to an error message and NEXTACTION to 'MESSAGE'.
- Otherwise it maps the customer details into fields INFO_1 through INFO_7 and NEXTACTION to 'VS_CUSTM2'.

If you want to make your own version of this program, copy the source code from member **TU4CUSTM2P** in source file and compile it with a different name.

Now log on to aXes as a developer.

Go to your applications HOME page by calling TU2DRIVER and set up the ***onClick*** scripting for your ***Details – Meth 2*** button.



The scripting you need is something like this:

```
var sCustNo = FIELDS("CUSTOMERNUMBER").getValue();

var iCustNo = parseInt(sCustNo,10);

if ((isNaN(iCustNo))||(iCustNo < 1)||(iCustNo > 9999999))
   alert("Customer number " + sCustNo + " is not a valid customer number.");

else
   USERENV.MOBILE.gotoScreen(ENV,"TU4CUSTM2P",sCustNo);
```

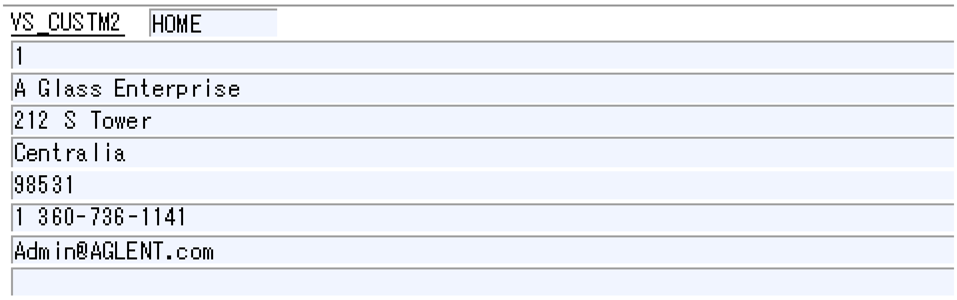
The logic is:

- The customer number is crudely validated.
- If it is no good, a message box is displayed.
- Otherwise the TU4DRIVER program is instructed to call program TU4CUSTM2P, passing the customer number value into it in field INFO_1 (if you have your own program use its name instead of TU4CUSTM2P).

Now try out your ***Details – Meth 2*** button with some bad customer numbers first to check the error handling.

Now try a valid customer number - like 1, 2 or 5.

The resulting screen should look something like this:



```
VS_CUSTM2   HOME
1
A Glass Enterprise
212 S Tower
Centralia
98531
1 360-736-1141
Admin@AGLENT.com
```

By now you should be able to identify this as a new virtual screen named **VS_CUSTM2.**

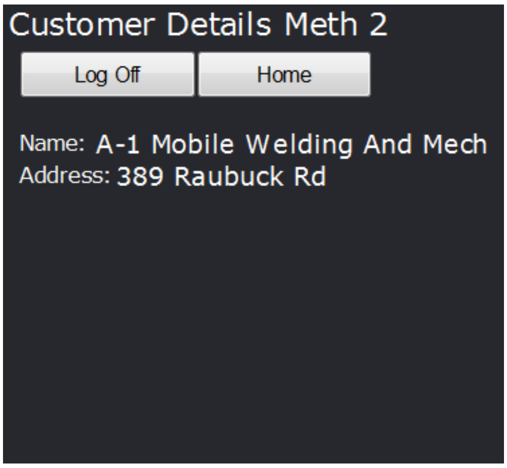
You can also see the information returned by RPG program TU4CUSTM2P (or your version).

On the new virtual screen named VS_CUSTM2 you should now complete these steps:

- Complete the preceding New Screen Checklist activities.
- Add and style a title like "Customer Details Meth 2"
- Add a Home push button that does USERENV.MOBILE.gotoScreen(ENV,"HOME");
- Add a Log Off push button that does USERENV.MOBILE.gotoScreen(ENV,"EXIT");
- Add and style a label with ***text*** property "Name:"
- Add and style a label beside the name that sets its ***text*** by executing the script ENV.returnValue = USERENV.MOBILE.getINFO(ENV,2);
- Add and style a label with ***text*** property "Address:"

- Add and style a label beside the name that sets its ***text*** by executing the script ENV.returnValue = USERENV.MOBILE.getINFO(ENV,3);

The result should look something like this:


Customer Details Meth 2

Name: A-1 Mobile Welding And Mech
Address: 389 Raubuck Rd

By adding more labels you can display the values of 5250 fields INFO_4 (City), INFO_5 (Zip Code), INFO_6 (Phone Number) and INFO_7 (Email address) by using the getINFO(ENV,N) function to retrieve the values. This equates directly to these lines in the RPG program TU4CUSTM2P (or your version of it):

```
info_1 = %char(CustNumbr);
info_2 = CustName;
info_3 = CustAddr;
info_4 = CustCity;
info_5 = CustZip;
info_6 = CustPhone;
info_7 = CustEmail;
```

The key point to understand in this method is that the data extraction is performed by an RPG program – but the final mapping of that information onto the screen is performed by client side scripting. Here the server side RPG program TU4CUSTM2P (or your version) is acting as a "service" or "subroutine" to the client side logic.

Optional Technical Note:
The INFO_1 through INFO_10 fields are used to map data into and out of RPG programs.

There are only 10 of these fields in the tutorial – but of course you could add more.

Also note that they are 130 bytes long (aXes nearly always uses 132 wide screens) – so by specific positioning or by using the separator character and the JavaScript "split" function you can easily and generically compact multiple items into one 5250 screen field.

There are also various other techniques for passing very large amounts of data from RPG programs out to the client application - including data queues and JSON strings. Contact your product vendor for more details and examples.

Method 3 - Using a classic 5250 RPG Program

In the customer inquiry method 1 tutorial you built a virtual screen to display customer details. This was done entirely by the client and no specialized server RPG side RPG program was required.

The method 2 tutorial involved creating an RPG service or subroutine program. It was called to return agreed values which the client side then arranged on the screen.

The final method (3) involves using an RPG program that uses a real 5250 screen. This is classical way of presenting information from an RPG program – possibly the way that you are most familiar with.

The display file is named **TU4CUSTM3D** and its DDS looks like this:

```
A                                      DSPSIZ(27 132)
A                                      REF(AXMBCUST)
A          R RECORD
A            THISSCREEN    10A  B  1  2DSPATR(PR)
A            NEXTACTION    10A  B  1 13
A                                  3  2'Number'
A                                  4  2'Name'
A                                  5  2'Address'
A                                  6  2'City'
A                                  7  2'Zip Code'
A                                  8  2'Phone'
A                                  9  2'Email'
A            XUSTNUMBR R        B  3 20REFFLD(CUSTNUMBR)
A            XUSTNAME  R     O  B  4 20REFFLD(CUSTNAME) CHECK(LC)
A            XUSTADDR  R     O  B  5 20REFFLD(CUSTADDR) CHECK(LC)
A            XUSTCITY  R     O  B  6 20REFFLD(CUSTCITY) CHECK(LC)
A            XUSTZIP   R        B  7 20REFFLD(CUSTZIP)  CHECK(LC)
A            XUSTPHONE R        B  8 20REFFLD(CUSTPHONE) CHECK(LC)
A            XUSTEMAIL R        B  9 20REFFLD(CUSTEMAIL)  CHECK(LC)
A            ERRORMSG     700O  O 10 20
```

This a really simple set of DDS. A couple of things worth noting are:

- The use of fields THISSCREEN and NEXTACTION on line 1. These allow this screen to be easily hooked into the logic of driver program TU4DRIVER. They will be hidden on the resulting screen of course.

- The 700 byte ERRORMSG field on line 10 position 20. This will be used to send composite error message details out to the smart phone. There is no need to worry about the length – only as much data as the field contains is actually transmitted – not all 700 bytes.

- The simplicity of the screen layout. All the labels are arranged down the screen starting at position 2 and all the data fields similarly at position 20. The reason for this simple layout is that you are going to move these fields around and even change the way they are visualized in the final result – so any consideration of 5250 screen positions is really moot. It is probably easier to create DDS like this by editing the DDS source file instead of using a 5250 screen design facility. Smart phone screens are ultimately not 5250 screens in the normal sense - so don't waste time "designing" the 5250 screen layouts as 5250 screens.

The associated RPG program is named **TU4CUSTM3P** and its RPG code looks like this:

```
FTU4CUSTM3DCF   E             WORKSTN
FAXMBCUST  UF   E           K DISK
D/copy QTUTORIAL,TU4INCLUDA
D                 DS
D ThisProgram                         Like(ThisScreen)
D Request                             Like(Info_1)
D ErrorCount                     7P00
C     *ENTRY        PLIST
C                   PARM                    StateBlock
C/free

     ThisProgram = ThisScreen;

     Dou (NextAction <> ThisProgram);

        Request  = Info_1;
        Monitor;
           CustNumbr  = %dec(Info_2:7:0);
        On-Error 105;
           CustNumbr  = 0;
        EndMon;

        StateBlock = ' ';
        ThisScreen = ThisProgram;
        NextAction = ThisProgram;
        Info_1     = 'UPDATE';
        Info_2     = %char(CustNumbr);
        ErrorCount = 0;
        ErrorMsg   = 'NONE';

        if (Request = 'UPDATE');
           Exsr Update_Customer;
        else;
           Exsr Display_Customer;
        endif;

        If (NextAction = ThisProgram);
          Exfmt Record;
        Endif;

     EndDo;

     *inlr = *on;
     return;
     // ---------------------------------------------
     Begsr  Display_Customer;
        Chain CustNumbr AXMBCUSTR;
        If NOT %Found(AXMBCUST);
            StateBlock = ' ';
            Info_1 = 'No customer with number ' + %char(CustNumbr)
                      + ' could be found (from TU4CUSTM3P)';
            NextAction = 'MESSAGE';
         else;
            Unlock AXMBCUST;
            Exsr Map_Out;
         Endif;
      Endsr;
      // ---------------------------------------------
      Begsr  Update_Customer;
         Exsr Validate_Customer;
         If (ErrorCount = 0);
            StateBlock = ' ';
            Chain CustNumbr AXMBCUSTR;
           If NOT %Found(AXMBCUST);
              Info_1 = 'No customer with number ' + %char(CustNumbr)
                     + ' could be found (from TU4CUSTM3P)';
           else;
              Exsr Map_In;
              Update AXMBCUSTR;
              Info_1 = 'Details of customer ' + %trim(CustName)
                     + ' have been sucessfully updated.';
           Endif;
           NextAction = 'MESSAGE';
        Endif;
     Endsr;
     // ---------------------------------------------
     Begsr  Validate_Customer;
        ErrorMsg = ' ';
        If (XustName = ' ');
           ErrorCount += 1;
           ErrorMsg = %trim(ErrorMsg)
                    + ' A customer name is required.';
        endif;
        If (XustAddr = ' ');
```

```
            ErrorCount += 1;
            ErrorMsg = %trim(ErrorMsg)
                     + ' An address is required for all customers.';
         endif;
         If (XustPhone = ' ');
            ErrorCount += 1;
            ErrorMsg = %trim(ErrorMsg)
                     + ' A phone number is required for all customers.';
         endif;
         If (ErrorCount = 0);
            ErrorMsg = 'NONE';
         Endif;
      Endsr;
       // --------------------------------------------
      Begsr  Map_Out;
         XustNumbr = CustNumbr;
         XustName  = CustName;
         XustAddr  = CustAddr;
         XustCity  = CustCity;
         XustZip   = CustZip;
         XustPhone = CustPhone;
         XustEMail = CustEMail;
      Endsr;
      // --------------------------------------------
      Begsr  Map_In;
         CustName  = XustName;
         CustAddr  = XustAddr;
         CustCity  = XustCity;
         CustZip   = XustZip;
         CustPhone = XustPhone;
         CustEMail = XustEMail;
      Endsr;
  /end-free
```

TU4CUSTM3P a fairly simple RPG program. A couple of things worth noting are:

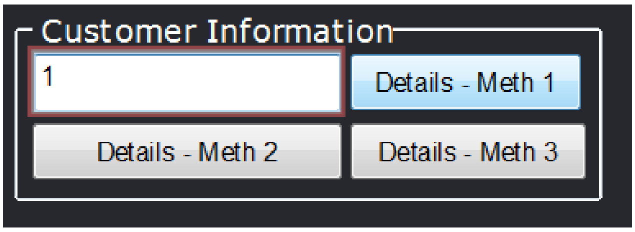
- It's designed to be called by TU4DRIVER because it has the common StateBlock parameter (a data structure defined in RPG source member TU4INCLUDA).
- It loops until the client instructs it to do something else - in which case it ends and yields control back to TU4DRIVER.
- It can display customer details.
- It can update customer details.
- It has validation rules that can cause an update to be rejected.

First you need to write and compile your 5250 display file and the RPG program.

The shipped example display file is **TU4CUSTM3D** (source in QTUTORIAL) and the RPG program is called **TU4CUSTM3P** (source in QTUTORIAL).

You can either use the shipped objects (which are already compiled) or copy the source code and create your own versions with different names.

Next, log on to aXes as a developer, go to your application's home page by calling TU4DRIVER and set up the ***onClick*** scripting for your ***Details – Meth 3*** button



The scripting you need is something like this:

```
/* Get the customer number from the screen */

var sCustNo = FIELDS("CUSTOMERNUMBER").getValue();

/* Invoke program TU4CUSTM3P sending request and customer number */

USERENV.MOBILE.gotoScreen(ENV,"TU4CUSTM3P","DISPLAY",sCustNo);
```

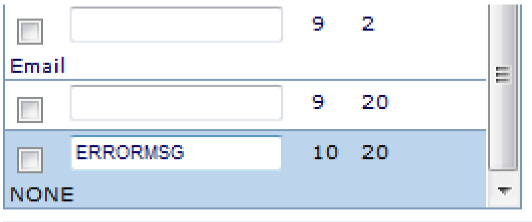
Now try out your ***Details – Meth 3*** button.

This time the screen displayed is not a virtual screen – it is a real 5250 screen presented by program TU4CUSTM3P according to the DDS defined in display file TU4CUSTM3D.

By now you should know the drill – HOWEVER – in this case most of the fields on the screen ***should not be hidden***.

Fields THISSCREEN (line 1 pos 2), NEXTACTION (line 1 pos 13) and ERRORMSG (line 10 pos 20) should be hidden - all of the other fields should be left visible.

Additionally, the long error message field name on line 10 at position 20 should be named ERRORMSG in the aXes screen definition, like this:



□		9	2
Email			
□		9	20
□	ERRORMSG	10	20
NONE			

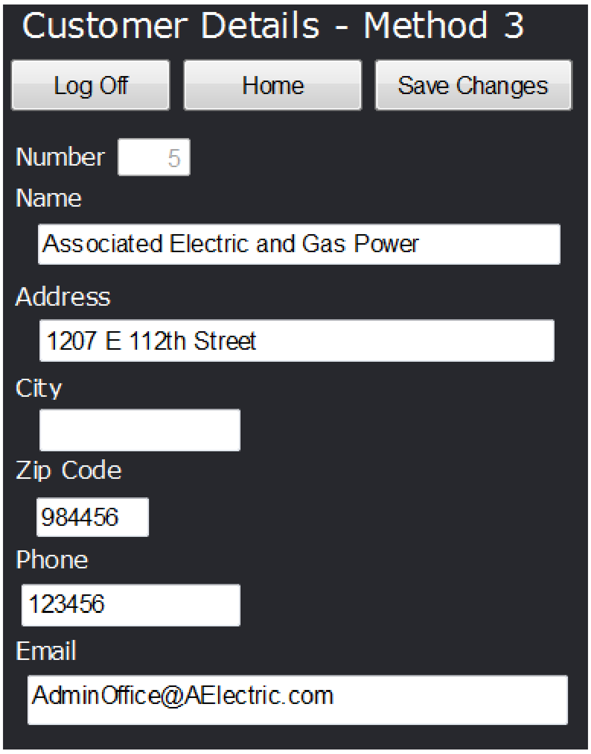
Save your screen definition changes after you apply a name to the ERRORMSG field.

Now you should:

- Complete the New Screen Checklist
- Add a screen title "Customer Details – Method 3".
- Add the standard "Log Off" and "Home" push buttons
- Move and size the other label and input fields. Style the labels.
- Add a "Save Changes" push button. Execute this script when it is clicked:

```
USERENV.MOBILE.gotoScreen(ENV,"TU4CUSTM3P");
```

The resulting screen looks something like this (note that it has slightly different style to the screens used in the preceding tutorial steps):



Customer Details - Method 3

Number 5

Name

Associated Electric and Gas Power

Address

1207 E 112th Street

City

Zip Code

984456

Phone

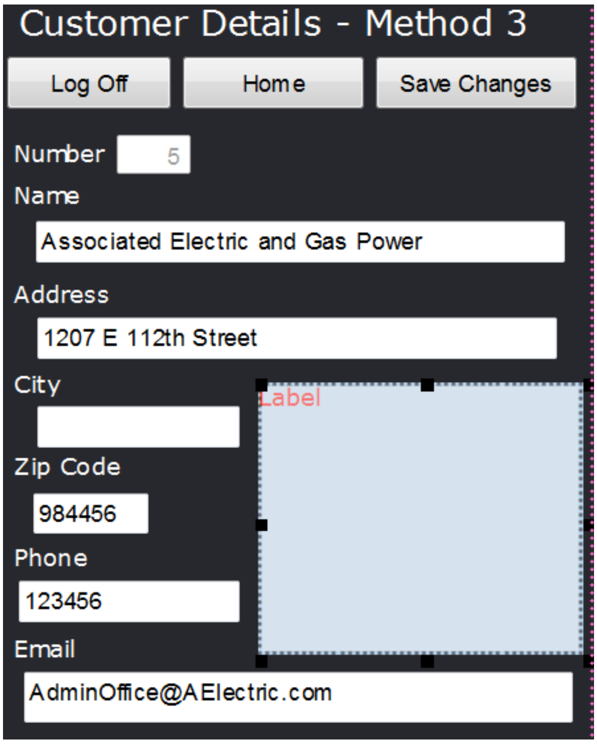
123456

Email

AdminOffice@AElectric.com

Save all your screen customization changes.

Now add a label eXtension to your screen and apply base style errorMessage to it, position it in a free area on your screen – something like this:



Customer Details - Method 3

Number 5

Name

Associated Electric and Gas Power

Address

1207 E 112th Street

City

Zip Code

984456

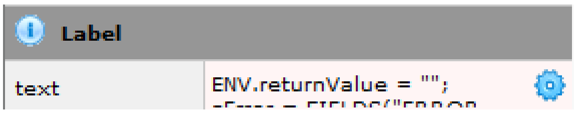
Phone

123456

Email

AdminOffice@AElectric.com

Change it so that its text property is derived by executing this script:



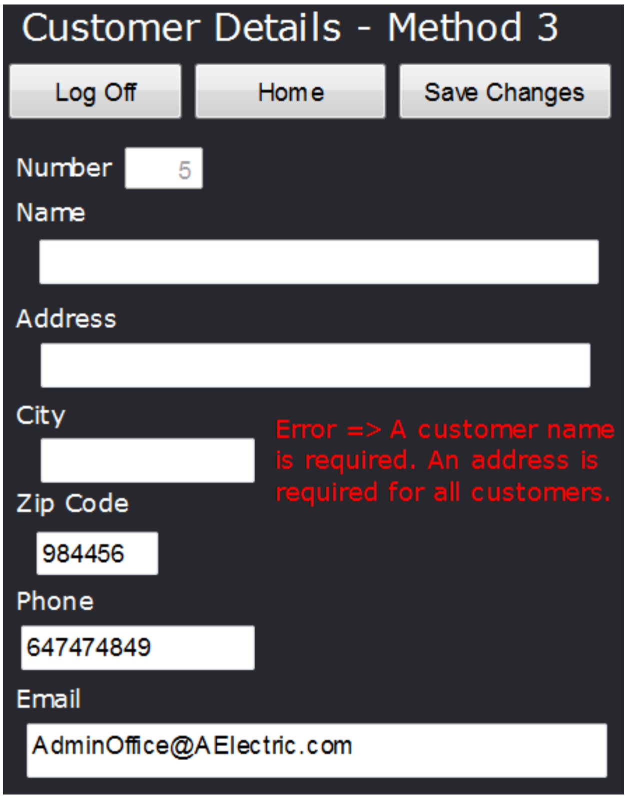
Label

text ENV.returnValue = "";

```
ENV.returnValue = "";
sError = FIELDS("ERRORMSG").getValue();
if (sError != "NONE") ENV.returnValue = "Error =>" + sError;
```

This script gets the value of the field you named ERRORMSG (which is hidden from view) and puts its value into the eXtension – *providing that it does not contain the message "NONE".*

The end result is that when you try to update a customer with a blank name, address or phone number the messages are made visible to the user …



Customer Details - Method 3

Number 5

Name

Address

City

Error => A customer name is required. An address is required for all customers.

Zip Code 984456

Phone 647474849

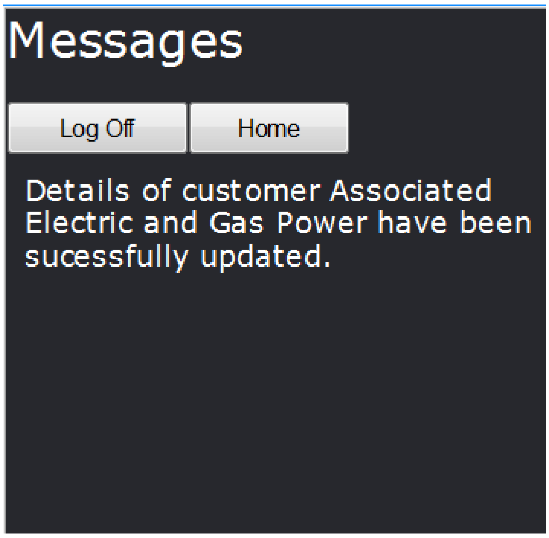
Email AdminOffice@AElectric.com

Save your customization changes and test your new screen. Try saving after blanking out the customer name.

There are many ways to present error messages and this technique is very simple.

Its main benefit is that it shows all the error messages to the user without needing another screen interaction. When designing a real application you should develop a standard way to show error messages and use it consistently on all screens.

Notice that when you update a customer screen your MESSAGE screen is displayed to confirm the action to the user:



Messages

Details of customer Associated Electric and Gas Power have been sucessfully updated.

Doing this type of positive confirmation of user's actions is useful on smart phones where variable response times are common.

The "Messages" button on your Home screen is now redundant.

In this application there is no reason for the user to display the messages (and in fact it may show incorrect details at times). Confirmation messages are presented automatically by the application when it is appropriate.

Remove the "Messages" button from your home page now and save your changes.

Review of the 3 Interface and Access Methods

In the last 3 tutorial steps the 3 main ways of assembling and processing information on smart phone screens has been covered. These are:

Method	Description	Virtual Screen Used	Characteristics
1	Client Side	Yes	The client scripting dynamically creates the screen content entirely by executing script and SQL commands, etc. No special server side RPG logic is required to do this.
2	Client and "Service" or "subroutine" program	Yes	The client scripting executes an RPG (or even CL) server side program as a service or subroutine to gather or process information required for the client side screen.
3	Server 5250 program	No	A classic RPG 5250 server program is used. It processes information through classic 5250 DDS. The client side is limited to just arranging how the information is presented.

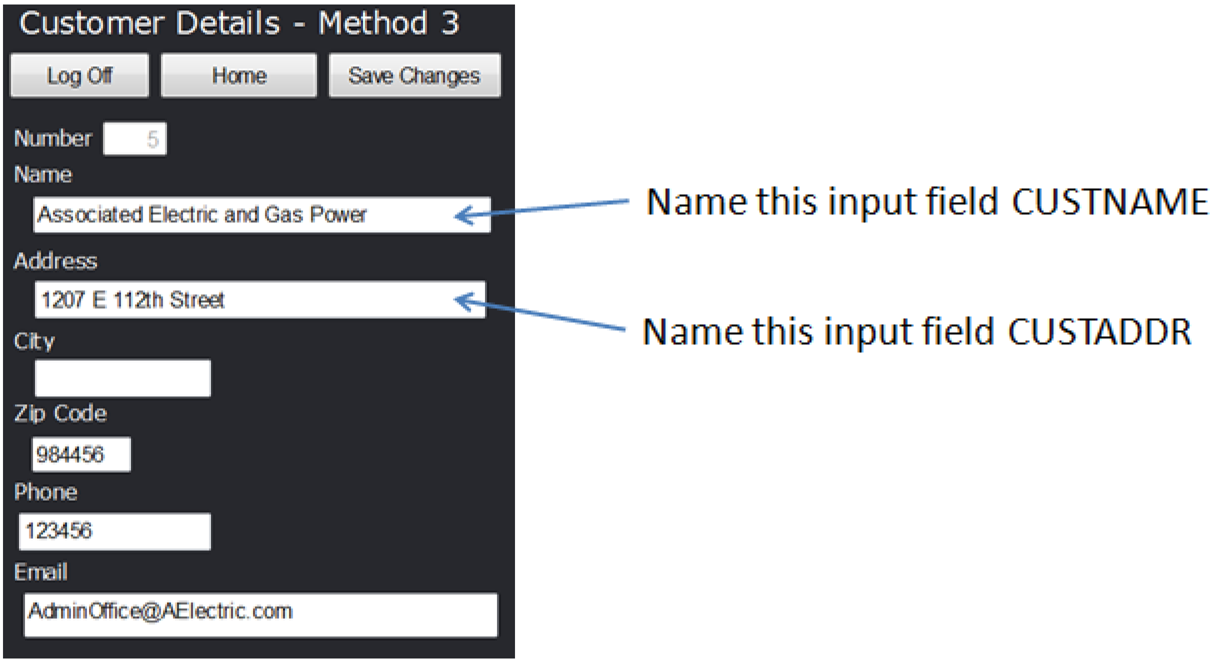
If you understand all three of these methods you should be able to decide which one to apply to which problem when designing and implementing a smart phone application.

The following tutorial steps are optional – they cover a broad range of topics that you may not need to know about right now.

Optional Steps

Client Side Validation

In the preceding Customer Details – Method 3 example data validation is performed by server RPG program TU4CUSTM3P. You can also easily do client side validation. Display your Customer Details – Method 3 screen and name the input fields as indicated.



Customer Details - Method 3

Number: 5

Name: Associated Electric and Gas Power

Address: 1207 E 112th Street

City:

Zip Code: 984456

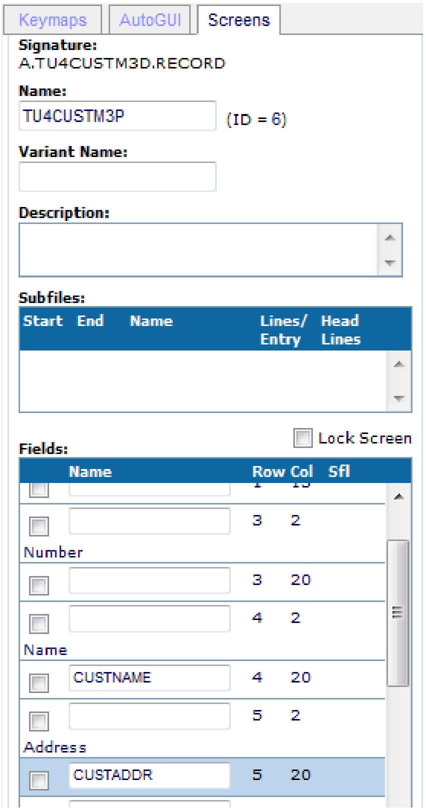
Phone: 123456

Email: AdminOffice@AElectric.com

Name this input field CUSTNAME

Name this input field CUSTADDR

Since these are real 5250 screen fields and not newly added eXtensions you name them here:



Signature: A.TU4CUSTM3D.RECORD

Name: TU4CUSTM3P (ID = 6)

Variant Name:

Description:

Subfiles:

Start	End	Name	Lines/ Entry	Head Lines

Fields: □ Lock Screen

	Name	Row	Col	Sfl
□		3	2	
Number				
□		3	20	
□		4	2	
Name				
□	CUSTNAME	4	20	
□		5	2	
Address				
□	CUSTADDR	5	20	

Save your changes.

Now locate the screen level onLeave property and change it to execute this script:

```
var CUSTNAME = FIELDS("CUSTNAME").getValue();
var CUSTADDR = FIELDS("CUSTADDR").getValue();
var errormessage = "";

if (CUSTNAME == "") errormessage += " You must specify a customer name.";
if (CUSTADDR == "") errormessage += " You must specify a customer address.";

if (errormessage != "")
{
   alert(errormessage);
   ENV.returnValue = false; /* Stop send of data to server */
}
else
{
   USERENV.MOBILE.onLeave(ENV);
}
```

Save your changes.

You should now not be able to save customer changes unless you have specified a name and address – and when the data is invalid it is stopped before it is sent to the server.

Validating at the client and then (re)validating at the server is a strongly recommended practice.

Remembering Values and Initial Values

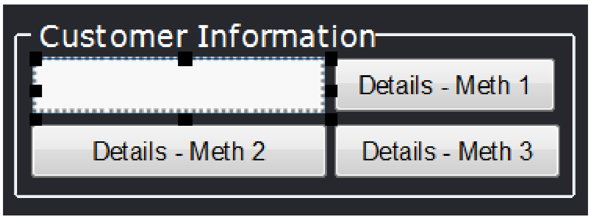
In the preceding tutorial steps you may have noticed that the customer number is lost every time you (re)display the home page. This could be annoying and it is easily solved.

Edit the USERENV object and add a new property to your USERENV.MOBILE object definition:

```
              currentCUSTOMER   : null,

              HOME_CustomerNumber : "1", /* Remembered Customer Number */

              deviceWidth       : function(env) { env.returnValue = 320; },
```

Save your USERENV changes and follow any restart instructions.

On your home page the field where they enter the customer number is named CUSTOMERNUMBER:



Change the field CUSTOMERNUMBER to derive its value by executing this script:

```
          ENV.returnValue = USERENV.MOBILE.HOME_CustomerNumber;
           if (USERENV.MOBILE.HOME_CustomerNumber != null)
           {
              FIELD.setValue(USERENV.MOBILE.HOME_CustomerNumber);
           }
```

Change the onLeave event of your home page to be like this:

```
        USERENV.MOBILE.onLeave(ENV);
        USERENV.MOBILE.HOME_CustomerNumber = FIELDS("CUSTOMERNUMBER").getValue();
```

You should now see that as you start the application the default value of "1" is displayed initially and that when you go back to the home page the last value entered is remembered.

On phones data entry can be difficult – so the more places you can remember last used values and set initial defaults that do not need to be changed, the better for the phone user.

Ease of Use – Current and Dense Information Presentation

Smart phone applications are generally harder to work with than devices with full keyboards.

You can make the user's ability to select things easier by maintaining the concept of the "currently selected thing" (e.g.: the current Order, Product, Customer, Spare Part, etc).

To demonstrate this approach, add this to your project's static tables file:

```
DefineObjectInstance {
      className         = "StaticTable",
      name              = "AXMBPART",
      source            = "sql",
      selectSQLcommand   = "PARTNUMBR,PARTDESC,PARTAVAIL,PARTWHOLE,PARTRETL,PARTGIFFL from
AXESDEMO.AXMBPART ORDER BY PARTDESC ",
      resultColumnNames  = {
"PARTNUMBR","PARTDESC","PARTAVAIL","PARTWHOLE","PARTRETL","PARTGIFFL" }
    };
```

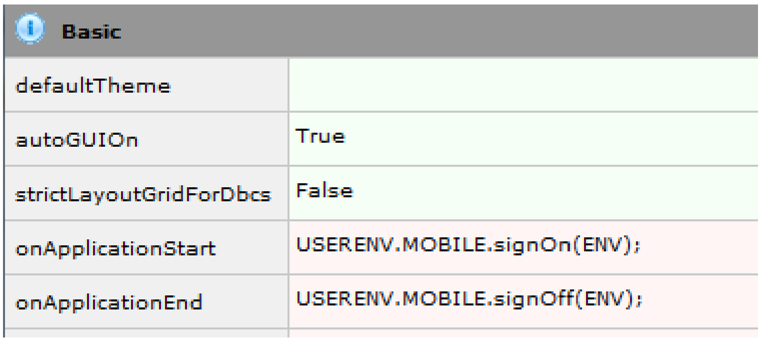
This will create an aXes static table named AXMBPART from the spare parts data base table AXMBPART in the AXESDEMO library.

Then add the highlighted lines to your USERENV.MOBILE object definition:

```
             lockZoom          : function(env) { env.returnValue = true; },

             curAXMBPART : null,  /* Reference to current AXMBPART row */

             signOn : function(env)
             {
                USERENV.MOBILE._isiPhone = (navigator.userAgent.match(/iPhone|iPod/) !=
null);
                env.SHOWAXESMENUBAR(false);
                env.SHOWAXESSTATUSBAR(false);

                /* Load all static tables */
                env.TABLEMANAGER.loadStaticTables(USERENV.staticTablesFile,null,false);

                /* Set a reference to the first row in the AXMBPART table */
                if (this.curAXMBPART == null)
                {
                   var t = env.TABLEMANAGER.getTable("AXMBPART");
                   this.curAXMBPART = t.child(0);
                }

             },
```

These force the loading of all static tables. It then keeps a reference to the "current" spare part in the property USERENV.MOBILE.curAXMBPART. This means that anywhere in your scripting you can now reference USERENV.MOBILE.curAXMBPART.PARTNUMBR (the current spare part's number) or USERENV.MOBILE.curAXMBPART.PARTWHOLE (the current spare part's wholesale price).

Note: The USERENV.MOBILE.signOn() function runs because you have previously set this up in your project:

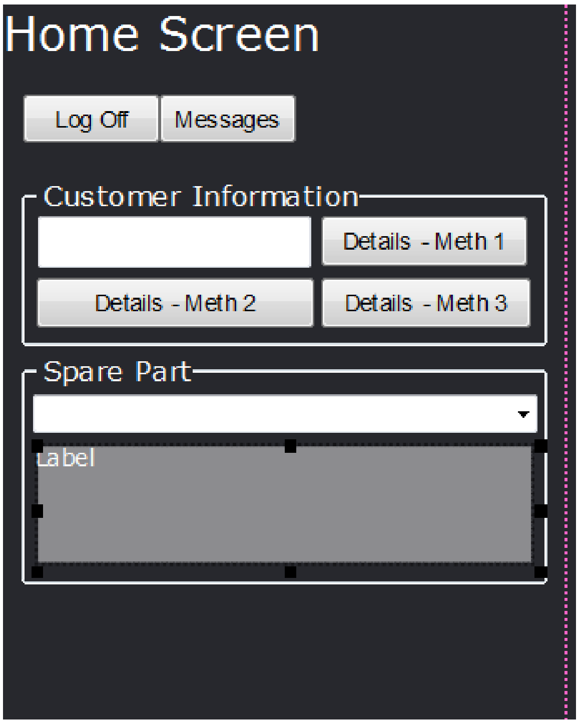


Basic	
defaultTheme	
autoGUIOn	True
strictLayoutGridForDbcs	False
onApplicationStart	USERENV.MOBILE.signOn(ENV);
onApplicationEnd	USERENV.MOBILE.signOff(ENV);

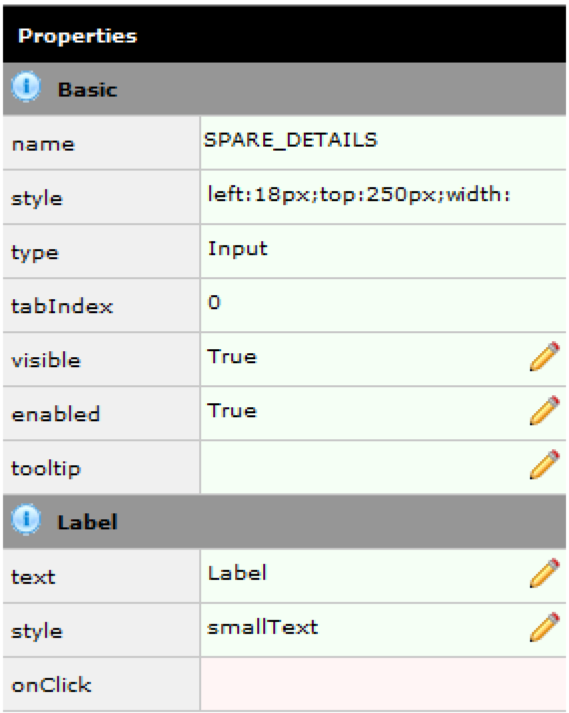
Log off and restart so that all your changes are loaded.

Now that we have a table of spare parts available and a reference to the "current" spare part it is easy to show the user what it is and also allow them to change it.

Add a group box, a drop down and a label field to your **home page**, something like this:



Name the label field SPARE_DETAILS



Properties

Basic

name	SPARE_DETAILS
style	left:18px;top:250px;width:
type	Input
tabIndex	0
visible	True
enabled	True
tooltip	

Label

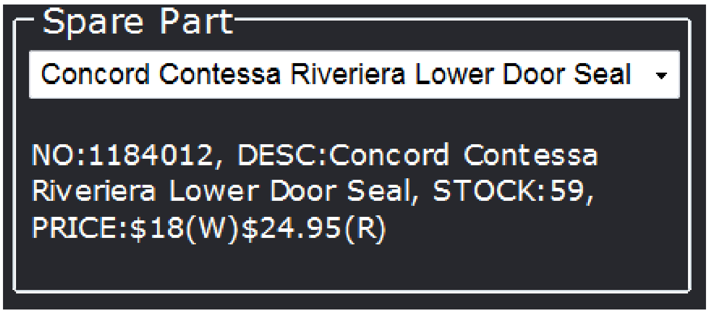
text	Label
style	smallText
onClick	

Then change its ***text*** property to be evaluated by executing this script:

```
var Text = "No spare part details are available";
var P = USERENV.MOBILE.curAXMBPART;

if (P != null)
{
   Text = "NO:" + P.PARTNUMBR;
   Text += ", DESC:" + P.PARTDESC;
   Text += ", STOCK:" + P.PARTAVAIL;
   Text += ", PRICE:$" + P.PARTWHOLE + "(W)$" + P.PARTRETL + "(R)";
}

ENV.returnValue = Text;
```

Save your changes.

The label field should now start to show details about the current spare part (USERENV.MOBILE.curAXMBPART) in a very "dense" manner – something like this:



NO:1184012, DESC:Concord Contessa Riveriera Lower Door Seal, STOCK:59, PRICE:$18(W)$24.95(R)

Now change the drop down to have these properties:

Property	Value	What does it mean?
dataSourceType	Static Table	Fill from a static table
tableName	AXMBPART	The name of the static table
onFillDropDown	ROW.PARTDESC	Use the spare part description as the visible value in the drop down
onSelectValue	ROW == USERENV.MOBILE.curAXMBPART	Select the drop down row if it is the same as the current spare part

onSelectValueChange	USERENV.MOBILE.curAXMBPART = ROW;
	FIELDS("SPARE_DETAILS").refresh();
		When the user selects a spare part in the dropdown, change the current spare part reference to the row selected - then ask the "SPARE_DETAILS" label to refresh its own content based on this change.

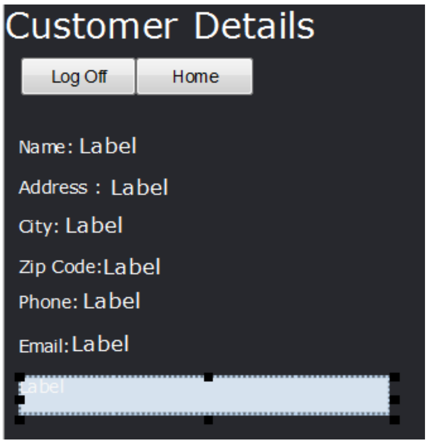
Save your changes.

What you should now find is that you can select different spare parts in the drop down and the label area changes to show more details.

In effect you have created **a very dense spare parts stock and price inquiry screen** – using a very simple structure and very small amount of screen real estate (you could of course use multiple labels instead of one large label to present the information in a classical column format).

The concept of the "current" selection passes on to other screens.

Display your ***Customer Details – Meth 1*** screen and add a label eXtension like this:



Customer Details

Name: Label

Address : Label

City: Label

Zip Code:Label

Phone: Label

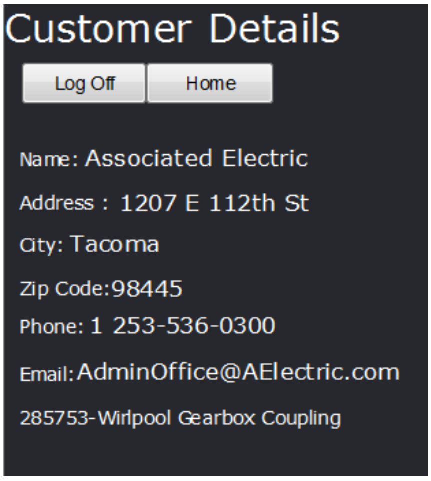
Email:Label

Label

Change the text property of the new label to be dynamically evaluated by this script:

```
var Text = "No spare part details are available";
var P = USERENV.MOBILE.curAXMBPART;
if (P != null) Text = P.PARTNUMBR + "-" + P.PARTDESC;
ENV.returnValue = Text;
```

You should find that the Customer Details – Meth 1 screen displays the details of whatever spare part was selected on the home screen, like this:



Customer Details

Name: Associated Electric

Address : 1207 E 112th St

City: Tacoma

Zip Code:98445

Phone: 1 253-536-0300

Email:AdminOffice@AElectric.com

285753-Wirlpool Gearbox Coupling

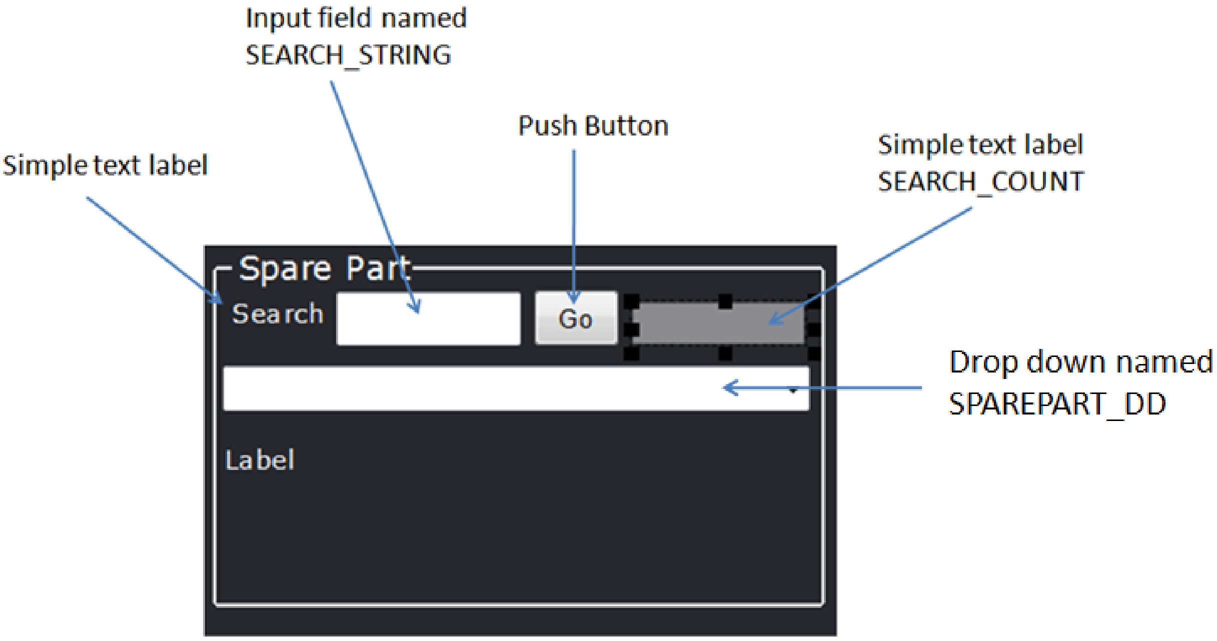
You can also use the current selection and dense display approaches to manage searches on much larger data sets.

Add this new dynamic SQL table definition to your project:

```
   DefineObjectInstance {
      className          = "DynamicTable",
      name               = "AXMBPART-DYNAMIC",
      source             = "sql",
      selectSQLcommand   = "PARTNUMBR,PARTDESC,PARTAVAIL,PARTWHOLE,PARTRETL,PARTGIFFL from
AXESDEMO.AXMBPART where upper(PARTDESC) like '%:SQLVariable_DESC%' ORDER BY PARTDESC ",
      resultColumnNames  = {
"PARTNUMBR","PARTDESC","PARTAVAIL","PARTWHOLE","PARTRETL","PARTGIFFL" }
    };
```

When executed, this SQL request will produce an aXes table named AXMBPART-DYNAMIC that contains all the spare parts that have a description containing a specified value.

Now alter your home screen like this:


Input field named SEARCH_STRING
Push Button
Simple text label
Simple text label SEARCH_COUNT
Drop down named SPAREPART_DD
Spare Part
Search
Go
Label

Double check that you have:

- Named the input field SEARCH_STRING
- Named the small blank label to the right SEARCH_COUNT
- Named the drop down SPAREPART_DD

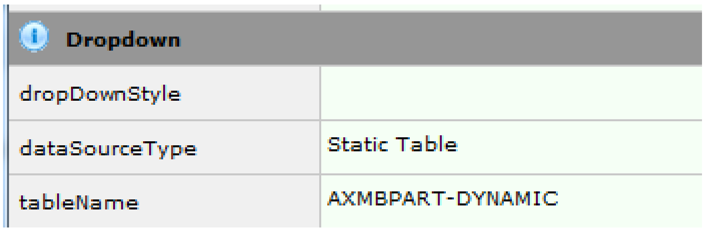
Select the "Go" push button to execute this script when clicked:

```
var SearchValue = FIELDS("SEARCH_STRING").getValue().toUpperCase();
if (SearchValue == "")
{
   alert("You need to enter a search value");
}
else
{
   TABLEMANAGER.loadDynamicTable("AXMBPART-
DYNAMIC",USERENV.dynamicTablesFile,{SQLVariable_DESC:SearchValue});
   var table = TABLEMANAGER.getTable("AXMBPART-DYNAMIC");
   FIELDS("SEARCH_COUNT").setProperty("text",("Found " + table.childCount().toString()));
   USERENV.MOBILE.curAXMBPART = table.child(0);
   FIELDS("SPAREPART_DD").refresh();
   FIELDS("SPARE_DETAILS").refresh();
   FIELDS("SEARCH_COUNT").refresh();
}
```

This script:

- validates what has been input to the SEARCH_STRING field.
- Executes the "AXMB_PART-DYNAMIC" SQL command defined earlier – substituting the SQLVariable_DESC with the user's SEARCH_STRING value.
- Updates the SEARCH_COUNT label's text property with a count of how many rows were found by the SQL command.
- Updates the "current" spare part reference (USERENV.MOBILE.curAXMBPART) to point to the first row of aXes table that the SQL command produced.
- Asks the drop down, spare parts details and search count label all to refresh their displays with the updated information.

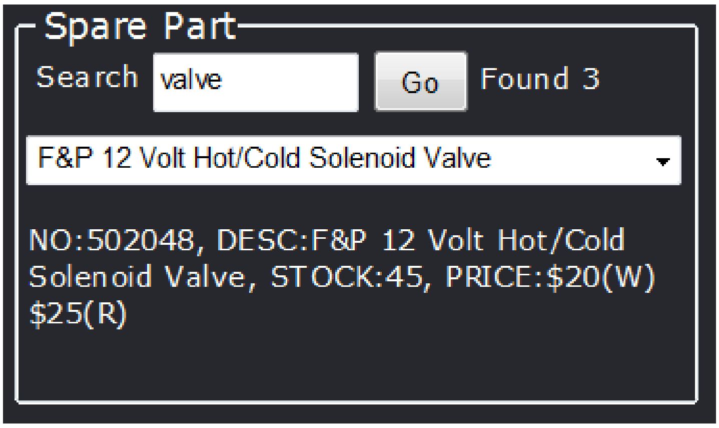
Finally change the static table (yes, *static table*) source for the drop down SPAREPARTS_DD to be from AXMBPART-DYNAMIC - like this:



Dropdown	
dropDownStyle	
dataSourceType	Static Table
tableName	AXMBPART-DYNAMIC

Save all your changes.

You should find that you can now search by clicking the "Go" button for words like "valve" or "agit" to find various spare part valves and agitators ….



Spare Part

Search valve Go Found 3

F&P 12 Volt Hot/Cold Solenoid Valve

NO:502048, DESC:F&P 12 Volt Hot/Cold Solenoid Valve, STOCK:45, PRICE:$20(W) $25(R)

In this simple example you have now managed to produce **a generically searchable spare parts stock/cost enquiry – all managed and presented in a very small and dense screen area.**

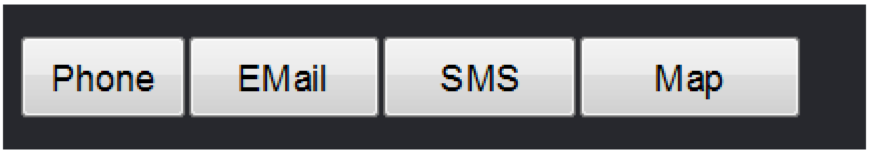
Phone, Mail, SMS and Maps Integration

You can integrate with various capabilities of the smart phone.

Add these example functions to your USERENV.MOBILE object definition in your projects USERENV.JS file, save the changes and then close/restart as directed:

```
             escape : function(s) { return(encodeURIComponent(s)); },

             Phone : function(env,number)
             {
                var URL  = "tel:";
                URL += USERENV.MOBILE.escape(number.toString());
                window.open(URL,"_blank");
             },

             SMS : function(env,number)
             {
                var URL  = "sms:";
                URL += USERENV.MOBILE.escape(number.toString());
                window.open(URL,"_blank");
              },

             Mail : function(env,address,subject,body)
             {
                var URL  = "mailto:";
                URL += address;
                URL += "?subject=" + USERENV.MOBILE.escape(subject);
                if ((body != null) && (body != "")) URL += "&body=" +
USERENV.MOBILE.escape(body);
                window.open(URL,"_blank");
             },

             Map : function(env,address)
             {
                var URL  = "http://maps.google.com/maps?q=";
                URL += USERENV.MOBILE.escape(address);
                window.open(URL,"_blank");
             }, /* Watch out for whether you need or don't need this last comma */
```

Now on your ***Customer Details – Meth 1*** screen add 4 buttons towards the bottom of the screen like this:



Then set up the onClick scripting for each button like this:

Button	onClick Script
Phone	`USERENV.MOBILE.Phone(ENV,USERENV.MOBILE.currentCUSTOMER.CUSTPHONE);`

Email	`var MOB = USERENV.MOBILE;`
	`MOB.Mail(ENV,MOB.currentCUSTOMER.CUSTEMAIL,"This is a subject", "This is a message.");`

SMS	`USERENV.MOBILE.SMS(ENV,USERENV.MOBILE.currentCUSTOMER.CUSTPHONE);`

Map	`var MOB  = USERENV.MOBILE;`
	`var C    = MOB.currentCUSTOMER;`
	`var Addr = C.CUSTADDR + " " + C.CUSTCITY + " " + C.CUSTZIP;`
	`MOB.Map(ENV,Addr);`

The e-mail and map options should work on your PC browser.

To test the phone and SMS options you will have to use a real phone or phone emulator.

Note: some Android versions appear to have technical issues with SMS: requests.

Forget about using ……..

- 5250 pop-up windows in new smart phone applications (never ever do this).
- Function key driven applications (there are no function keys on mobile devices).
- Deep program call stacks. Keep the application flat and simple with simple navigation requests coming from buttons or hyperlinks to move between screens. The more screens that allow you to move directly between them, the easier the application will be to use.

Mostly Forget about using ……..

- Generally you should not use 5250 subfiles in smart phone applications. There may be valid use – but you should consider alternatives first, especially when selecting business instruments like orders, products, customer, etc.

Using a "Flat" Screen Navigation Model

In 5250 applications the use of complex program call stacks is common. You log on to Menu A, then option 6 calls up Menu B, where option 2 calls up Menu C, where option 4 calls up the Order Inquiry program. You now have 4 programs all active and in a complex program stack. This is called "stateful programming". In many respects this type of stateful call stack implementation is unique to the IBM i community.

In smart phone designs you should try to minimize the use of stateful programming techniques. Such a change is actually very easy to make and will usually produce a simpler and far more flexible and extensible application. Additionally, any programs you produce may be very easily converted into servicing other things (e.g. web services) sometime in the future.

Using a Home Screen

You should use a single Home screen. Its role is:

- The screen the user sees initially when they log on.
- It is the place in which most interactions are initiated from.
- All other screens should have a button that goes directly back to the home screen. This is something akin to using F12 on a 5250 screen or the Cancel button on a Windows form.

The home screen always has a log off button.

Log off from every screen

You should have a log off button on every screen.

Using a Messages Screen

You should use a single Messages screen. Its role is to show status, completion or error message details to the user. Normally it has only two buttons enabled. One to go back to the Home screen (allowing another activity to be initiated) and the other to log off.

Portrait or Landscape – Choose One

Generally you should specifically design for Portrait or Landscape mode. In this tutorial all designing is done in Portrait mode. Supporting both modes in your application adds a significant level of complexity. You should assess the effort/reward equation before committing to support both modes.

Prototyping Your Application

If you have completed the preceding tutorial steps you should have figured out by now that it is quite easy to prototype a smart phone application by chaining together different virtual screens.

By adding navigation buttons the basic layout and flow of many different screens can be prototyped, chained together and even demonstrated to stakeholders – even to the point of inserting temporary images onto the screens to demonstrate what the final content (or body) of the application will look like.

Making a Real Application

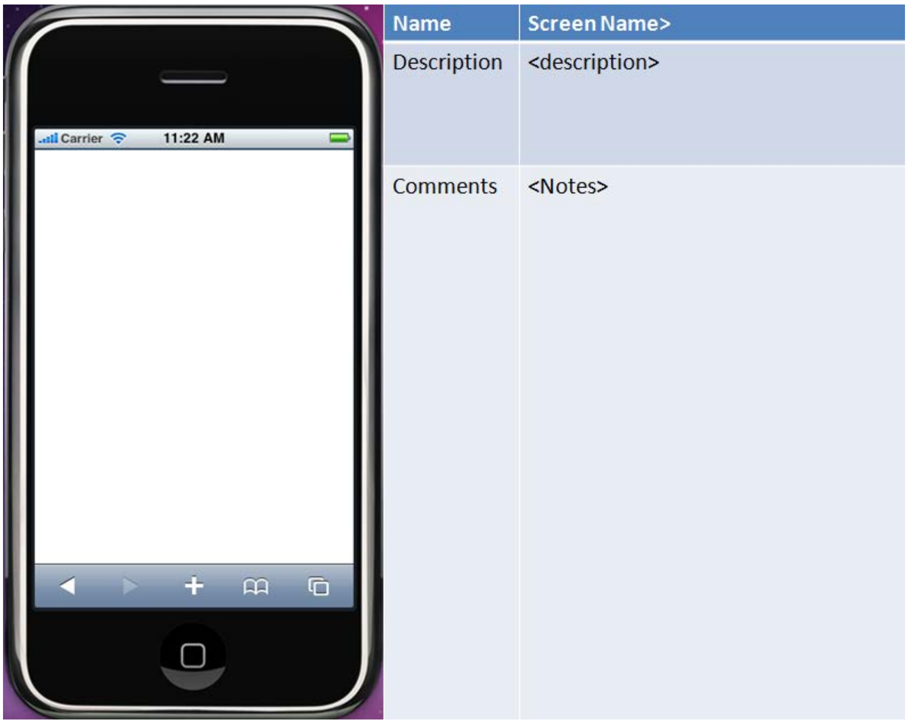
When designing and implementing a real application you should ***never use objects from the AXESDEMO library***. Any objects that you intend to use in your application should be copied from source file AXESDEMO/QTUTORIAL and then recompiled with a different object name in a different library. When you change the name of objects you may have to make equivalent changes to copied source code references.

Using the Supplied MS-PowerPoint Set to Draft an Initial Design

It is very important that you draft your smart phone application design before you write a single line of code to implement it or physically begin to design any screens with software.

This process is called **design drafting**.

Usually a draft design is reviewed several times with the project stakeholders before you begin to implement it in any form of software.



Name	Screen Name>
Description	<description>
Comments	<Notes>

A straightforward and effective way to draft out your proposed application design is by using something like the associated MS-PowerPoint (PPT) set. Alternatively you can possibly buy draft design pads like this from your local computer bookstore.

Copy the shipped ***Extension_Tutorial 12 - Smart Phone Applications.ppt*** file from the ***/docs*** folder on your aXes server to your PC and then open it. The shipped PPT has the following structure:

- An aXes tutorial identification page - which you can immediately remove.
- A title page listing the project name, objective, dates of update and versions.
- A screen summary page that summarizes all the screens in your design
- Multiple detailed screen layouts – one for each screen in your design
- A control and image page that contains images you use to produce your design. The shipped version contains some basic details – but you would certainly add to these as your design progresses.

You sketch out your proposed screen designs and make notes about the usage of the screen using one PPT page per screen. Once the design sketch has been reviewed and approved by the project stakeholders, you can begin to implement, possibly by producing a working prototype. The PPTs may act as a simple guiding specification summary – which you might choose to extend either in the PPT set by adding additional pages or by some other means such as MS-Word documents.

aXes® Documentation Library

eXtensions Tutorial 13 - Utilities

Prerequisites for Completing this Tutorial

To follow this tutorial you must have:

3. An existing projects

Overview and Objectives

The following material provides backup and restore functionality for an existing project.

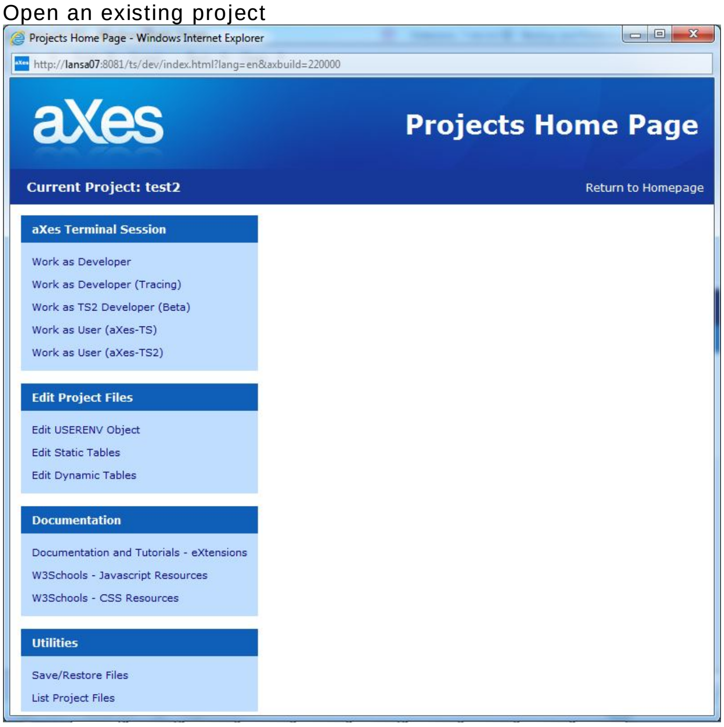
The objective of this tutorial is to guide you to create a backup of an existing project, download the backup file for safe keeping, and restore the backup and view a list of files in a project.

Backup and Restore

Getting Started

You need to have an existing project

1.) Open an existing project



Projects Home Page - Windows Internet Explorer

http://lansa07:8081/ts/dev/index.html?lang=en&axbuild=220000

Projects Home Page

Current Project: test2

Return to Homepage

aXes Terminal Session

- Work as Developer
- Work as Developer (Tracing)
- Work as TS2 Developer (Beta)
- Work as User (aXes-TS)
- Work as User (aXes-TS2)

Edit Project Files

- Edit USERENV Object
- Edit Static Tables
- Edit Dynamic Tables

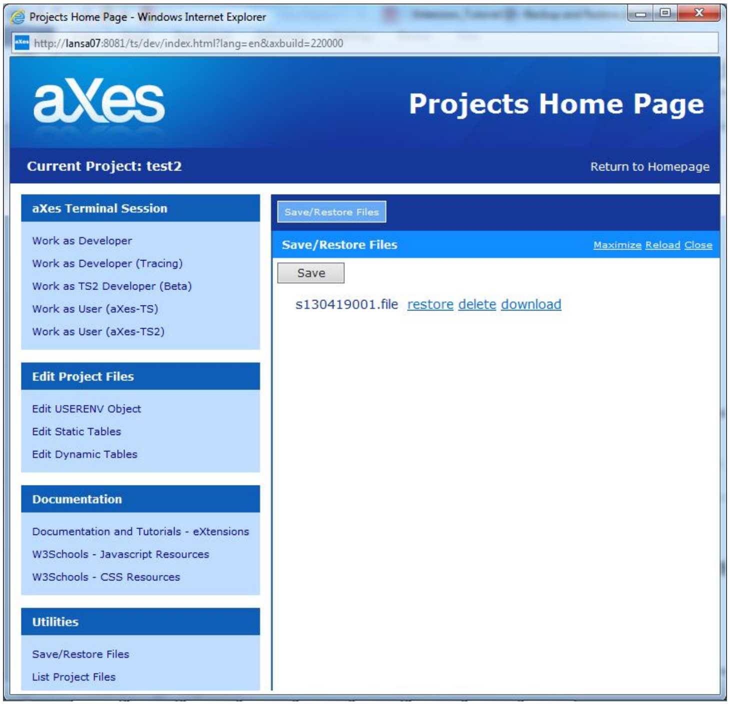
Documentation

- Documentation and Tutorials - eXtensions
- W3Schools - Javascript Resources
- W3Schools - CSS Resources

Utilities

- Save/Restore Files
- List Project Files

2.) Click on the Utilities section -> Save/Restore Files



Projects Home Page - Windows Internet Explorer

http://lansa07:8081/ts/dev/index.html?lang=en&axbuild=220000

Projects Home Page

Current Project: test2

Return to Homepage

aXes Terminal Session

- Work as Developer
- Work as Developer (Tracing)
- Work as TS2 Developer (Beta)
- Work as User (aXes-TS)
- Work as User (aXes-TS2)

Edit Project Files

- Edit USERENV Object
- Edit Static Tables
- Edit Dynamic Tables

Documentation

- Documentation and Tutorials - eXtensions
- W3Schools - Javascript Resources
- W3Schools - CSS Resources

Utilities

- Save/Restore Files
- List Project Files

Save/Restore Files

Save/Restore Files Maximize Reload Close

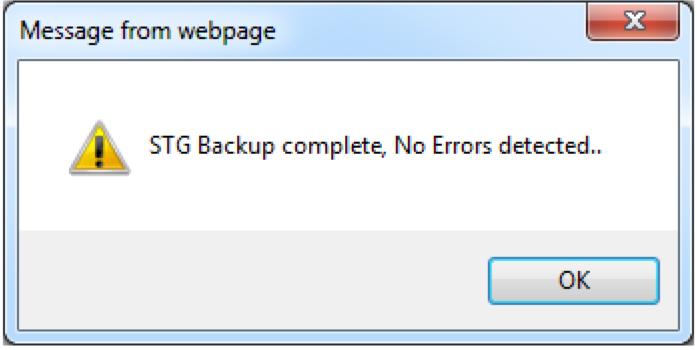
Save

s130419001.file restore delete download

Backup

Create a backup

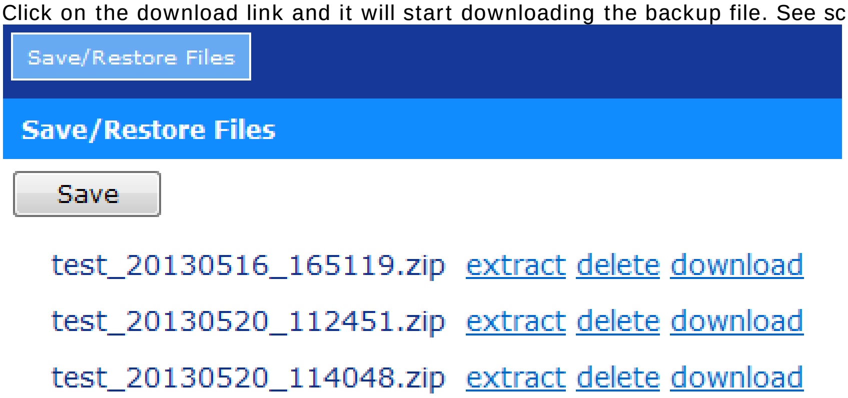
To get started on this tutorial, log on to aXes as a developer and open an existing project you want to backup. Click the Save button, and a backup file will be created. A dialog showing the message "Backup complete, No errors detected" will be displayed on completion.



STG Backup complete, No Errors detected..

Downloading backup file

Once a backup file is created, a 'download' link will be displayed adjacent to the Project filename. Click on the download link and it will start downloading the backup file. See screenshot below.



test_20130516_165119.zip extract delete download

test_20130520_112451.zip extract delete download

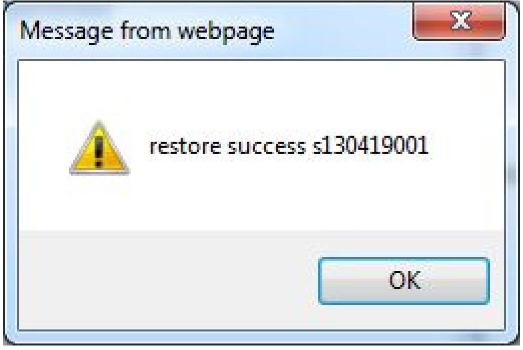
test_20130520_114048.zip extract delete download

Restore

To restore a backup

To Restore from backup, log on to aXes as a developer and open an existing project you wanted to restore a backup, make sure a backup file is already created. Click the Url link "restore" to start the restore process. A dialog showing a message "restore success syymmdd 000" pops up upon completion.

Note: (yymmdd000 indicate yy-year, mm-month, dd-day and 3 zero indicate count).



restore success s130419001

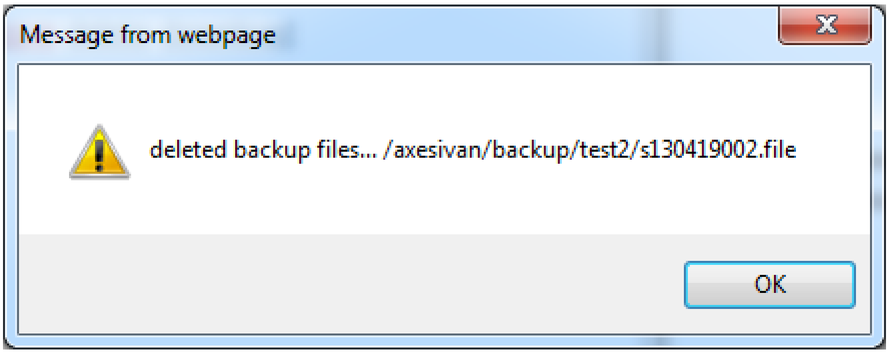
Warning and Disclaimer:
When restoring a backup file, please ensure you restore from the correct backup file of the specific Project. A rollback or recovery procedure does not exist.

Delete

To delete a backup

To delete a backup, log on to aXes as a developer and open an existing project for the backup you wish to delete. Clicking on the Url link “delete” will delete a backup file. A dialog showing the message “deleted backup files /axes/backup/<project>/syymmdd000.file” will be displayed.

Click on the Reload link to refresh the screen.



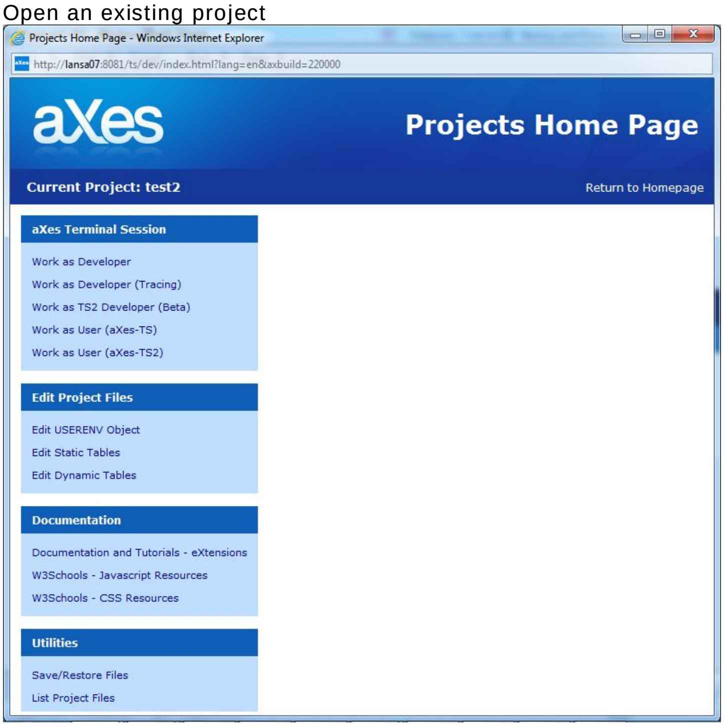
deleted backup files... /axesivan/backup/test2/s130419002.file

List Project files

Getting Started

You need to have an existing project

1.) Open an existing project



Projects Home Page - Windows Internet Explorer

http://lansa07:8081/ts/dev/index.html?lang=en&axbuild=220000

Projects Home Page

Current Project: test2

Return to Homepage

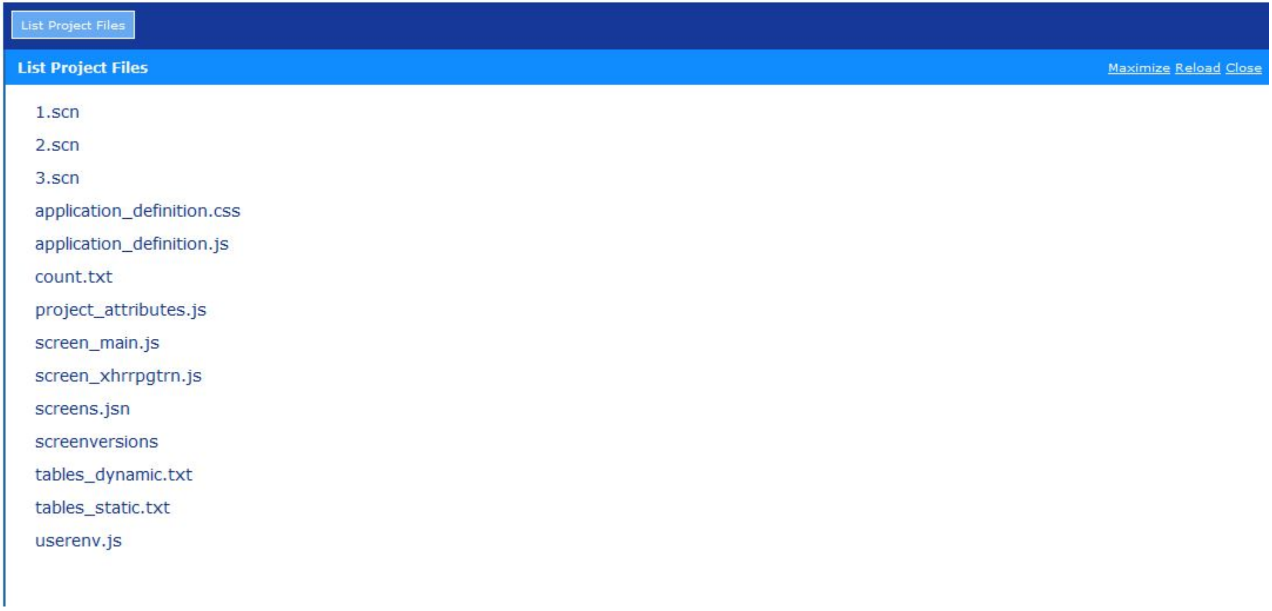
aXes Terminal Session
- Work as Developer
- Work as Developer (Tracing)
- Work as TS2 Developer (Beta)
- Work as User (aXes-TS)
- Work as User (aXes-TS2)

Edit Project Files
- Edit USERENV Object
- Edit Static Tables
- Edit Dynamic Tables

Documentation
- Documentation and Tutorials - eXtensions
- W3Schools - Javascript Resources
- W3Schools - CSS Resources

Utilities
- Save/Restore Files
- List Project Files

2.) Click on the Utilities section -> List Project Files, A Project List will be displayed.



List Project Files

List Project Files

Maximize Reload Close

1.scn
2.scn
3.scn
application_definition.css
application_definition.js
count.txt
project_attributes.js
screen_main.js
screen_xhrrpgtrn.js
screens.jsn
screenversions
tables_dynamic.txt
tables_static.txt
userenv.js

Right side shows all project files.

Documentation Library

eXtensions Tutorial 14 - TS2 Developer Tools

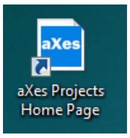
Basic Screen Enhancement

You must complete Tutorial 1 and 2 first

TS1 Developer tool is assumed knowledge for this tutorial.
If you have not completed related to TS1 developer tools please do so before attempting this tutorial.

Using the aXes Designer Window

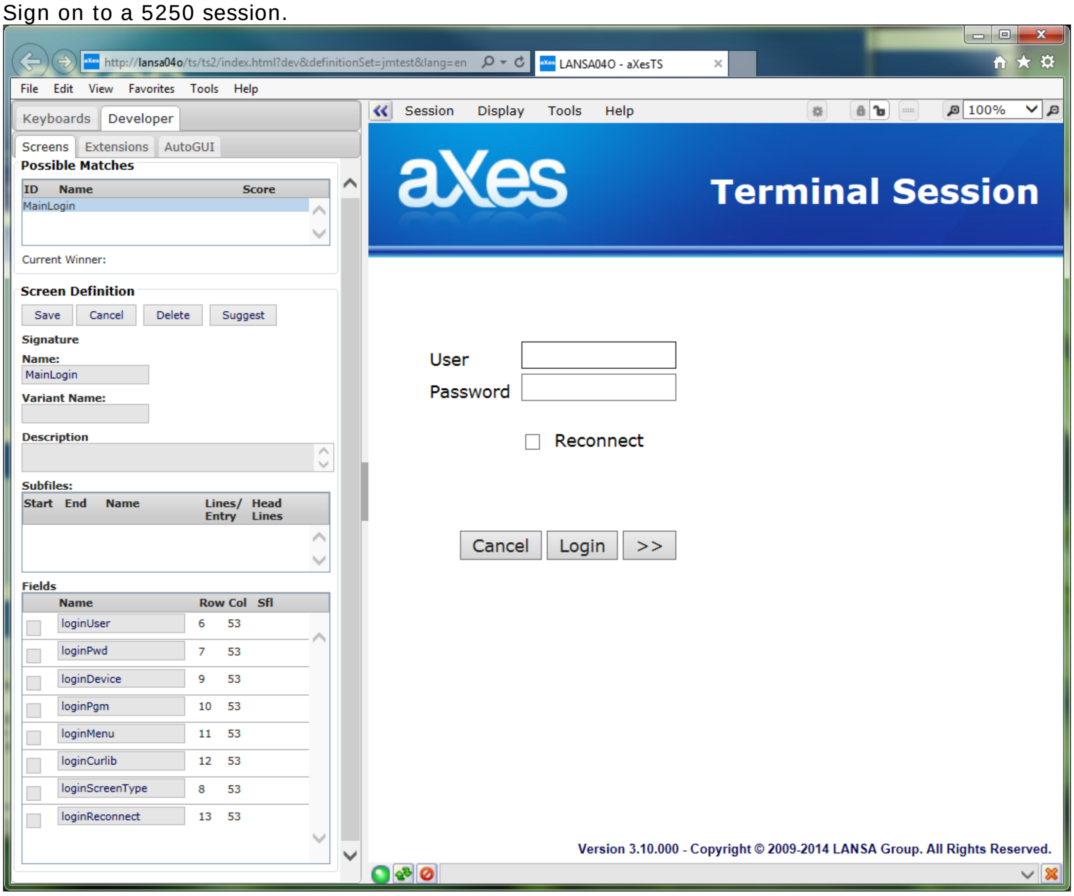
Using your desktop short cut open the aXes *Projects Home Page.*



Sign on as an aXes developer and select your project from the list on the right.

Then use the *Work as TS2 Developer* option to start an aXes development session.

Sign on to a 5250 session.


http://lansa04o/ts/ts2/index.html?dev&definitionSet=jmtest&lang=en
LANSA04O - aXesTS
File Edit View Favorites Tools Help
Keyboards Developer
Screens Extensions AutoGUI
Session Display Tools Help
100%
aXes
Terminal Session
Save Cancel Delete Suggest
Cancel Login >>
Reconnect

Possible Matches

ID	Name	Score
	MainLogin	

Current Winner:

Screen Definition

Signature

Name: MainLogin

Variant Name:

Description

Subfiles:

Start	End	Name	Lines/ Entry	Head Lines

Fields

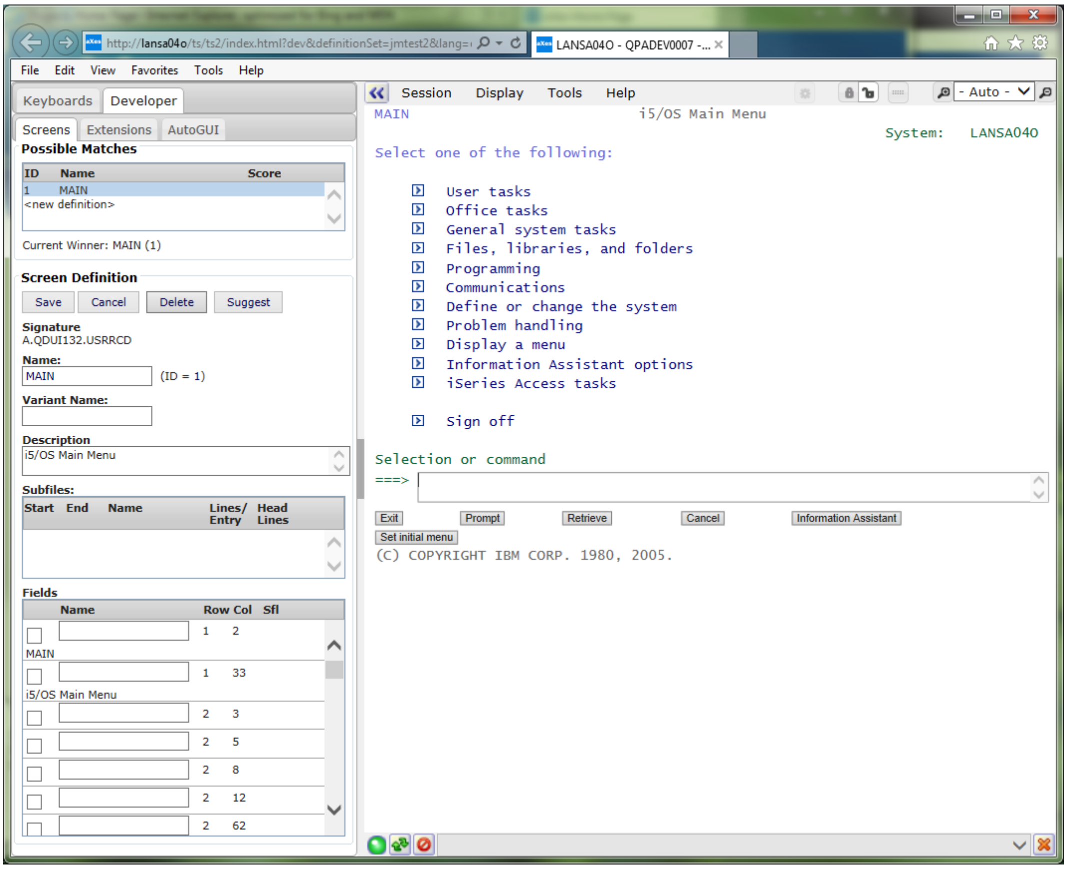
	Name	Row	Col	Sfl
□	loginUser	6	53	
□	loginPwd	7	53	
□	loginDevice	9	53	
□	loginPgm	10	53	
□	loginMenu	11	53	
□	loginCurlib	12	53	
□	loginScreenType	8	53	
□	loginReconnect	13	53	

User
Password

□ Reconnect

Version 3.10.000 - Copyright © 2009-2014 LANSA Group. All Rights Reserved.

Display the System I Main Menu.



http://lansa04o/ts/ts2/index.html?dev&definitionSet=jmtest2&lang=
LANSA04O - QPADEV0007 -...
File Edit View Favorites Tools Help
Keyboards Developer
Screens Extensions AutoGUI
Session Display Tools Help
- Auto -

Possible Matches

ID	Name	Score
1	MAIN	
	<new definition>	

Current Winner: MAIN (1)

Screen Definition

Save Cancel Delete Suggest

Signature
A.QDUI132.USRRCD

Name: MAIN (ID = 1)

Variant Name:

Description
i5/OS Main Menu

Subfiles:

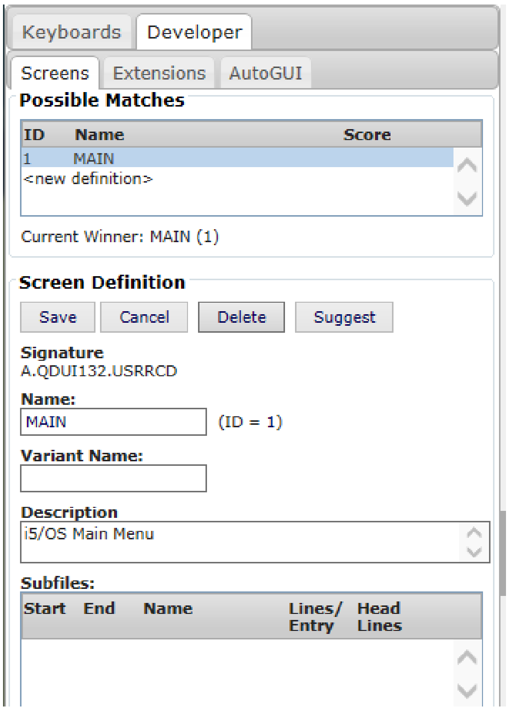
Start	End	Name	Lines/ Entry	Head Lines

Fields

	Name	Row	Col	Sfl
□ MAIN		1	2	
□ i5/OS Main Menu		1	33	
□		2	3	
□		2	5	
□		2	8	
□		2	12	
□		2	62	

```
MAIN                              i5/OS Main Menu
                                                      System:   LANSA040
Select one of the following:

     User tasks
     Office tasks
     General system tasks
     Files, libraries, and folders
     Programming
     Communications
     Define or change the system
     Problem handling
     Display a menu
     Information Assistant options
     iSeries Access tasks

     Sign off

Selection or command
===>

Exit   Prompt   Retrieve   Cancel   Information Assistant
Set initial menu
(C) COPYRIGHT IBM CORP. 1980, 2005.
```

Check that the aXes *Screens* tab shows the screen name MAIN, which you assigned in tutorial 1(ts1 development):



Possible Matches

ID	Name	Score
1	MAIN	
	<new definition>	

Current Winner: MAIN (1)

Screen Definition

Save Cancel Delete Suggest

Signature
A.QDUI132.USRRCD

Name:
MAIN (ID = 1)

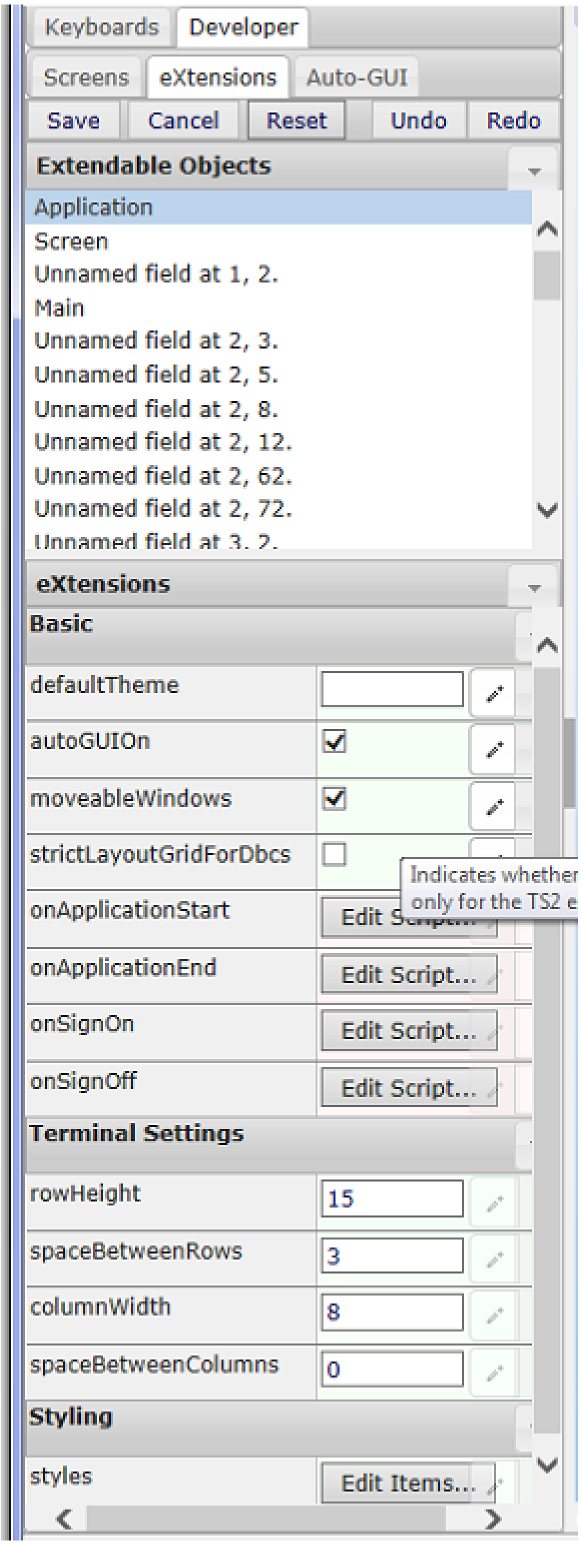
Variant Name:

Description
i5/OS Main Menu

Subfiles:

Start	End	Name	Lines/ Entry	Head Lines

Now click on the eXtensions Tab and select application. It should look like this:



Extendable Objects

- Application
- Screen
- Unnamed field at 1, 2.
- Main
- Unnamed field at 2, 3.
- Unnamed field at 2, 5.
- Unnamed field at 2, 8.
- Unnamed field at 2, 12.
- Unnamed field at 2, 62.
- Unnamed field at 2, 72.
- Unnamed field at 3, 2.

eXtensions

Basic

defaultTheme	
autoGUIOn	☑
moveableWindows	☑
strictLayoutGridForDbcs	☐
onApplicationStart	Edit Script...
onApplicationEnd	Edit Script...
onSignOn	Edit Script...
onSignOff	Edit Script...

Indicates whether
only for the TS2 e

Terminal Settings

rowHeight	15
spaceBetweenRows	3
columnWidth	8
spaceBetweenColumns	0

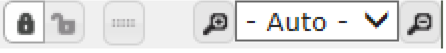
Styling

styles	Edit Items...

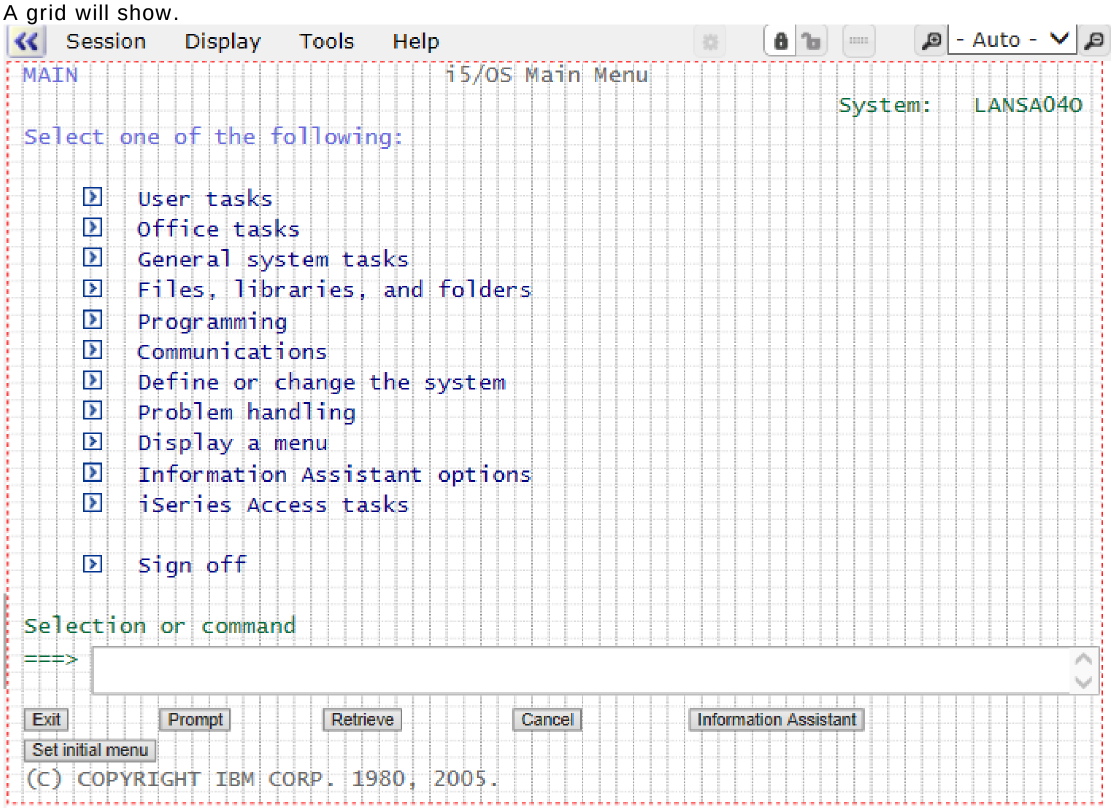
Screen Mode

TS2 developer tools have 2 screen mode that can be switched between designer screen mode and normal screen mode. When customizing a screen it is recommended to switch to designer screen mode.

Designer Screen mode (lock screen)

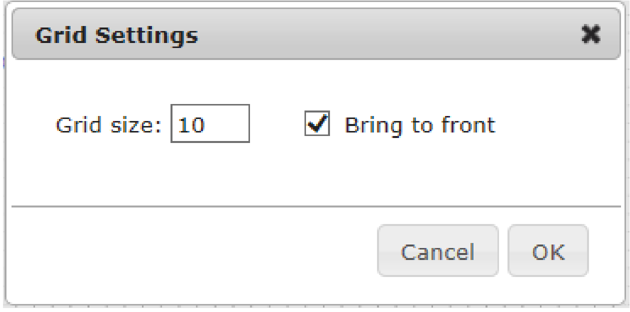


A grid will show.



```
MAIN                                 i5/OS Main Menu
                                                          System:   LANSA040
Select one of the following:

     User tasks
     Office tasks
     General system tasks
     Files, libraries, and folders
     Programming
     Communications
     Define or change the system
     Problem handling
     Display a menu
     Information Assistant options
     iSeries Access tasks

     Sign off

Selection or command
===>

Exit   Prompt   Retrieve   Cancel   Information Assistant
Set initial menu
(C) COPYRIGHT IBM CORP. 1980, 2005.
```

Adjusting Grid click this button and a dialog box grid setting will pop up.



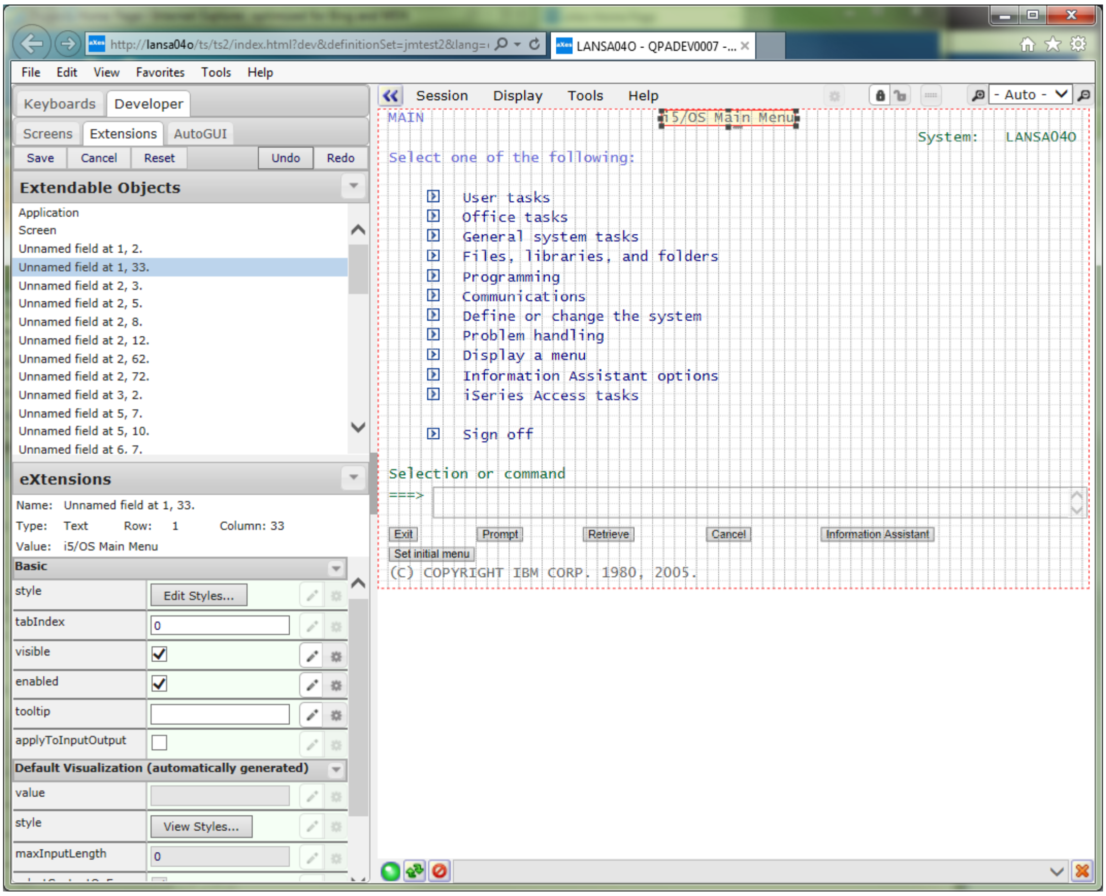
Grid Settings

Grid size: 10   ☑ Bring to front

Screen Customization

To start customizing the screen you can choose on customizing automatically generated field or add a user field.

Customize automatically generated field.



http://lansa04o/ts/ts2/index.html?dev&definitionSet=jmtest2&lang=
LANSA04O - QPADEV0007 -...
File Edit View Favorites Tools Help
Keyboards Developer
Screens Extensions AutoGUI
Save Cancel Reset Undo Redo
Session Display Tools Help
- Auto -

Extendable Objects

- Application
- Screen
- Unnamed field at 1, 2.
- Unnamed field at 1, 33.
- Unnamed field at 2, 3.
- Unnamed field at 2, 5.
- Unnamed field at 2, 8.
- Unnamed field at 2, 12.
- Unnamed field at 2, 62.
- Unnamed field at 2, 72.
- Unnamed field at 3, 2.
- Unnamed field at 5, 7.
- Unnamed field at 5, 10.
- Unnamed field at 6, 7.

eXtensions

Name: Unnamed field at 1, 33.
Type: Text Row: 1 Column: 33
Value: i5/OS Main Menu

Basic

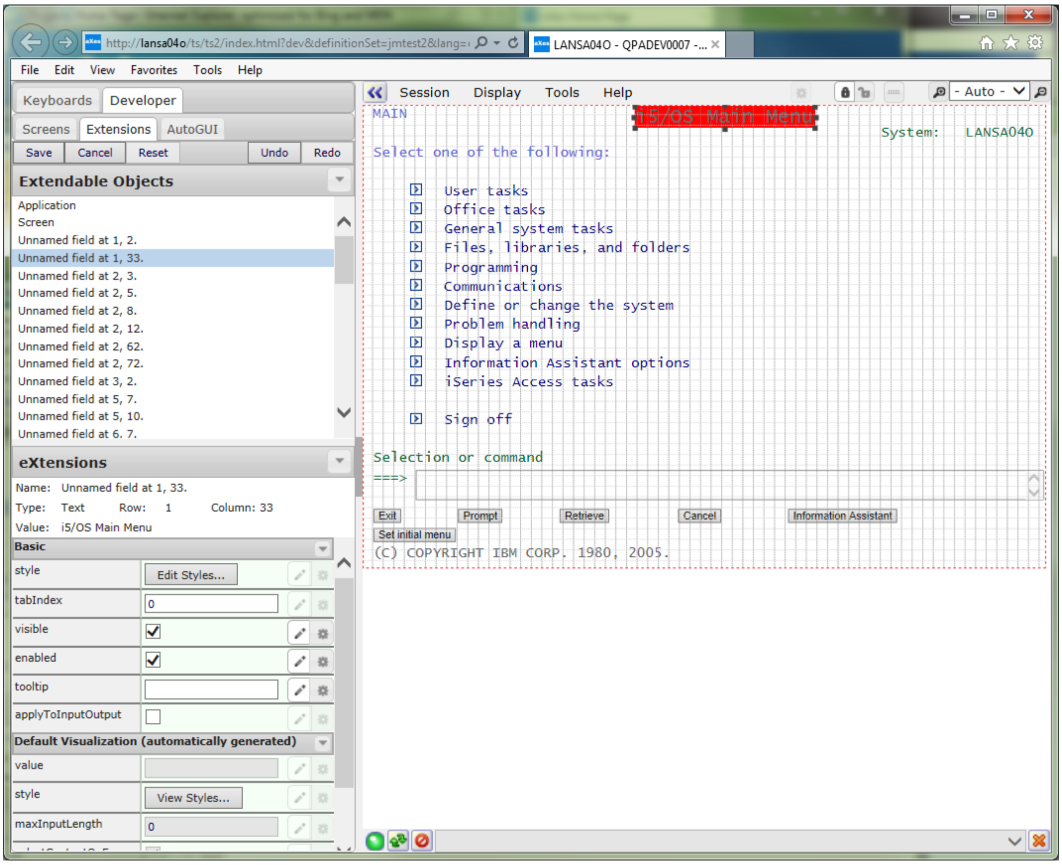
style	Edit Styles...
tabIndex	0
visible	☑
enabled	☑
tooltip	
applyToInputOutput	□

Default Visualization (automatically generated)

value	
style	View Styles...
maxInputLength	0

```
MAIN                          i5/OS Main Menu
                                                          System:   LANSA040
Select one of the following:

     User tasks
     Office tasks
     General system tasks
     Files, libraries, and folders
     Programming
     Communications
     Define or change the system
     Problem handling
     Display a menu
     Information Assistant options
     iSeries Access tasks

     Sign off

Selection or command
===>

Exit      Prompt      Retrieve      Cancel      Information Assistant
Set initial menu
(C) COPYRIGHT IBM CORP. 1980, 2005.
```

1. Select any one of Unnamed field
2. Click the Customize Field menu items, properties of default visualization becomes enabled (view styles become Edit Styles.. button)
3. Click Edit style… button on the style properties of default visualization.
4. Then Edit Style dialog appear where on you can add or defines your custom styles.



Extendable Objects

- Application
- Screen
- Unnamed field at 1, 2.
- Unnamed field at 1, 33.
- Unnamed field at 2, 3.
- Unnamed field at 2, 5.
- Unnamed field at 2, 8.
- Unnamed field at 2, 12.
- Unnamed field at 2, 62.
- Unnamed field at 2, 72.
- Unnamed field at 3, 2.
- Unnamed field at 5, 7.
- Unnamed field at 5, 10.
- Unnamed field at 6, 7.

eXtensions

Name: Unnamed field at 1, 33.
Type: Text Row: 1 Column: 33
Value: i5/OS Main Menu

Basic

style	Edit Styles...
tabIndex	0
visible	☑
enabled	☑
tooltip	
applyToInputOutput	□

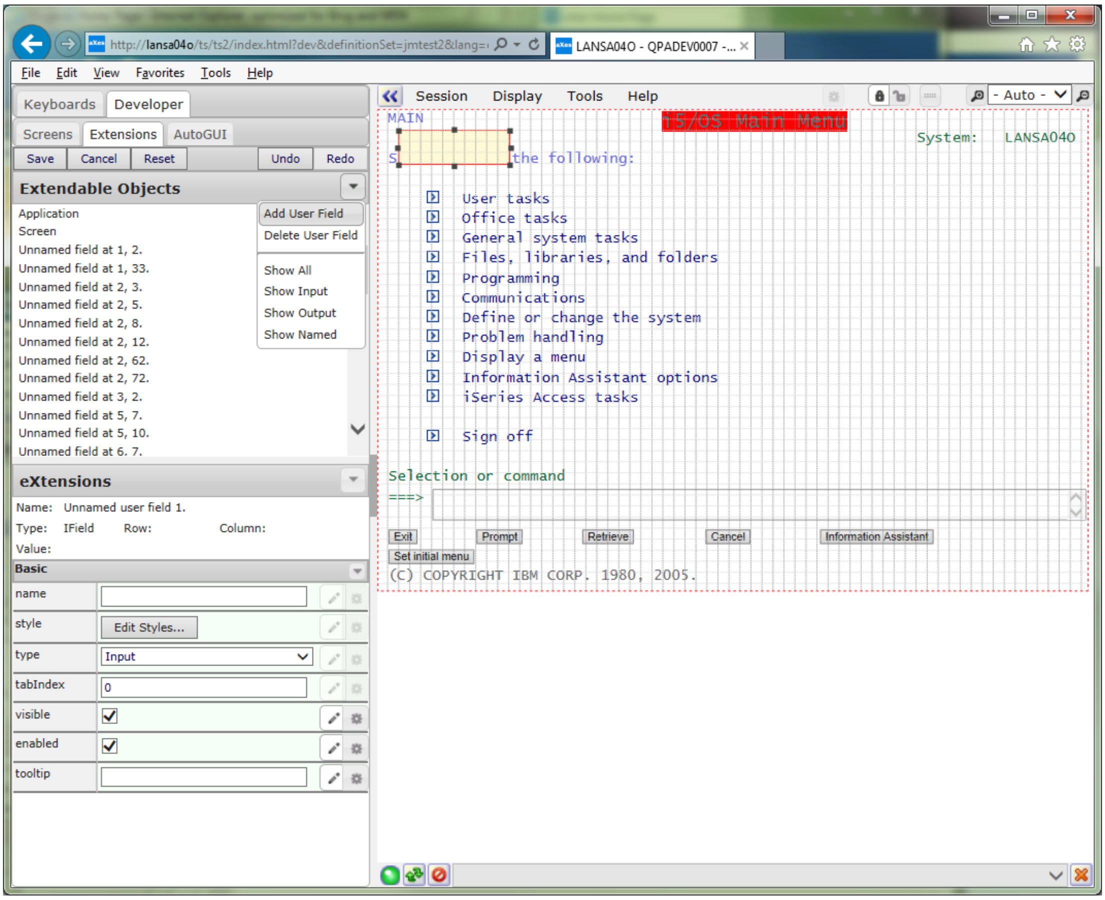
Default Visualization (automatically generated)

value	
style	View Styles...
maxInputLength	0

```
MAIN                            i5/OS Main Menu
                                                            System:   LANSA040
Select one of the following:

     User tasks
     Office tasks
     General system tasks
     Files, libraries, and folders
     Programming
     Communications
     Define or change the system
     Problem handling
     Display a menu
     Information Assistant options
     iSeries Access tasks

     Sign off

Selection or command
===>

(C) COPYRIGHT IBM CORP. 1980, 2005.
```

5. Click Ok, styles will be applied instantly.
6. Click Save Button on the upper left corner of Extension Tab to save your customization.

Customize a User field.



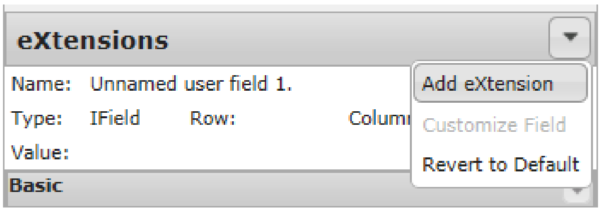
Application
Screen
Unnamed field at 1, 2.
Unnamed field at 1, 33.
Unnamed field at 2, 3.
Unnamed field at 2, 5.
Unnamed field at 2, 8.
Unnamed field at 2, 12.
Unnamed field at 2, 62.
Unnamed field at 2, 72.
Unnamed field at 3, 2.
Unnamed field at 5, 7.
Unnamed field at 5, 10.
Unnamed field at 6, 7.

Name: Unnamed user field 1.
Type: IField Row: Column:
Value:

name	
style	Edit Styles...
type	Input
tabIndex	0
visible	☑
enabled	☑
tooltip	

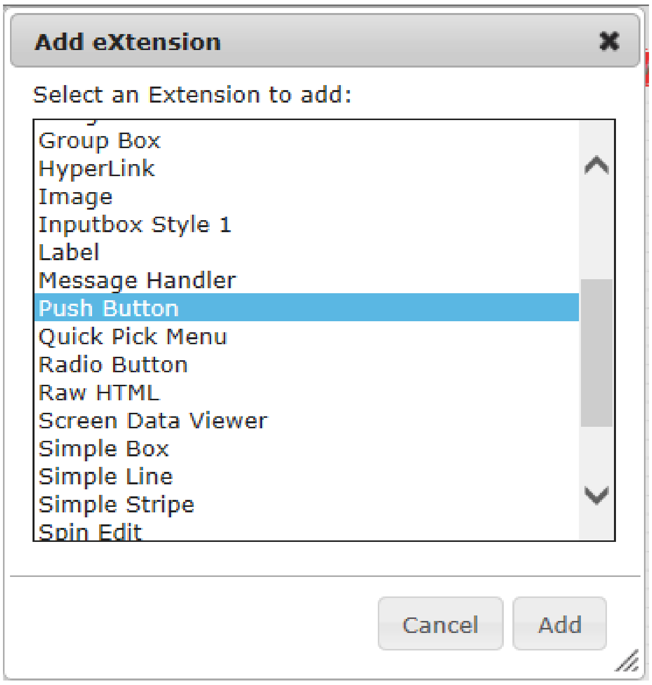
```
MAIN                          i5/OS Main Menu
                                                          System:   LANSA040
S           the following:

    User tasks
    Office tasks
    General system tasks
    Files, libraries, and folders
    Programming
    Communications
    Define or change the system
    Problem handling
    Display a menu
    Information Assistant options
    iSeries Access tasks

    Sign off

Selection or command
===>

(C) COPYRIGHT IBM CORP. 1980, 2005.
```

1. Click Add User Field menu item
2. Drag the User Field across the desired position of the screen
3. Click Add Extension menu items



Name: Unnamed user field 1.
Type: IField Row: Column
Value:

4. An add eXtension Dialog box will show up.

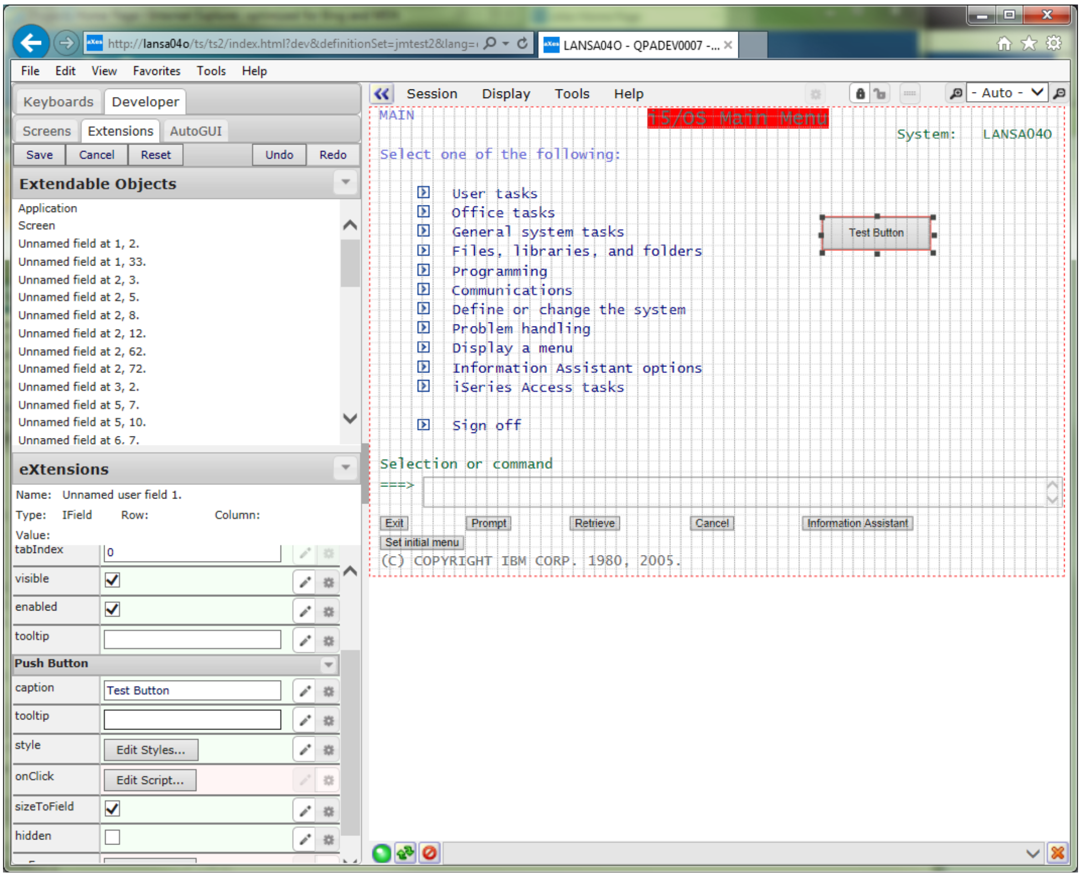


Add eXtension

Select an Extension to add:

Group Box
HyperLink
Image
Inputbox Style 1
Label
Message Handler
Push Button
Quick Pick Menu
Radio Button
Raw HTML
Screen Data Viewer
Simple Box
Simple Line
Simple Stripe
Spin Edit

5. Select Push Button and click Add.
6. A push button extension will be added to user field.
7. Then you can fill up its properties, like caption, styles and events.
8. Click Save Button on the upper left corner of Extension Tab to save your customization.



http://lansa04o/ts/ts2/index.html?dev&definitionSet=jmtest2&lang=
LANSA040 - QPADEV0007 -...
File Edit View Favorites Tools Help
Keyboards Developer
Screens Extensions AutoGUI
Save Cancel Reset Undo Redo
Session Display Tools Help
- Auto -

Extendable Objects

Application
Screen
Unnamed field at 1, 2.
Unnamed field at 1, 33.
Unnamed field at 2, 3.
Unnamed field at 2, 5.
Unnamed field at 2, 8.
Unnamed field at 2, 12.
Unnamed field at 2, 62.
Unnamed field at 2, 72.
Unnamed field at 3, 2.
Unnamed field at 5, 7.
Unnamed field at 5, 10.
Unnamed field at 6, 7.

eXtensions

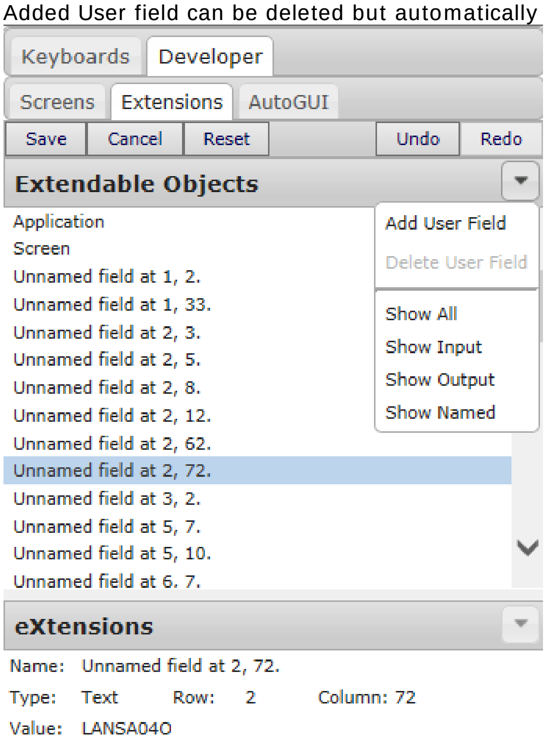
Name: Unnamed user field 1.
Type: IField Row: Column:
Value:

tabIndex	0
visible	☑
enabled	☑
tooltip	
Push Button	
caption	Test Button
tooltip	
style	Edit Styles...
onClick	Edit Script...
sizeToField	☑
hidden	□

```
MAIN                           i5/OS Main Menu
                                                   System:   LANSA040
Select one of the following:

     User tasks
     Office tasks
     General system tasks
     Files, libraries, and folders
     Programming
     Communications
     Define or change the system
     Problem handling
     Display a menu
     Information Assistant options
     iSeries Access tasks

     Sign off

Selection or command
===>

(C) COPYRIGHT IBM CORP. 1980, 2005.
```

Deleting User Field

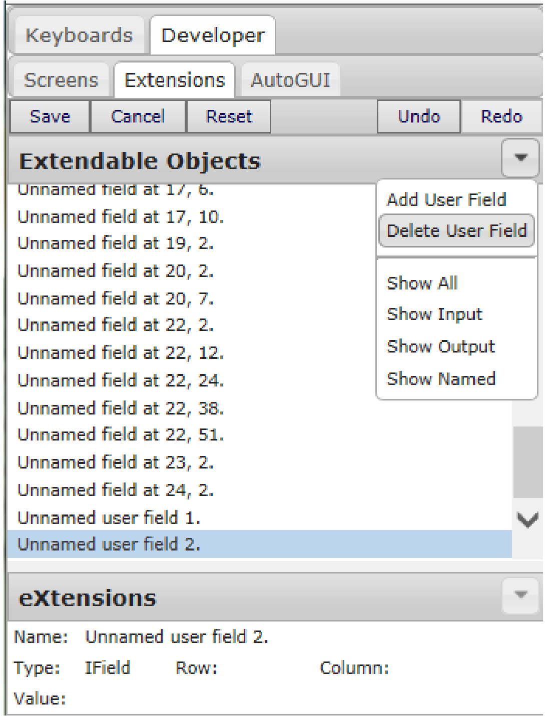
Added User field can be deleted but automatically generated field is not allowed.



Application
Screen
Unnamed field at 1, 2.
Unnamed field at 1, 33.
Unnamed field at 2, 3.
Unnamed field at 2, 5.
Unnamed field at 2, 8.
Unnamed field at 2, 12.
Unnamed field at 2, 62.
Unnamed field at 2, 72.
Unnamed field at 3, 2.
Unnamed field at 5, 7.
Unnamed field at 5, 10.
Unnamed field at 6, 7.

Name: Unnamed field at 2, 72.
Type: Text Row: 2 Column: 72
Value: LANSA040

Select a user field and the delete user field option is enabled.



Unnamed field at 17, 6.
Unnamed field at 17, 10.
Unnamed field at 19, 2.
Unnamed field at 20, 2.
Unnamed field at 20, 7.
Unnamed field at 22, 2.
Unnamed field at 22, 12.
Unnamed field at 22, 24.
Unnamed field at 22, 38.
Unnamed field at 22, 51.
Unnamed field at 23, 2.
Unnamed field at 24, 2.
Unnamed user field 1.
Unnamed user field 2.

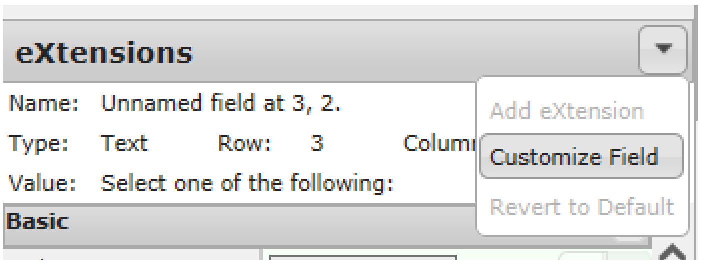
Name: Unnamed user field 2.
Type: IField Row: Column:
Value:

Select a automatically generated field and the delete user field options is disabled.

Field Customization

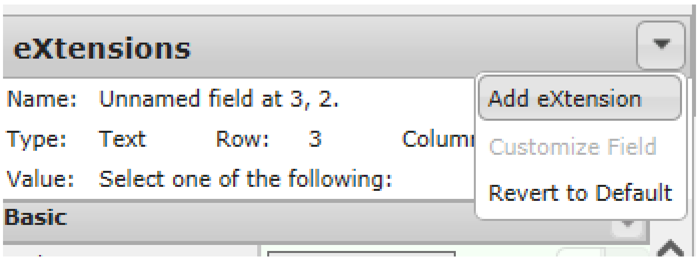
Substitute extension of automatically generated Field or replacing default visualization with other extensions.

1. Select the field and Click Customize field



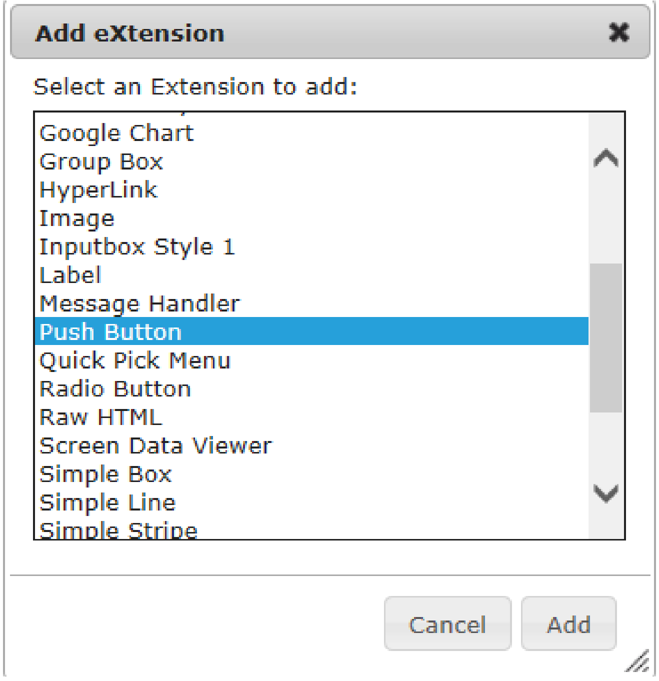
Name: Unnamed field at 3, 2.
Type: Text Row: 3 Colum
Value: Select one of the following:

2. Click then Add eXtension

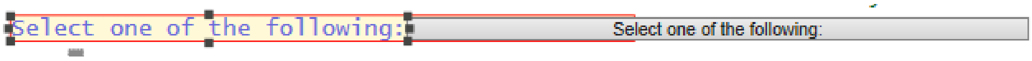


Name: Unnamed field at 3, 2.
Type: Text Row: 3 Colum
Value: Select one of the following:

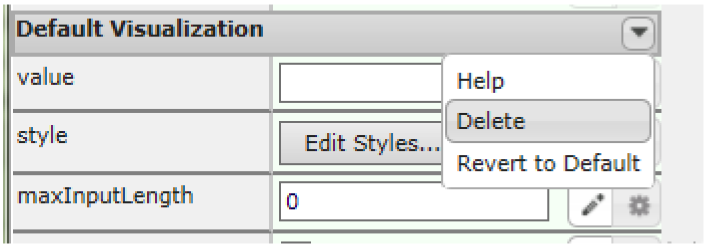
3. Select push button from the eXtension dialog box and Click Add



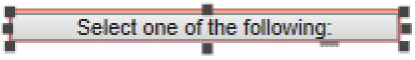
4. A Button extension will be added to the field



5. Then click Delete menu items to delete the existing default visualization

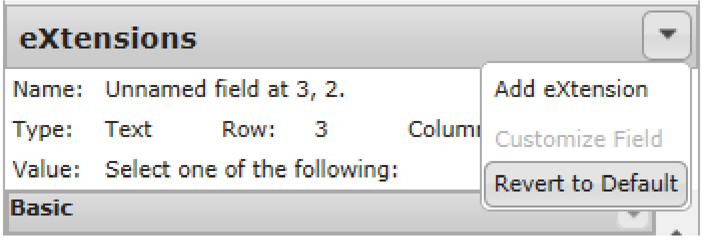


6. The results is only the button extension remain on the field

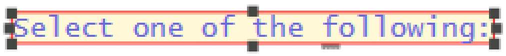


Revert to default visualization

1. Click Revert to Default menu items then it will change back to default visualization

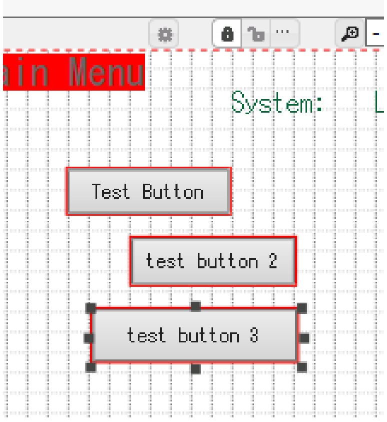


2. This is the results.

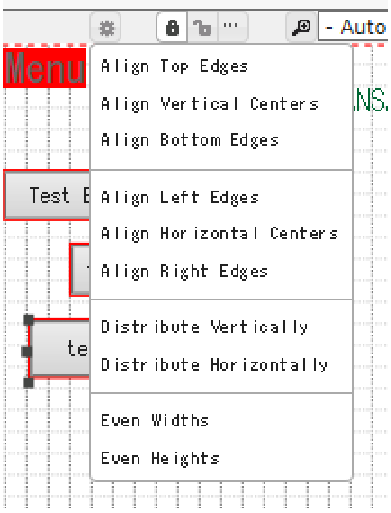


Alignment Tools

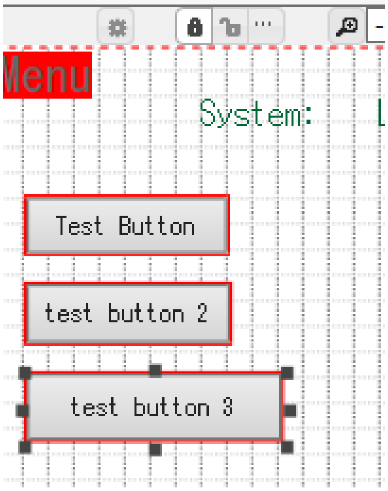
1. Selecting multiple extension (field) items will enable the alignment button.



2. Then you can choose from the alignment menu items which function do want to perform.



3. Click Align left Edges, will align all the selected extension (field) items to the left.



Advanced Tutorials

Advanced aXes tutorials are not shipped with the product. To get a copy of an advanced tutorial, please contact your aXes vendor.

These advanced tutorials are available:

Topic	Content
External Hosting	Shows you how to imbed aXes-TS applications inside other web pages, portals or .NET applications.
Extended AutoGUI Techniques	Shows you how to use scripting to logically extend the aXes AutoGUI capabilities of aXes-TS.
Virtual Screens	Shows you how to use "virtual" 5250 screens to logically extend and enhance an existing 5250 application by introducing brand new screens that do not exist in the real 5250 application.
Directing 5250 screen access	Shows you how to initiate and automatically navigate aXes 5250 sessions from URLs sent to users in e-mails, SMS messages, etc. In effect how to use hyperlinks to support event- or action-based applications.
The Robot API	Show you how the aXes-Robot API may be used to programmatically extract or input data to/from 5250 screens.