Documentation Library

eXtensions Guide

Scripting Reference, Help Text and Problem Isolation Techniques

Scripting Reference

Important Warning
About This Document
aXes Scripting Basics
eXtension Scripting Global Functions & Objects
ELEMENT Object Members
TABLEMANAGER Object Members
TABLE Object Members
AXes Object Functions and Features
General JavaScript Resources

Important Warning

Important Warning

You should only use methods, objects and properties that are described in this documentation.

You are not permitted to use any scripting features that are not mentioned in this documentation.

If you ignore this warning then you are implicitly accepting responsibility for the risk of all of the following:

- The introduction into your application of incompatibilities with future versions of aXes.
- Voiding or limitation of any maintenance or support contract you may have in place for aXes.
- Being charged for problem resolutions that are traced back to being caused by such non-permitted usage.

These methods, objects and properties are usable in the AXES object:

Name	Type	Environment	Description
isTS2Engine	Boolean	TS and TS2	Indicates if the aXes TS2 rendering engine is currently be
displayMessage()	Function	TS2 only	AXES API that allows user defined messages to be added
isDeveloper()	Function	TS and TS2	Axes API that indicates whether that the aXes 5250 sessio development mode.

About This Document

This purpose of this document is to provide a complete reference of all methods, objects and properties available in the eXtension scripting library.
Any methods, objects and properties that are not mentioned in this documentation are not part of the eXtension scripting library and should not be used in your scripts.

Important Warning

You should only use methods, objects and properties that are described in this documentation.
You are not permitted to use any scripting features that are not mentioned in this documentation.
If you ignore this warning then you are implicitly accepting responsibility for the risk of all of the following:

1. The introduction into your application of incompatibilities with future versions of aXes.
2. Voiding or limitation of any maintenance or support contract you may have in place for aXes.
3. Being charged for problem resolutions that are traced back to being caused by such non-permitted usage.

aXes Scripting Basics

What scripting language does aXes support?

All scripting in aXes is done in JavaScript. Please note that aXes does not have its own scripting engine - it uses the current browser's scripting engine. It is therefore your responsibility to test that the scripts you write are running on your targeted browser version.

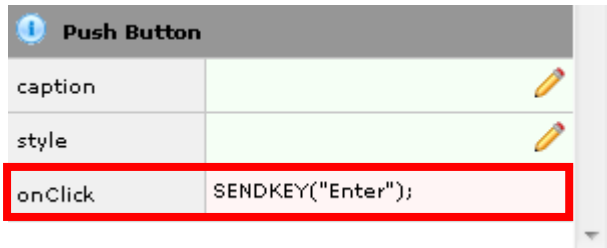
Inserting Your Scripts into aXes

There are 2 places where you can put your scripts:

- Various eXtension's properties / events
- USERENV.js

Scripting in eXtension's Properties / Events

The most common way of scripting is by typing scripts into an eXtension's property in the property sheet. Below is an example of a script that sends an enter key to the server when a user clicks a button.

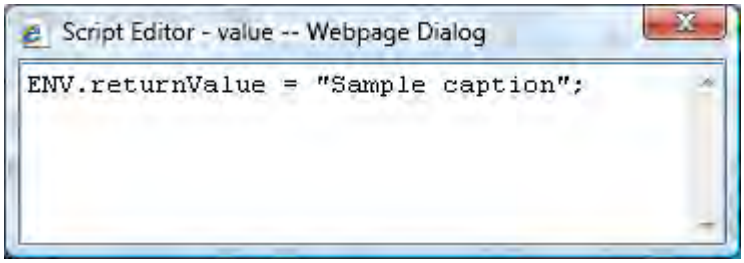


The objects available in the eXtension scripting library will be covered in detail in the next few sections.

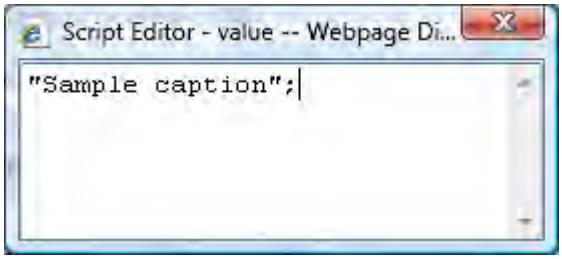
One special object worth mentioning here is the ENV object. It is an object that represents the current eXtension's context. Most features in ENV object are available to any scripts, however some features are specific to the event or property the script is for.

When you are writing a script for a property, the script needs to yield a value that will be used as the value of the property. The script can do this in two ways:

- Setting the **ENV.returnValue** property (this is the preferred way).



- The last statement in the script yields the value.



Another common use of ENV is to give access to the eXtension scripting objects to an arbitrary function by passing the ENV object to the function as a parameter. This will be discussed in the next section.

Scripting in USERENV.js

USERENV.js is a special file that aXes automatically includes in its runtime environment. This file is **not** optional and it must exist in your aXes project folder.

aXes comes with a standard USERENV.js file that you can extend with your own scripts. This file is an ideal place to put your global functions - functions that you want to make available for use from any scripts defined elsewhere.

Note also that some predefined methods that come with the default USERENV.js file are used by aXes eXtensions, so care should be observed when removing existing methods from this file.

It is important to note that eXtension scripting library is not available for use from USERENV.js, as many of the objects and methods in the library are context-sensitive. Take as an example the property FIELD which returns a reference to the field object currently associated with the eXtension. It will not make any sense to just say FIELD in a global level as there is no way aXes could tell which eXtension is associated with FIELD.

The solution is to pass the context when calling functions defined in the USERENV.js, this way those functions will have access to the scripting library.

Supposing that we have a function in the USERENV.js that copies the content of a subfile column to your computer's clipboard. The first parameter to the function will be the context of the eXtension, which will give this function access to various eXtension features.

```
function copyColumnToClipboard(context, fieldName)
{
        // we now have access the eXtension scripting objects such as FIELD
        var field = context.FIELD;
        . . .
}
```

Assuming that we use a button to trigger execution of the `copyColumnToClipboard`, we need to put the following code in the ***onClick*** event of the button:

```
copyColumnToClipboard(ENV, "employeeNo");
```

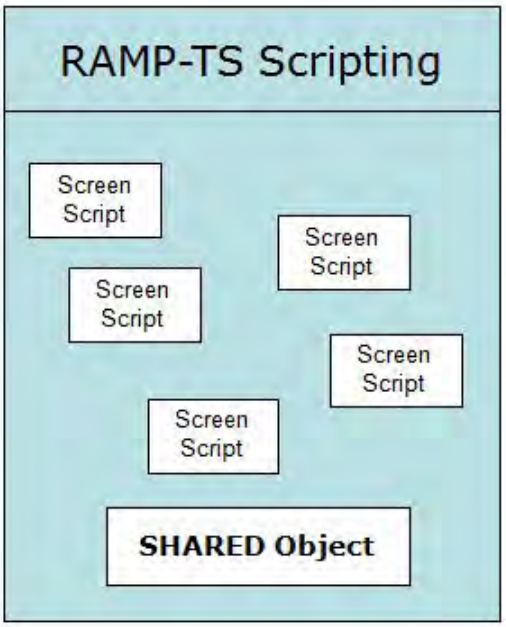
The tutorial 5 of aXes eXtensions tutorial series discusses USERENV.js in more details.

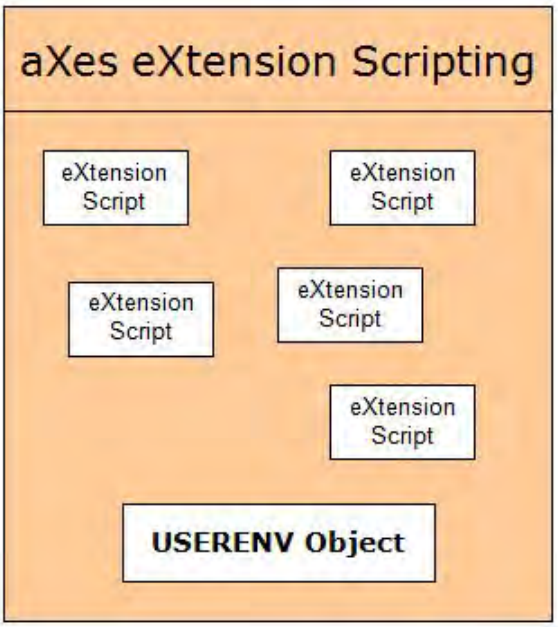
Cross-Scripting: Using RAMP-TS functions and properties in your aXes scripts and vice versa

(For RAMP-TS developers.)

In the RAMP-TS environment, functions can be shared by scripts by using the SHARED object.

In the aXes environment, functions can be shared by scripts by using the USERENV object.





aXes scripts can use RAMP-TS functions and properties, including the SHARED object, *if* the aXes script is being executed within a RAMP-TS application at runtime. But they must prefix the function name with "RAMP." , and they must be coded to handle the absence of the RAMP runtime environment.

```
if (RAMP != null)
{
   RAMP.ALERT_MESSAGE("my alert message");
}
```

or

```
if (RAMP != null)
{
   RAMP.SHARED.Test ;
}
```

Conversely, RAMP-TS scripts can always refer to aXes functions in USERENV

```
USERENV.asString(26);
```

Key points to remember:

- RAMP-TS screen navigation scripts and aXes eXtension scripts execute in different name spaces.
- The aXes eXtensions scripting environment is always available.
- The RAMP-TS scripting environment exists only when the RAMP-TS execution environment is running.
- The RAMP-TS scripting environment is not available when you execute an eXtension's script in Axes all by itself - or when executing it from RAMP-Tools.

Example: To execute RAMP functions in an Axes Script

Suppose you are writing an onClick script for a push button extension, and you want to be able to use the RAMP-TS function ALERT_MESSAGE("My message").

To use the function in an aXes script you must specify that it is a RAMP script. You do this by using the RAMP name space.

```
RAMP.ALERT_MESSAGE("My message");
```

This will work provided that the aXes script always runs in aXes within a RAMP-TS application. But what if it is sometimes run in a standalone aXes environment, or in the RAMP Tools window?

You handle this by checking for the existence of RAMP, before trying to use ALERT_MESSAGE.

```
if (RAMP != null)
{
   RAMP.ALERT_MESSAGE("My message");
}
else
{
   /* use some other method, or do nothing */
}
```

Similarly, if your aXes onClick routine wants to use a function that the RAMP-TS scripts share through the SHARED object, (held in UF_SY420_RTS.js),

```
if (RAMP != null)
{
   RAMP.SHARED.myFunction1("x", "y", "z");
}
else
{
   /* use some other method, or do nothing */
}
```

Example: To execute an aXes USERENV function in a RAMP Script

Suppose you are writing a RAMP-TS vHandle_ARRIVE routine, and want to use function asString that the aXes eXtension scripts all share in USERENV.

If using RAMP-TS, then the aXes scripting environment is always available, so you just write:

```
USERENV.asString(26);
```

A table of scripting options

Situation	In a RAMP-TS Script	In an eXtension script
Use RAMP-TS function ALERT_MESSAGE("Hello")	Use ALERT_MESSAGE("Hello") Or RAMP.ALERT_MESSAGE("Hello")	Must use RAMP.ALERT_MESSAGE("Hello") and cater for situations where RAMP is null.
Access SHARED property named Test.	Use SHARED.Test Or RAMP.SHARED.Test	Must use RAMP.SHARED.Test and cater for situations where RAMP is null.
Access USERENV property named Test	Use USERENV.Test	Use USERENV.Test

eXtension Scripting Global Functions & Objects

Summary of global functions & objects:

Name	Type	Description
ENV	Object	Represents the current execution context
FIELD	Object	Gets a reference to the element associated with the eXtension that this script is written for.
TABLEMANAGER	Object	Gets a reference to the eXtension table manager.
LANGUAGE	String	Gets the 2-character code representing the current language used (for example English is en, French is fr).
SCREENNAME	String	Gets the name of the current screen.
SCREENVARIANTNAME	String	Gets the variant name of the current screen.
FIELDS	Method	Get a reference to an element with the specified name and optionally the specified index (if the element is a subfile column).
TRACE	Method	Output a message to the trace window.
DEBUG	Method	Output a message to the debug window.
CTEXT	Method	Get the text resource with the specified key from the user-defined resource file (Texts_Cust_ll.txt). Note that if you change a Texts_Cust_ll.txt file you should clear your browser cache to pick up the new file version.
SET5250CURSORTO	Method	Set the **5250 logical cursor** to the specified field.
SET5250CURSORPOS	Method	Set the **5250 logical cursor** to the specified row and column.
SHOWAXESMENUBAR	Method	Show or hide aXes menu bar.
SHOWAXESSTATUSBAR	Method	Show or hide aXes status bar.
GETACTIVEELEMENT	Method	Gets reference to the element that currently has input focus.
GETFIELDATCURSOR	Method	Gets a reference to the field at the current 5250 cursor position (TS2 only)

ENV

ENV represents the current execution context of the script. Each eXtension has their own context. This means that the context of a script that you wrote for an ***onClick*** event in button A will be different from the context of a script that you wrote for an ***onClick*** event in button B.

ENV also acts as a big container that holds all the global functions & objects of the eXtension scripting library. This makes it possible to make the library accessible to functions defined anywhere by passing this ENV object to that function.

When we type in the following statement in the aXes script editor:

```
var empNoField = FIELDS("employeeNo");
```

It is equivalent to the following:

```
var empNoField = ENV.FIELDS("employeeNo");
```

The first way is the preferred way since it is clearer and more succinct.

As mentioned earlier, another common use of ENV is passing it to a function defined somewhere else such as in UserENV.js:

```
copyToClipboard(ENV, "empNoField");
```

Some eXtensions will also provide additional methods and objects in their context that are specific for the event or properties. A good example for this is the **onClick** event of the **Quick Pick** eXtension. This event is triggered when a user clicks on a menu item. Since most likely there are more than 1 menu item the eXtension needs to tell the script which menu item is clicked. The **Quick Pick** does this by exposing additional properties in the **ENV** object: ***itemNumber*** and ***itemCaption***.

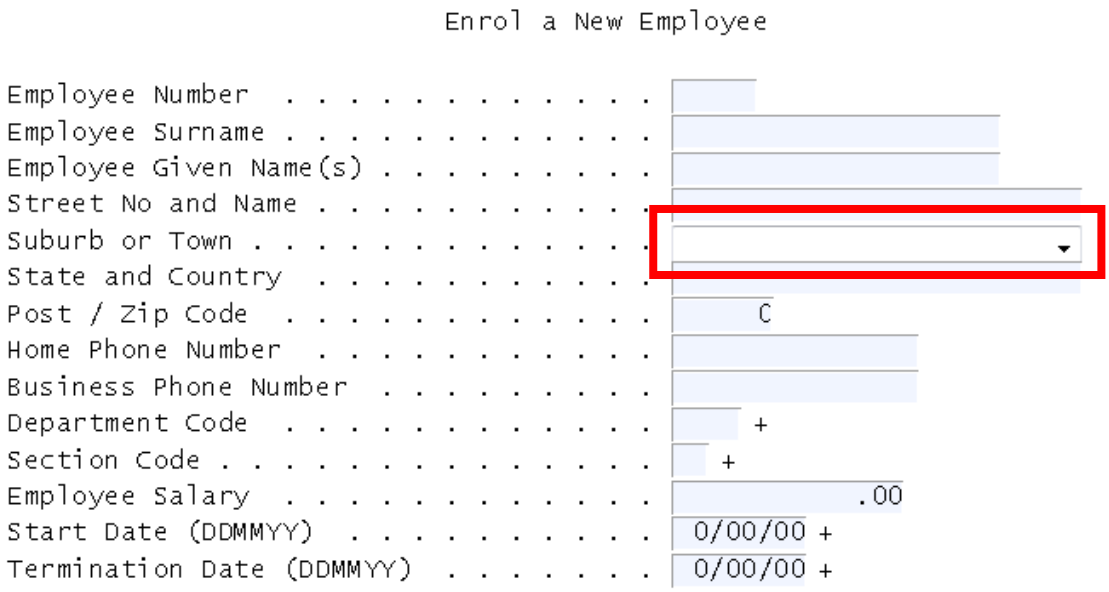
An example of how the script can access those additional properties:

```
var item = ENV.itemNumber;
var text = ENV.itemCaption;
alert("Item is clicked: " + text);
```

FIELD

FIELD returns a reference to the element associated with the eXtension.

Supposing you have applied a **Dropdown** eXtension to a field on the screen as illustrated below:



In the ***onSelectedValueChanged*** (an event that is triggered when a user selects an item in the dropdown) event we will have a script that look like this:

onSelectedValueChanged	FIELD.setValue(ROW.value)

The code above gets a reference to the **field element** associated with this **Dropdown** eXtension, and then sets the value to the currently selected row in the dropdown.

Since FIELD returns an element object, it will also have its own methods.

Summary of methods & properties:

Name	Type	Description
getValue	Method	Gets the current value of the element. Always returns a string.
setValue	Method	Sets the current value of the element.
isInputField	Method	Checks if the element is an input field.
getHtmlElement	Method	Gets the HTML DOM element associated with this element.
getProperty	Method	Gets the property value of an eXtension associated with this element.
setProperty	Method	Sets the property value of an eXtension associated with this element.
refresh	Method	Refreshes this element to reflect visually any changes made to the properties of this element.
focus	Method	Sets the input focus to this element.
set5250Cursor	Method	Position the **5250 logical cursor** on this element.
getIndex	Method	Gets this element's row index (starts from 1) in a sub-file.

The methods & properties will be covered in more details in its own section.

TABLEMANAGER

TABLEMANAGER returns a reference to the eXtension table manager. eXtension table is a concept that is similar to relational database tables. The data itself however can come from anywhere and does not necessarily come from a physical DBMS.

Tables are used as data source by various eXtensions. Some examples of eXtensions that are using tables as their data source are **Dropdown** and **Radio Buttons**.

The use of tables is discussed in more details in tutorial 6 of aXes eXtensions tutorial series.

Summary of TABLEMANAGER methods & properties:

Name	Type	Description
getTable	Method	Gets a reference to a table with the specified name (create a new table if one does not exist).
deleteTable	Method	Deletes the table with the specified name.
loadDynamicTable	Method	Loads the specified dynamic table from the host.
loadStaticTables	Method	Loads all static tables from the host.
loadFromXML	Method	Loads a specified table from an XML document on the server.

LANGUAGE

Gets the 2-character code representing the current language used.

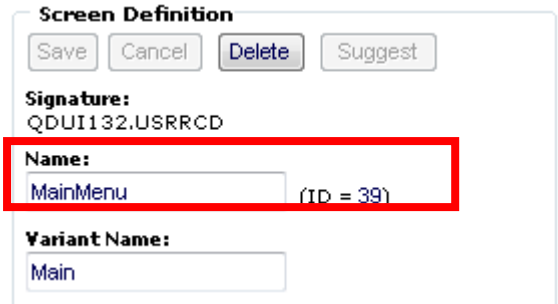
Example of usage:

```
var text = "";
if(LANGUAGE == "fr") text = "...";
else text = "...";
```

By default the current language is the language of the currently logged in Windows user, however this can be overridden by specifying the **lang** parameter as part of the URL when navigating to aXes: http://axeshost/ts/skins/ts.html?lang=fr

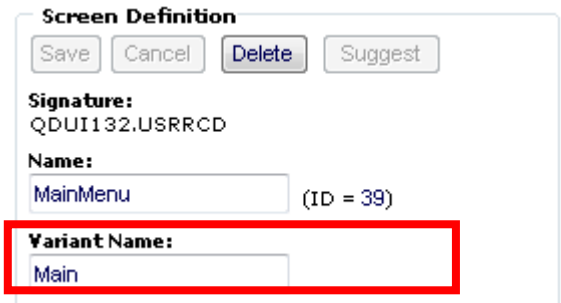
SCREENNAME

Gets the name of the current screen.



SCREENVARIANTNAME

Gets the variant name of the current screen.



FIELDS

Gets a reference to an element with the specified name on the current screen.

Parameters:

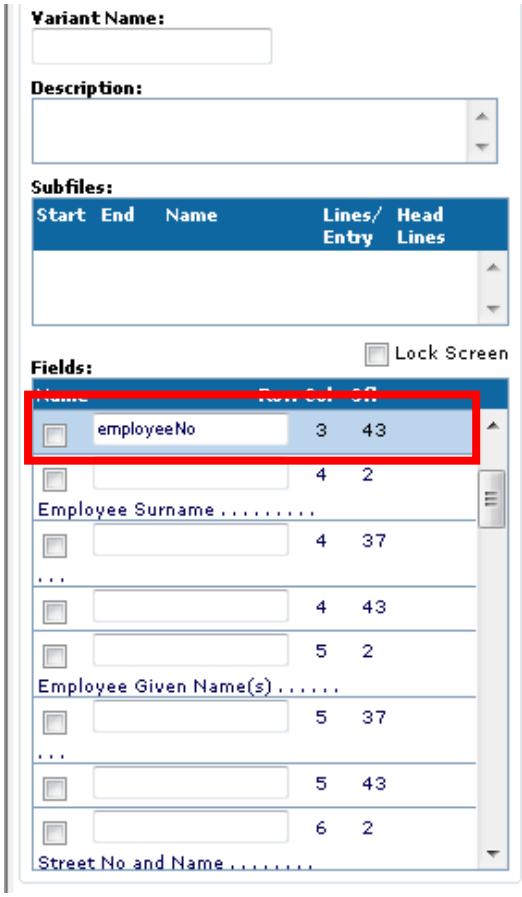
Name	Type	Mandatory	Description
name	String	Yes	Name of the element. Please see further explanation below about naming elements.
index	Integer	No	If the name describes a subfile column, index specifies which element in the column to return. Index starts from 1. If index is not specified, the first element in the column will be returned.

Return value:

Type	Description
ELEMENT	If an element with the specified name is found, it returns a reference to the element. If not found, null is returned.

Note that elements on the screen must be named first before they can be referenced through FIELDS function. The way to name screen elements depends on whether it's an existing 5250 field or a new screen element.

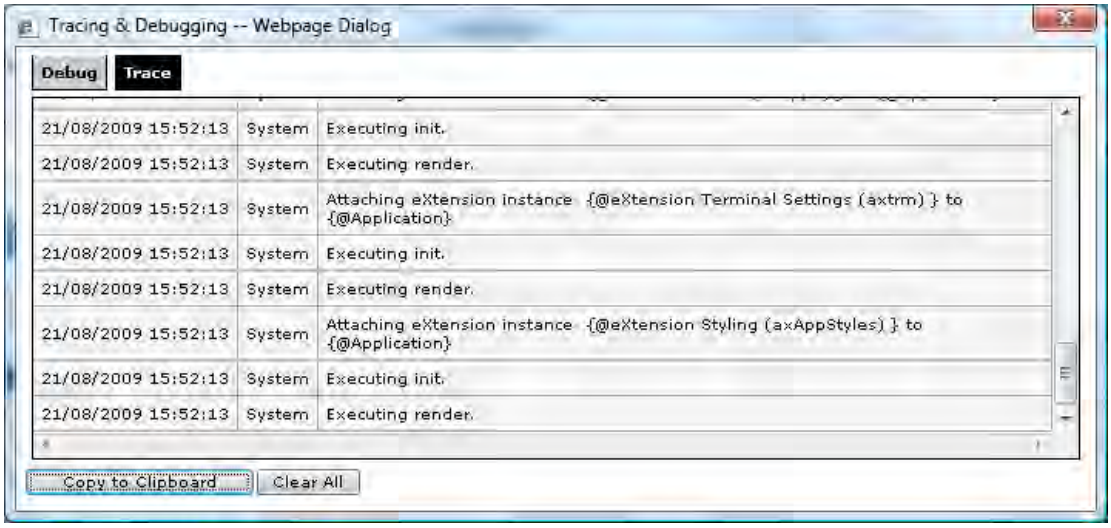
For existing 5250 fields, use the screen tab in the main aXes window (shown below) to assign names.



To assign a name to a new screen element, use the property sheet in the aXes Designer window (you need to select the element first by clicking on it).

TRACE

This function outputs a message to the aXes trace window. Trace messages will appear if parameter trace=all or trace=app is specified in the aXes URL.



aXes Tracing & Debugging window

Please refer to Tutorial 4 - Tracing and Debugging Techniques for more information on tracing & debugging techniques.

DEBUG

This function outputs a message to the aXes debug window. Unlike trace messages, debug messages will always appear without the need to specify additional parameter in aXes URL.

The difference between trace & debug messages is that trace messages are normally left in the system so later on it can be used to aid in troubleshooting of issues found after deployment, whereas debug messages are used during debugging. That's the reason why trace messages can be turned on and off whereas debug messages are always visible.

CTEXT

Get the text resource with the specified key from the user-defined resource file (Texts_Cust_ll.txt).

This method is mostly used for text externalization & globalization. Note that if you change a Texts_Cust_ll.txt file you should clear your browser cache to pick up the new file version.

Tutorial 3 discusses in more details about this topic.

SET5250CURSORTO

Set the **5250 logical cursor** to the specified field. This method is very useful for programs that are dependant on the cursor position in its navigation.

Please note that the cursor here does not necessarily refer to the actual cursor that might visible on the browser. You can still use this method even though your screen has been customized and you do not enable the limited cursor support property.

This method basically tells the physical program on the server that the cursor should move to the specified element.

Parameters:

Name	Type	Mandatory	Description
element	ELEMENT	Yes	Reference to the element to set the 5250 cursor to.

Return value:

None

SET5250CURSORPOS

Set the **5250 logical cursor** to the specified row and column. Please read about the **SET5250CURSORTO** method if you have not done so as it contains some important information about the cursor.

Parameters:

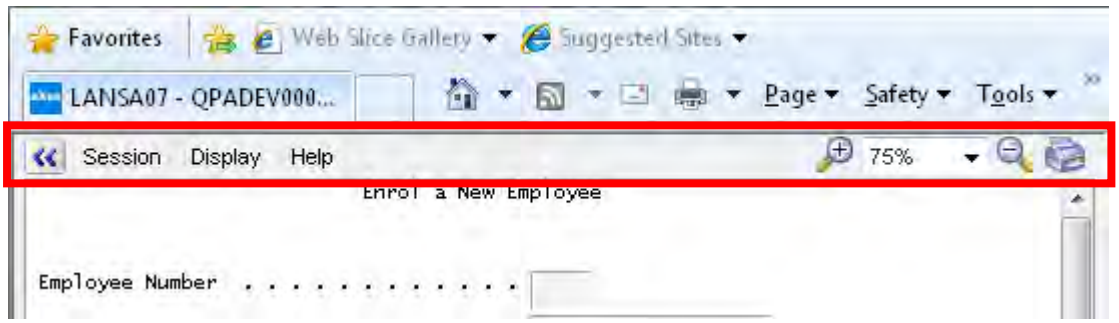
Name	Type	Mandatory	Description
row	Integer	Yes	Row number, starts from 1
column	Integer	Yes	Column number, starts from 1

Return value:

None

SHOWAXESMENUBAR

Show or hide aXes menu bar.



Parameters:

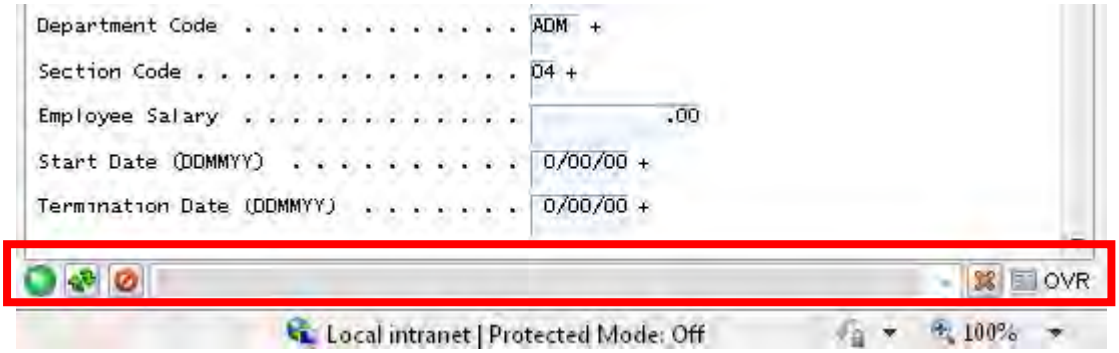
Name	Type	Mandatory	Description
show	Boolean	Yes	true → show menu bar, hide otherwise

Return value:

None

SHOWAXESSTATUSBAR

Show or hide aXes status bar.



Parameters:

Name	Type	Mandatory	Description
show	Boolean	Yes	true → show status bar, hide otherwise

Return value:

None

GETACTIVEELEMENT

Gets reference to the element that currently has input focus.

Parameters:

None

Return value:

Type	Description
ELEMENT	Element that currently has the input focus. If no element has the focus, null is returned.

GETFIELDATCURSOR

Gets a reference to the field at the current 5250 cursor position. This method is only available in aXes TS2.

Parameters:

None

Return value:

Type	Description
ELEMENT	Element at the current 5250 cursor position. If no element is at this position, null is returned.

ELEMENT Object Members

The object returned by **FIELD** & **FIELDS**() is an ELEMENT object.

Note that you you cannot return a reference to a field that uses extended DDS options to alter its visualization.

The members of an ELEMENT object are listed below.

getValue function

Gets the current value of the element.

Parameters:

None

Return value:

Type	Description
String	The current value of the element.

setValue function

Sets the current value of the element.

Parameters:

Name	Type	Mandatory	Description
value	String	Yes	the value to set

Return value:

None

isInputField function

Checks if the element is an input field or output field.

Parameters:

None

Return value:

Type	Description
Boolean	Returns true if input field, otherwise false

getProperty function

Gets the property value of an eXtension applied to this element.

Parameters:

Name	Type	Mandatory	Description
propertyName	String	Yes	Property name of the eXtension. Can be unqualified or fully qualified. A fully qualified name specifies the eXtension programmatic name in addition to the property name, separated by a period (for example: axButton.caption). Most of the time it's enough to specify an unqualified property name, however when the property name is ambiguous (two or more eXtensions have the same property), it is necessary to specify the fully qualified name of the property.

Return value:

The value of the property if an eXtension containing the specified property is found, null otherwise.

If the property contains a script, the script will **not** be evaluated. It will be returned as an **EMBEDDEDSCRIPT** object. Please see the example below for more information.

Example:

An element on the screen has the button eXtension applied to it. We need to get the value of the caption property of the button. Below is how to do it:

```
var element = FIELD;
// use un-qualified name – OK most of the time
var caption = element.getProperty("caption");
// if "caption" is ambiguous, fully qualify the name
var caption = element.getProperty("axButton.caption");
// check if the caption is a script
var caption = element.getProperty("axButton.caption");
if(caption instanceof EMBEDDEDSCRIPT)
{
	var script = caption.script;
}
```

setProperty function

Sets the property value of an eXtension applied to this element.

Note that for a change in a property value (that would affect the appearance of the element) to be reflected visually, **refresh** method needs to be invoked afterwards.

Parameters:

Name	Type	Mandatory	Description
propertyName	String	Yes	Property name of the eXtension. Can be unqualified or fully qualified. A fully qualified name specifies the eXtension programmatic name in addition to the property name, separated by a period (for example: axButton.caption). Most of the time it's enough to specify an unqualified property name, however when the property name is ambiguous (two or more eXtensions have the same property), it is necessary to specify the fully qualified name of the property.
value	Variant	Yes	The property value to set. To set the property as a script, create an EMBEDDEDSCRIPT object. See the example below for more information.

Return value:

None

Example:

An element on the screen has the button eXtension applied to it. We need to set the value of the caption property of the button. Below is how to do it:

```
var element = FIELD;
// use un-qualified name – OK most of the time
element.setProperty("caption", "My caption");
// if "caption" is ambiguous, fully qualify the name
element.setProperty("axButton.caption", "My caption");
// set caption to a script to show current date
var script = new EMBEDDEDSCRIPT("new Date().toString()");
element.setProperty("caption", script);
```

refresh function

Re-render the element to reflect any changes in properties.

Parameters:

None

Return value:

None

focus function

Set the input focus to this element.

Parameters:

None

Return value:

None

set5250Cursor function

Position the **5250 logical cursor** on this element. This method should be applied only to existing 5250 elements.

Please note that this method moves the 5250 cursor in the actual program in the host machine and is not directly related to the cursor that might or might not appear in the user's browser.

Parameters:

None

Return value:

None

Examples:

See Tutorial 11 – FAQ & Examples

getIndex function

Gets this element's row index (starts from 1) in a sub-file.

Parameters:

None

Return value:

Type	Description
Integer	Row index (starts from 1) if a this element is part of a sub-file, otherwise always returns 1.

get5250Attribute function

Get the 5250 attribute of this element.

Parameters:

None

Return value:

5250 attribute code.

If this element is not associated with a 5250 field, this method returns an empty string.

List of 5250 attribute code and their meanings:

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

TABLEMANAGER Object Members

getTable function

Gets a reference to a table with the specified name. If such a table does not exist, a new table will be created.

Parameters:

Name	Type	Mandatory	Description
Name	String	No	The name of the table. If not specified, a new unnamed table will be created.

Return value:

Type	Description
TABLE object	Existing table object matching the specified name or new table object

deleteTable function

Delete a table with the specified name.

Parameters:

Name	Type	Mandatory	Description
name	String	Yes	The name of the table to be deleted

Return value:

None

loadDynamicTable function

Loads the specified dynamic table from the host machine.

Parameters:

Name	Type	Mandatory	Description
name	String	Yes	The local name of the table. This will be the same name that you pass when calling getTable.
hostTableDefinitionFile	String	Yes	The name of the file on the host machine that defines how the data is to be retrieved.
parameters	Object	No	Object containing parameters for the data retrieval
keepLastKey	Boolean	No	If true, compare the key used to retrieve data with the key used in the previous request. If it is the same, do NOT load the table again. If false, the table will always be reloaded, even if the key is the same. The default is false.

Return value:

None

loadStaticTables function

Loads all static tables from the host machine.

Parameters:

Name	Type	Mandatory	Description
hostTableDefinitionFile	String	Yes	The name of the file on the host machine that defines the static tables
parameters	Object	Yes	Object containing parameters to be applied
forceReload	Boolean	No	Force reload of all static tables

Return value:

None

loadFromXML function

Loads all static tables from the host machine.

Parameters:

Name	Type	Mandatory	Description
hostXMLFile	String	Yes	The name of the XML file on the host machine that is to be used as the data source
table	TABLE	Yes	The table object to populate. Use getTable to create a new table if required.
forceReload	Boolean	No	Force reload of the XML.

Return value:

None

TABLE Object Members

insertChild function

Inserts a table child.

Parameters:

Name	Type	Mandatory	Description
parameters	Object	Yes	Object that represents a table row.
atBegin	Boolean	No	Add the child at the beginning of the table. Defaults to false.

Return value:

None

childCount function

Returns the number of table children (rows).

Parameters:

None

Return Value:

Type	Description
Integer	Number of table children (rows)

child function

Returns the table child object at the nominated index.

Parameters:

Name	Type	Mandatory	Description
index	Integer	Yes	The positional index of the child in the table.

Return Value:

Type	Description
Object	A table child (row) object

removeChildren function

Delete all table children (rows) effectively clearing the table.

Parameters:

None

Return Value:

None

childMember function

Returns a child *member* object at the nominated index with the nominated name.

Parameters:

Name	Type	Mandatory	Description
index	Integer	Yes	The positional index of the child in the table.
memberName	String	Yes	The name of the desired member. This is equivalent to the row property.
defaultValue	Variant	No	Default value when member not found.

Return Value:

Type	Description
Variant	The member when found, otherwise the default value.

aXes Object Functions and Features

With the introduction of aXes-TS2, various functions and features in the AXES object are being formally exposed for public use. These functions and features can be used in scripting provided the following warning is observed.

Important Warning

You should only use methods, objects and properties that are described in this documentation.

You are not permitted to use any scripting features that are not mentioned in this documentation.

If you ignore this warning then you are implicitly accepting responsibility for the risk of all of the following:

- ❖ The introduction into your application of incompatibilities with future versions of aXes.
- ❖ Voiding or limitation of any maintenance or support contract you may have in place for aXes.
- ❖ Being charged for problem resolutions that are traced back to being caused by such non-permitted usage.

These methods, objects and properties are for aXes-TS2 only.

Name	Type	Description
isTS2Engine	Boolean	Indicates if the aXes TS2 rendering engine is currently being used.
displayMessage	Method	AXES API that allows user defined messages to be added to the AXES status bar.

isTS2Engine

Use this Boolean value to determine if the aXes TS2 rendering engine is currently being used.

For example:

```
if(aXes.isTS2Engine) {<your TS2 code here>};
```

displayMessage

AXES API that allows user defined messages to be added to the AXES status bar.

In eXtension scripts this new API is invoked as:

```
AXES.Layout.displayMessage(msgtext,clearfirst,makesound)
```

In RAMP-TS screen definition scripts the API can also be invoked by using GLOBAL_oAXESInterface in place of AXES in the preceding example.

Using this API in RAMP-TS scripts would be very unusual because the aXes status bar is normally invisible. Typically messages are routed to the framework's main status bar area.

Parameters:

Name	Type	Mandatory	Description
msgtext	String	Yes	A mandatory string containing the text of the message to be added to the status bar.
clearfirst	Boolean	No	Default is false. It indicates whether other messages already present in the status bar area should be cleared when this message is added.
makesound	Boolean	No	Default is false. It indicates whether a sound should be made to draw the users attention to the message.

Return value:

None

General JavaScript Resources

External JavaScript Documentation

Click here to access some excellent online JavaScript documentation

http://www.w3schools.com/jsref/

There are also many good books available (such as JavaScript Bible by Danny Goodman, ISBN 0-7645-3188-3).

Converting Numbers to Strings

If you have a number in JavaScript variable and you want to convert it to a string use the toString() function. For example:

```
var number = 5.65;
var stringnumber = number.toString();

alert(stringnumber);


FIELD.setValue(stringNumber);
FIELD.setValue(number.toString());
```

Converting Strings to Numbers

If you have a string and want to convert it to a number then use the parseInt() method. For example this script returns integer values containing 1234 and 43 respectively into X:

```
X = parseInt("1234",10);
X = parseInt("34abc",10);
```

The second argument (10) specifies you want to use a base 10 numbering system. It's unusual to use anything for this parameter except 10 and you should always specify it as the default is a bit unpredictable.

If you need to have decimals then use parseFloat(). For example this script returns floating point values 1234.345 and 34.7 respectively into X:

```
X = parseFloat("1234.345");
X = parseFloat("34.7abc");
```

Remember that these are floating point values so they are not always as accurate or as predictable as signed or packed decimals numbers.

String Manipulation Functions

String variables in JavaScript have a number of very useful string functions. Here's a sample of the most commonly used:

Operation / Function	Example
Concatenation (+)	var S1 = "Customer"; var S2 = "123456"; var S3 = S1 + " " + S2 + "could not be found"; puts *Customer 123456could not be found* in variable S3.
IndexOf – finds first occurrence of a string in a string	/* 012345678901 */ var S1 = "ABCDHELLOABC"; var pos = S1.indexOf("HELLO"); will put the number 4 into variable pos.
lastIndexOf - finds last occurrence of a string in a string	/* 012345678901 */ var S1 = "ABCDHELLOABC"; var pos = S1.lastIndexOf("AB"); will put the number 9 into variable pos.

charAt – returns the character at a specific position in a string	/* 012345678901 */
	var S1 = "ABCDHELLOABC";
	var S2 = S1.charAt(4);
	var S3 = S1.charAt(9);
	will put "H" into S2 and "A" into S3.

length – returns the length of a string	/* 012345678901 */ var S1 = "ABCDHELLOABC"; var I = S1.length; will put the number 11 into variable I.
substring – returns the substring of string using a starting and ending point.	/* 01234567789 */ var a = "Hello World"; var b = a.substring(4,8); will put "o Wor" into b.
substr – returns the substring of a string using a starting position and a length	/* 01234567789 */ var a = "Hello World"; var b = a.substr(2,3); will put "llo" into b.
toLowerCase – returns the lowercase of string	var a = "Hello World"; var b = a.toLowerCase(); will put "hello world" into b.
toUpperCase – returns the uppercase of a string	var a = "Hello World"; var b = a.toUpperCase(); will put "HELLO WORLD" into b.

There are more string functions like these available.
See http://www.w3schools.com/jsref/jsref_obj_string.asp for more details.

Using alert()

The alert() function is your most useful tool for debugging errant scripts.
For example:

```
alert("About to send the enter key");
alert("The value of x is " + x.toString());
alert("The customer number is " + USERENV.currentCustomerNo);
```

Is This Variable Number or String?

Sometimes you have a variable in JavaScript and do not know whether it is a number or a string. You can test the type of a variable by using the typeof() operator like this:

```
var x = 1.234;
var y = "Hello";
var Type1 = typeof(x);
var Type2 = typeof(y);

alert(Type1 + " and " + Type2);
```

This code displays the message "number and string".

There are six possible values that typeof returns - "number," "string," "boolean," "object," "function," and "undefined." The most useful are "number", "string" and "undefined". The "undefined" is useful because it tells you that something does not exist yet (ie: it is undefined) so sometimes you see code like this:

```
if (typeof(USERENV.CustomerNumber) == "undefined") USERENV.CustomerNumber =
"12345";
```

Strange Behaviour in Scripts

A very common cause of strange behavior in scripts comes from not using the "==" comparison correctly. This simple script demonstrates a very common and time wasting scripting problem:

```
var X = 1;
alert("X is " + X);
if (X = 2) alert("X is 2");
```

If you execute this script this first alert message will show X is 1 and the second will show shows X is 2 ... which is not possible. The cause of this problem is of course that the if statement should have been

```
if (X == 2) alert("X is 2");
```

“Object Expected” or “XXXX is null or not an object” errors

When executing your scripts you will inevitably get an "object expected" or "xxxxxx is null or not an object" error at some stage.

This means you have referred to something in your script that does not exist.

The most common cause of these errors are simple typographic errors or even case errors. These simple script lines:

```
var iLimit = 1;
ilimit++;
```

will produce an "object expected" error - the variable is named i**L**imit.

This code:

```
var field = FIELDS("Commandline");
field.setValue("Hello");
```

will produce a "field is null or not an object" error. This is because the FIELDS() function returns null when it cannot find the named field on the current screen. The field name should have been "Command**L**ine", so variable field is set to null, causing the field.setValue() operation to fail.
When you get an "Object expected" or "XXXX is null error"

- Check the spelling of the name of objects you are referencing.
- Check the case of the names (eg: is it Userprofile or UserProfile?).

Extension Help Text

Application Level Basic Properties

Application Style Collection

Application Terminal Settings

axButton – Insert a Push Button into screen

axCheckBox – Checkbox Option

axDate – Insert date picker into screen

axDefaultVisualization - Default Visualization for a Screen Element

axDropDown – Insert a drop down into screen

axElementBasicProperties - 5250 Screen Element Basic Properties

axFKeyPanel – Insert a function key panel into screen

axFrame – Insert a frame into the screen

axGoogleChart – Insert a Google Chart into screen

axGraphPaper – Place Graph Paper on Screen

axGroupBox – Insert a group box into screen

axHTML – Insert raw HTML into screen

axHyperLink – Insert a hyperlink into screen

axImage – Insert image into screen

axInputboxStyle1 – Insert an HTML input element

axJQueryTheme – Change existing jQuery theme

axKeypad – provides a jQuery keypad

axLabel – Insert label into screen

axMessageHandler – Read and Display Messages from the 5250 Screen

axPopupWindow – Window Settings

axQuickPick – Insert quick pick menu into screen

axRadioButton – Insert radio buttons into screen

axScreenBasicProperties - Screen Level Basic Properties

axScreenDataViewer – Display data from another screen

axScreenSize - Auto Zoom Screen Size

axSimpleBox – Insert a simple box into screen

axUserElementBasicProperties - New Screen Element Basic Properties

axSimpleLine – Insert a simple line into screen

axSimpleStripe – Insert a simple stripe into screen

axSpinEdit – Insert a Spin Edit

axSubfileEnhancer – Enhance Subfile Appearance

axSubfileScroller – Insert Subfile Scroll buttons into a 5250 Screen

axTimer – Insert a timer into the screen

axUserElementBasicProperties - New Screen Element Basic Properties

Application Level Basic Properties

Description

Application level basic properties apply to an entire application.
An application is defined by the content of its definition set folder.

Properties

Name	Description and Comments	JavaScript	Shipped default
defaultTheme	Default theme to be used when AXES starts up.	Yes.	None.
autoGUIOn	The initial state of the AutoGUI feature. Overrides any value set in the TS1 skin by setting the gui_mode variable.	Yes.	True.
moveableWindows	Determines if 5250 windows can be moved by the user. This property applies to the TS2 engine only and will have no effect on TS1.	Yes.	True.
onApplicationStart	JavaScript code that is to be executed when the application starts up in the browser.	Yes.	None.
onApplicationEnd	JavaScript code that is to be executed when the browser navigates away from AXES.	Yes.	None.
onSignOn	JavaScript code that is to be executed as the aXes 5250 session is initiated after successful user sign-on.	Yes.	None.
onSignOff	JavaScript code that is to be executed as the aXes 5250 session is terminated normally after user sign-off.	Yes.	None.
strictLayoutGridForDbcs	Indicates whether strict layout grid should be used for languages such as Japanese	Yes.	False.

Notes, Comments and Warnings

You should not use alerts and any other operation that halts or interrupts the execution of JavaScript code during application initialization or termination.

Termination code may not be executed in an abnormal termination.

Most browsers have problems correctly converting mouse co-ordinates when sending events to objects that have been scaled. This can lead to some strange behaviour when dragging a window on a screen that has been scaled to anything other than 100%. In this case, you may want to use the moveableWindows property to prevent window movement.

Examples

(1). Specify this code in the **onApplicationStart** property:

```
TRACE("**** MY ONAPPSTART CODE IS EXECUTING. ****");
```

Specify this code in the **onApplicationEnd** property:

```
TRACE("**** MY ONAPPEND CODE IS EXECUTING. ****");
```

Now execute your aXes application with trace mode turned on.

You should see trace statements indicating that your code has been executed.

See Also

Using Themes

Can I create a script that runs when my application starts up and when the user has signed on?

How can I hide AXES menu bar and status bar?

Application Style Collection

Description

An application may contain a collection of named styles.

Named styles are designed to overcome a common problem ….

> Imagine that you decide that customized screens should display the most commonly used push button with a pale orange background.
>
> You have 50 screens and set about changing the style property for the most commonly used push button on each screen to #FF9933 - a pleasant shade of orange.
>
> Just after you finish, your marketing manager takes a look and declares that the shade of orange is "completely wrong" and that a pale yellow needs to be used instead.
>
> You now need to change 50 screen definitions, and risk having to do it again next week when the fickle marketing manager changes her mind yet again.
>
> The solution is to define an application style named "PrimaryButton" (say) that defines the color, and possibly other characteristics of the button. Then on each individual button you indicate that the style to be used is "PrimaryButton", instead of using a specific color.
>
> Next time the marketing manager changes her mind, you can implement a simple change to the application's "PrimaryButton" style - all the buttons on all 50 screens will change instantly.
>
> Understanding the benefits of this approach, and being quite wary of the fickle marketing manager, you will also understand why:

- Naming the application style "YellowButton" is a bad idea.
- Creating a style named "SecondaryButton" for all the other buttons is a good idea.

Properties

Name	Description and Comments	JavaScript	Shipped default
styles	Style collection. Each style in the collection has their own properties (see next table).	No.	None.

Properties of Style Item

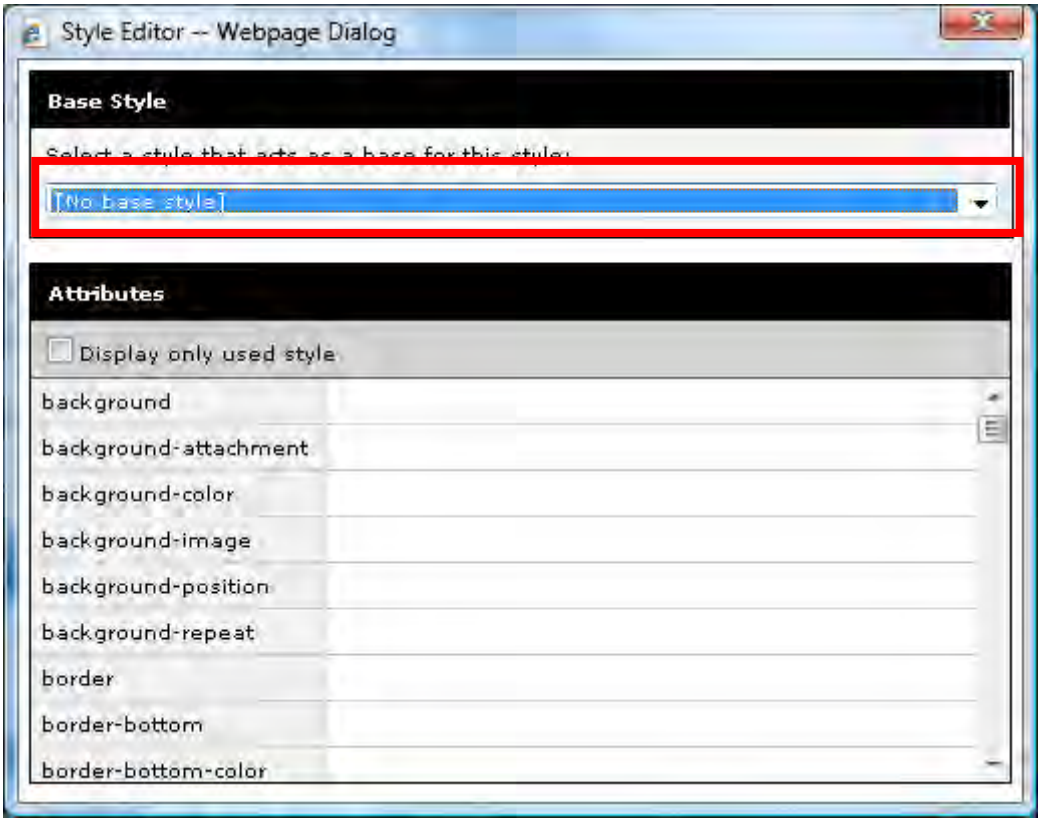
Name	Description and Comments	JavaScript	Shipped default
name	The symbolic name of the application style.	No.	None.
styleFor	If specified this style will automatically be applied to the specified element. Possible values for this property: ❖ **Blank** – the style will not be applied automatically to an element. To apply this style to an element you need to refer to this style from that element. ❖ **All** – the style applies to everything unconditionally. This is useful for specifying things such as font styles that should be consistent across the whole application. ❖ **Application Window** – the style applies to the main browser window. Most common use of this is to change the window background color or image. ❖ **Specified HTML Tag** – the style applies to the HTML tag specified in the ***htmlTag*** (see the next entry in the table) property. Note that this is an advanced feature and you should not attempt to use this unless you are familiar with Cascading Style Sheet.	No.	None.
htmlTag	Applicable only if ***styleFor*** attribute is set to ***Specified***	No.	None.

	HTML Tag.		
Style	The style details that define the style.	No.	None.
for5250Attributes	Any display attributes specified here will cause the style to be applied when a field with one of these attributes is encountered. This can be used, for example, to make fields in error appear differently to other input fields. Multiple display attributes can be specified for this property	No.	None.

Style Inheritance

A style can be based on another style – so if later you change some attributes in the base style, the inheriting style will immediately inherit those changes as well.

It is recommended that you design your styles in this fashion so as to avoid duplications and to make maintaining your styles easier.



See Also

Tutorial 3 - Advanced Screen Enhancement

Tutorial 10 - 5250 Screen Styling

Application Terminal Settings

Description

Application terminal settings specify how texts from 5250 terminal are rendered in the browser.

These properties will affect all screens in the entire application, including screens that haven't been customized.

Properties

Name	Description and Comments	JavaScript	Shipped default
rowHeight	The height of a row (in pixels).	No.	15
spaceBetweenRows	Space between two rows (in pixels)	No.	3
columnWidth	The width of a column (in pixels)	No.	8
spaceBetweenColumns	Space between two columns (in pixels)	No.	0

Notes, Comments and Warnings

You need to decide on the values of these properties prior to customizing any of your screens. If you decide to change these properties after some screens have been customized, some elements in the screens might move or become smaller or bigger and they will need to be manually repositioned and resized.

If it becomes necessary to modify any of these properties after you have started customization of screens, you will need to go through each of the customized screens and visually inspect that all elements are positioned correctly.

See Also

Changing 5250 Row / Column Size

axButton – Insert a Push Button into screen

Description

axButton allows a push button to be inserted into the 5250 screen.

Useability

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

Properties

Name	Description and Comments	JavaScript	Shipped default
caption	Specifies the text that is to appear on the button.	Yes. Must return a string.	None.
tooltip	The tool tip (or hint) that should appear when the user hovers the mouse over this button.	Yes. Must return a string.	None.
style	Additional style properties that should be applied to the button.	Yes. Must return a valid style object.	None.
sizeToField	Indicates whether this button should be sized to fill its containing field or automatically sized to match its content.	Yes.	True.
onClick	Optional. JavaScript code to be executed when the user clicks on the button.	Yes.	SENDKEY(ENV.defaultKey);
hidden	Indicates whether this button is hidden or visible.	Yes. Must return a Boolean.	False.
onFocus	Optional. JavaScript code to be executed when the element receives the focus.	Yes.	None.
onBlur	Optional. JavaScript code to be executed when the element lost the focus.	Yes.	None.

Notes, Comments and Warnings

None.

Examples

(1). On the System i Main Menu, with the command input field named Commandline, if this push button is added to the display:

```
MAIN                                    System i Main Menu

Select one of the following:

                                                          QPRINT Queue
      1.  User tasks
      2.  Office tasks
```

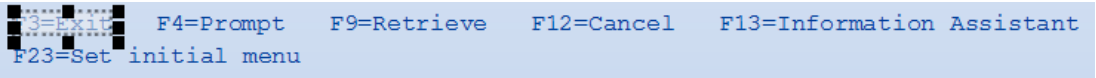
with this **onClick** code:

```
FIELDS("CommandLine").setValue("WRKOUTQ QPRINT");
SENDKEY("Enter");
```

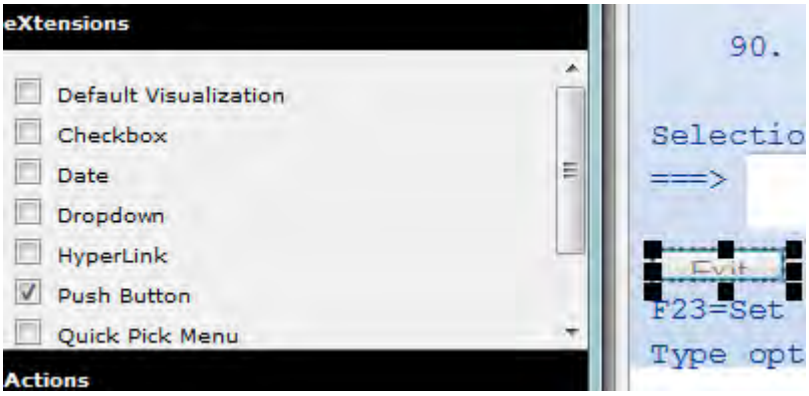
then clicking on the button will cause the output queue QPRINT to be displayed.

(2). On the System i Main Menu turn the function key strings Fn=xxxx into buttons:

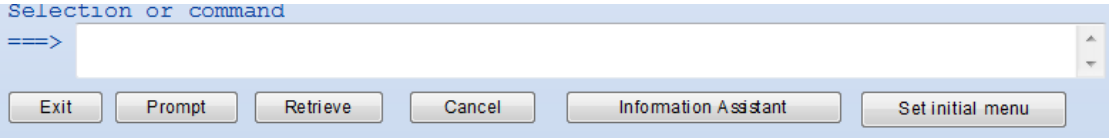
Click on F3=Exit to select it.



In the extensions list, select Push Button and F3=Exit should turn into a button with a caption of Exit:



Size the button as required and repeat for other desired function keys:



See Also

Using Screen Customization to Add Value

Using Scripting in Properties to Add Value

Push Buttons in Tutorial 3 - Advanced Screen Enhancement

Using Multilingual Captions on Buttons (and other places) in Tutorial 3 - Advanced Screen Enhancement

Using alert()

axCheckBox – Checkbox Option

Description

Presents a field as a Checkbox.

Useability

Context	Supported
Input screen fields	Yes
Output screen fields	Yes
Input subfile fields	Yes
Output subfile fields	Yes
New screen elements	Yes

Properties

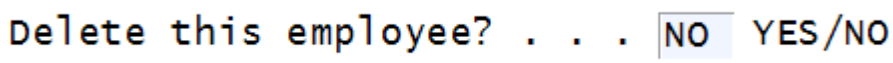
Name	Description and Comments	JavaScript	Shipped default
onSetValue	Specifies JavaScript code that returns true if the field contains the value set when the checkbox is checked.	Yes. Must return a boolean.	FIELD.getValue() == "Y"
onChecked	Sets the field to the nominated value when the checkbox is checked.	Yes.	FIELD.setValue("Y")
onUnchecked	Sets the field to the nominated value when the checkbox is unchecked.	Yes.	FIELD.setValue("N")
onChange	Optional. JavaScript code to be executed when the element changes state from unchecked to checked and vice versa.	Yes.	None.
onFocus	Optional. JavaScript code to be executed when the element receives the focus.	Yes.	None.
onBlur	Optional. JavaScript code to be executed when the element lost the focus.	Yes.	None.
style	Additional style properties that should be applied to the container of the label.	Yes.	None.

Notes, Comments and Warnings

The axChecBox extension is ideally suited to fields who's value has only two possibilities, e.g. "Y" or "N", "TRUE" or "FALSE", "Enabled" or "Disabled". The JavaScript of the supplied properties should be changed to use the required field values (see examples).

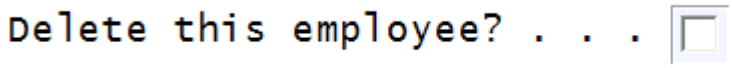
Examples

(1). If the value of the field can be either "YES" or "NO" the axCheckBox properties should be set as follows;



Property	Javascript
onSetValue	FIELD.getValue() == "YES"
onChecked	FIELD.setValue("YES ")
onUnchecked	FIELD.setValue("NO")

should produce a display like this …



Checking the checkbox displayed would set the field to "YES", confirming the deletion of the Employee to the underlying 5250 application.

(2). If the value of the field can be either "*ENABLED" or "*DISABLED" the axCheckBox properties should be set as follows;

Property	Javascript
onSetValue	FIELD.getValue() == "*ENABLED"
onChecked	FIELD.setValue("*ENABLED")
onUnchecked	FIELD.setValue("*DISABLED")

axDate – Insert date picker into screen

Description

axDate allows a date field to be displayed and edited in various ways.

This component can be dropped on a date field (input or output). The component will then display its own version of the field.

For input fields, a calendar button can also be displayed.

When the user clicks the calendar button, a calendar will display. The calendar will show the date from the field as the selected date, or if the date in the field is null or unrecognisable, the current date. If the user clicks on a day in the calendar, the date in the field will be changed accordingly. If the user does not click on a day, no change to the date will be made.

When the user clicks on the edit field (for input date fields only) the date displayed will be changed to the edit format, and selected. When the user moves away, the date format will revert to the normal display format.

Note: All properties can have a soft coded system default value. If a property is unspecified for an instance of this control, then the system default is checked, and if that is not found, the controls default value is used. This allows systems where the same date formats are used throughout to use DatePickers with no properties specified.

Useability

Context	Supported
Input screen fields	Yes
Output screen fields	Yes
Input subfile fields	Yes
Output subfile fields	Yes
New screen elements	Yes

Properties

Name	Description and Comments	JavaScript	Shipped default
dateFormatDisplay	This is the format the date will be shown in, unless the user has clicked on the date to edit it	Yes	language default
dateFormatEdit	This is the format the date will be shown in when the user clicks on the field to edit it	Yes	language default
dateFormatServer	This is the format the date would be shown in if there was no date extension applied	Yes	language default
showConfirmButton	Enables buttons for confirm and cancel on the calendar	Yes	false
calendarYearSelect	Allow the user to select a year from a drop down, on the calendar	Yes	false
calendarMonthSelect	Allow the user to select a month from a drop down, on the calendar	Yes	false
calendarThinImage	Use a thin image for the calendar icon	Yes	false
ignoreInvalidCharacters	If true this disables the aXes character checking that would occur when running in TS2 mode. This is useful when using this extension with numeric fields that use edit words to provide non-numeric date characters.	Yes	false
minimumDate	The minimum date that can be set.	Yes	None.
maximumDate	The maximum date that can be set.	Yes	None.
yearSelectValue	A positive value to be added and subtracted to the	Yes	None.

	current year; the corresponding range of values will be listed in the year dropdown element.		
onFocus	Optional. JavaScript code to be executed when the element receives the focus.	Yes.	None.
onBlur	Optional. JavaScript code to be executed when the element lost the focus.	Yes.	None.
onSelectedValueChanged	Optional. JavaScript code to be executed when the field value has changed.	Yes.	None.

Property Values for date formats

Caption	Description and Comments	Value
language default	This means use the format for the language the user is working in. (This format is specified in Text_Cust_xx.txt, where xx is the language code for the user)	default
day dd month yyyy	Show a value like "Sunday 26 April 2009"	DD d MM yy
dd/mm/yy	Show a value like 17/03/09, or 7/03/09	d/mm/y
dd/mm/yyyy	Show a value like 17/03/2009 or 7/03/2009	d/mm/yy
ddmmyy	Show a value like 170309 or 070309	ddmmy
similarly for other formats:		
mm/dd/yy		mm/dd/y
mm/dd/yyyy		mm/dd/yy
mmddyy		mmddy
yy/mm/dd		y/mm/dd
yyyy/mm/dd		yy/mm/dd
yymmdd		ymmdd
yyyy-mm-dd		yy-mm-dd
yy-mm-dd		y-mm-dd

Notes, Comments and Warnings

The *value* for date formats is specified according to the format rules for the jQuery Datepicker. The caption is a more intuitive version.

The properties of the Date Picker can be stored at different levels:

You can specify a default date format for all date pickers in the application, by using edit application properties, and then editing the properties of the date extension.

You can override the default date formats for an individual date picker on a screen, by editing its properties.

If you are building an application where different users will use different date formats depending on their language, then you need to specify the default date formats you want for each language (for dateFormatDisplay, dateFormatEdit, and dateFormatServer), in the Text_Cust_xx.txt file. Then, if the date format properties of the datepicker are "language default", it will use the language dependent format.

(The language for each user will be determined from their iSeries user profile)

The ignoreInvalidCharacters option is usually not necessary. It is only needed when working with a numeric field where the iSeries edit word for the field causes non-numeric characters (e.g. typically the / characters) to appear in some circumstances and not in others. This could happen if when the date is zero, the slashes are not shown, and when the date is non-zero the slashes are shown. For fields where the non-numeric characters always appear, then there is no need to use the ignoreInvalidCharacters option.

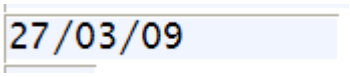
ignoreInvalidCharacters is applicable in TS2 mode only.

On mobile devices (Android, iPad, etc..), when the date input field receives the focus, the virtual keypad will appear prompting the user to enter the date string.
When the calendar button receives the focus, a calendar dialog only will appear, and the virtual keypad is hidden.

Examples

(1).

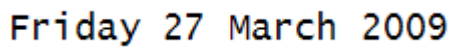
If the date was normally presented like this (without an extension)



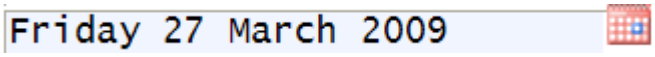
And a Date visualization was added to the field, and the properties were set as follows:
dateFormatDisplay: day dd month yyyy
dateFormatEdit: ddmmyy
dateFormatServer : dd/mm/yy

(where the dateFormatServer has been chosen to match the way the date would be shown without an extension)

When the date field appears in output mode, it will look like this:



When the date field appears in input mode, it will look like this:



If the user clicks on the field in input mode, it will look like this:

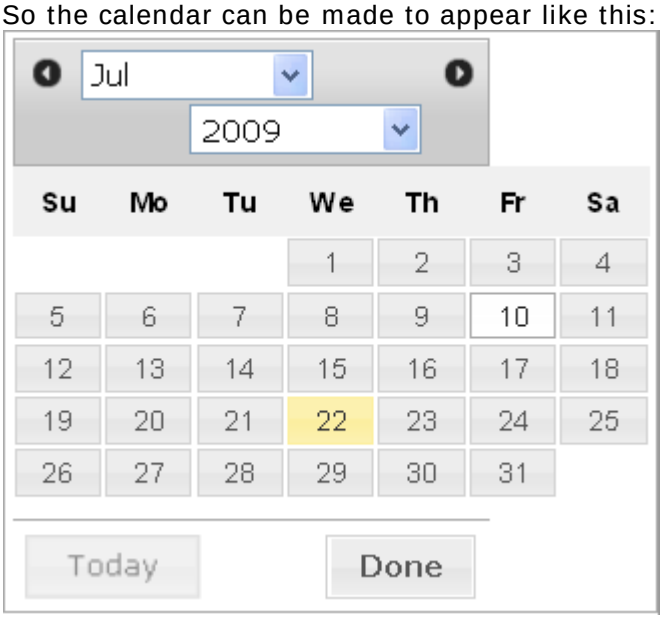
If the user clicks on the calendar button, a calendar will appear, like this:

The presence of the month and year selection dropdowns are controlled by the properties:

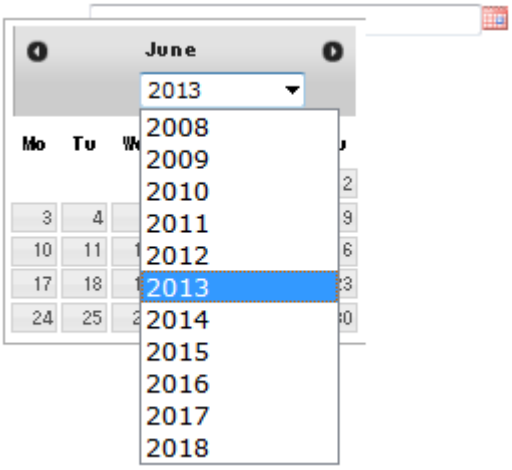
calendarMonthSelect
calendarYear Select

The presence of the Today and Done buttons is controlled by the property showConfirmButton

So the calendar can be made to appear like this:



To enable the setting of calendar year range, first set calendatYear to true, remove the minimumDate and maximumDate values, then set the yearSelectValue. Below is the example when yearSelectValue is set to 5.



In mobile devices where touch screens are prevalent and virtual keypad is the means for user input, clicking the calendar button will only show the date picker and not the virtual keypad. The virtual keypad will only show when the date field is clicked.

See Also

Dates in Tutorial 3 - Advanced Screen Enhancement

Example – Using a date format not available in the Date eXtension

axDefaultVisualization - Default Visualization for a Screen Element

Description

All 5250 and new screen elements have a default visualization.

It is used to visualize the element - unless you *specifically* indicate not to do this, typically because you want to:

1. Hide the element - by not visualizing it at all
2. Use an alternate visualization - like a radio button.

Useability

Context	Supported
Input screen fields	Yes
Output screen fields	Yes
Input subfile fields	Yes
Output subfile fields	Yes
New screen elements	Yes

Properties

Name	Description and Comments	JavaScript	Shipped default
value	A value override that the element should be set to initially.	Yes. Must return a string.	None.
style	Additional style properties that should be applied to the default visualization.	Yes. Must return a valid style object.	None.
removeCUADots	When a screen element is an output field you can use this option to remove CUA style double spaced trailing dots from it.	Yes. Must return a boolean.	False.
maxInputLength	When a screen element is an input field this property specifies the maximum number of characters the field will accept. Value of "0" is a special value and its meaning depends on the type of the element: • For existing 5250 field, value of 0 means that the original 5250 field's length will be used. • For new elements, value of 0 means that the field's length is unlimited. This property is designed intended for use with new input elements added to a 5250 screen. It is not intended for use with existing 5250 screen elements.	Yes. Must return a number.	0.
imeMode	When a screen element is an input field this property indicates whether the Input Method Editor (IME) should be active or inactive for this field. Possible values are 'Default', 'Active' and 'Inactive'. The 'Default' setting means that AXES will determine automatically whether to turn IME on or off based on the language used.	Yes. Must return "default", "active", or "inactive"	Default
selectContentOnFocus	Applies only to an input element. This property indicates whether the whole content of the input element should be selected when focus moves to the element.	Yes. Must return a boolean.	True

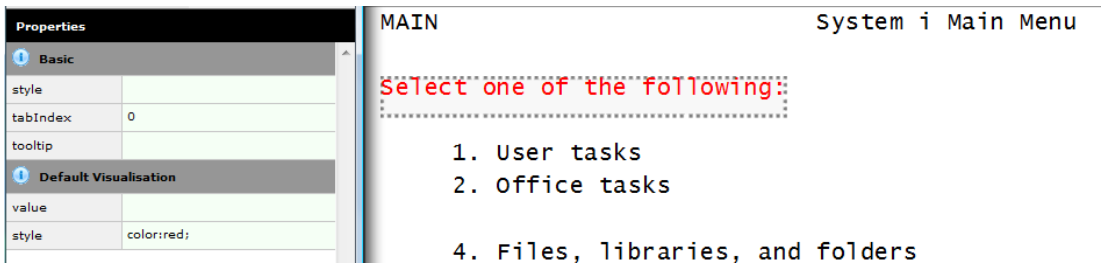
Notes, Comments and Warnings

Use of the value property should be considered carefully and tested.

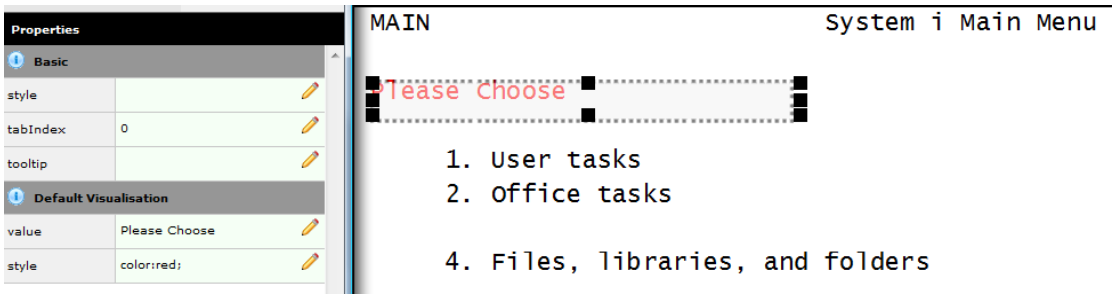
For browser compatibility reasons, setting the font-size in the style property will not work in TS2. Use the style property of the field's basic properties for this purpose instead.

Examples

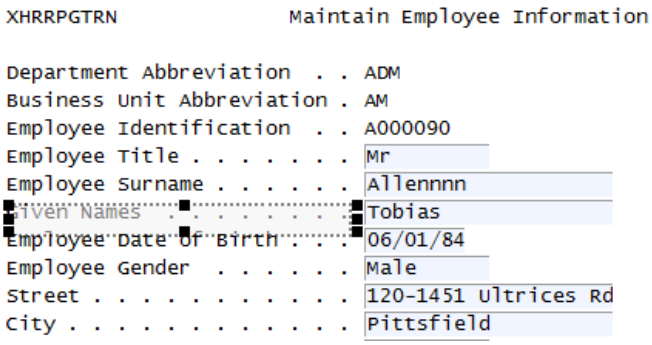
(1). Here the default visualization of the instruction line on the System i Main menu has been altered to use color red in the **style** property:



(2). Now the instruction has been altered by setting its **value** property to *Please Choose*:



(3). On this 5250 screen the *Given Name* field label is selected.



The **removeCUADots** property is changed to true, so the output field is now displayed like this:

```
XHRRPGTRN              Maintain Employee Information

Department Abbreviation  . . ADM
Business Unit Abbreviation . AM
Employee Identification  . . A000090
Employee Title . . . . . . . Mr
Employee Surname . . . . . . Allennnn
Given Names                  Tobias
Employee Date of Birth . . . 06/01/84
Employee Gender  . . . . . . Male
Street . . . . . . . . . . . 120-1451 Ultrices Rd
City . . . . . . . . . . . . Pittsfield
```

axDropDown – Insert a drop down into screen

Description

axDropDown allows a drop down to be inserted into the 5250 screen.

Useability

Context	Supported
Input screen fields	Yes
Output screen fields	Yes
Input subfile fields	Yes
Output subfile fields	Yes
New screen elements	Yes

Properties

Name	Description and Comments	JavaScript	Shipped default
dropDownStyle	CSS to apply to the drop down	No	Blank
dataSourceType	Indicate where the source data is obtained from.	No	fixedValues
tableName	For **Static Table** data source type, this must be the value set for the name = property.	No	Blank
onFillDropDown	JavaScript code to execute to fill the drop down.	Yes	Blank (i.e. no script to execute – interpreted as ROW.text)
xmlFileName	Name of XML file when the source type is XML.	No	Blank
xmlFileLocation	Path to locate the xmlFileName when the source type is XML.	No	*DEFAULT which is equal to /ts/screens/<private folder>
fixedValues	For **Fixed values** data source type you must enter the value pairs used to fill the drop down	No	No
additionalEntries	Use this property to add one or more entries to the drop down. Note that if you use this option with static tables, dynamic tables or XML documents, the values you are adding are inserted into the underpinning table – which subsequently causes them to appear in the drop down. This may impact other eXtensions or scripting that also use the same underpinning table.	Yes	No
onSelectValue	Determines the drop down entry to select.	Yes	Blank (i.e. no script to execute – interpreted as ROW.value)
onSelectedValueChanged	Event raised when the selected drop down entry changes	Yes	FIELD.setValue(ROW.value)
sqlQueryName	For **Dynamic Table** data source type the value set for the name = property	No	Blank

sqlVariables	For **Dynamic Table** data source the variables used in your query.		ENV.SQL.SQLVariableLibrary = USERENV.dftSQLDataLibrary
keepLastKey	For **Dynamic Table** data source. If this option is true, a dynamic table will only be reloaded if the key changes.	No	false
onFocus	Optional. JavaScript code to be executed when the element receives the focus.	Yes.	None.
onBlur	Optional. JavaScript code to be executed when the element lost the focus.	Yes.	None.

Notes, Comments and Warnings

***dataSourceType*:**

Determines the method used to populate the drop down. The options are:

Fixed Values: use this setting for drop downs with a small number of values when the values don't change.

Static Table: static tables are loaded once from the server at start up. The name of the static table must be the name specified in *tableName.*

Static tables are very efficient and they can contain hard coded values or can be loaded by executing SQL commands.

XML File: the data to fill the table is obtained from an XML file located by default in /ts/screens/<private folder>. To specify another location change the *xmlFileLocation* property.

Dynamic Table: use this option carefully as it is the only one where the data is loaded at every screen arrival with an expected performance impact in the rendering of the screen. When you use this option you must specify the name of the query in *sqlQueryName* as defined in your dynamic tables file.

This value indicates that the data should come from a dynamically created table.

- A dynamic table is (re)loaded every time a screen is displayed.
- Dynamic tables are expensive to use compared to static tables.
- Only use them when the table content needs to vary according to some dynamic criteria (eg: current company, current product, etc).
- Dynamic tables can only be loaded by SQL commands.

You can use the keepLastKey option to improve performance, if multiple requests using the same query key are commonly made.

Should I use a Static or a Dynamic Table?
You do not have to use a dynamic table just because you want to execute an SQL command to load the table. You can load static tables from SQL commands as well.
You can use the highly efficient static tables in ways you may not know about.
Imagine a scenario where a currency table's content needs to change according to a company number (ie: You can only use certain currencies for certain companies).
Your immediate reaction may be to just use a dynamic table - passing the company number as an SQL variable to control which currencies are selected by the SQL command that is executed to fill the dynamic table.
This will work, but compared to using a static table this approach is expensive to execute.

You can improve this approach in several ways by using static tables instead:
Option 1: Create a single Static Table named Currency:
It would contain all the currency details for all the companies - and the associated company number as an extra column. When filling a drop down you can use scripting in the *onFillDropDown* property to omit table entries that are not valid for the current company by returning ENV.returnValue as null (meaning no visible entry is to be added to the drop down).
Option 2: Create multiple Static Tables named Currency_CC:
Where _CC is the company number (say).
Script the drop down's *tableName* property to dynamically change.
You are using static tables - but now you are dynamically changing the table name - instead of the table content.
Note: This approach also works well with XML sourced tables. For example, you could also create an RPG program that runs every night (say).
It would read corporate "static" data files and flatten them out into a highly structured series of XML documents - where the name of the document (ie: table) is meaningful.
Option 3: Use start up or log on logic to load the "correct" set of static table data
When there is a clear point in the application where the user logs on and selects a company to work with - you can script an onArrive or onLeave events to force all static tables to load or reload.
Where they are loaded from SQL you can pass variables to the SQL executed to load them (eg: the selected company number).
Now the static table Currency would only ever contain the currencies for the currently selected company.
All the "users" of the Currency static table - like drop downs - no longer have to worry about the content - it will always be correct for the currently selected company.

What is a ROW.

The drop down makes use of the aXes table structures. It means that the data to populate the drop down is stored in memory in a table-like structure consisting of rows and columns. For example, consider a table defined like this:

```
DefineObjectInstance { className = "StaticTable",

  name    = "Gender",
  source  = "inline",

  rows = {
                {value="M",text="Male"},
                {value="F",text="Female"},
            },
  };
```

(Note that for such a table you would most likely set the *dataSourceType* to *Fixed Values* and type in the values manually).

At start up aXes populates the tables with one called *Gender*. It will have 2 rows, each **ROW** consisting of columns *value* and *text.* Using the Gender table, *ROW.value* would yield the current row's *value*= value and *ROW.text* the current row's *text*= value.

There is no constraint on how you name the row columns. If you use other names you must remember to change the default values to reflect the column names.

For example, if your file or fixed values define your columns like this:

```
  rows = {
                {codigo="M",descripcion="Masculino"},
                {codigo="F",descripcion="Femenino"},
             },
  };
```

The default values or script code should refer to *ROW.codigo* and *ROW.descripcion*.

***onFillDropDown* event**:

This event populates the drop down. It will fire (following our table example) whenever a screen (*any screen)* that has a drop down whose *tableName* was set to *Gender* arrives.

It will locate the *Gender* table and iterate through all its entries.

For each entry it will add an entry where the visible text in the drop down is equal to the value in *text* column of the Gender table. The result in our example will be a drop down with 2 entries: Male and Female.

This is the simplest way to use *onFillDropDown.*

A more powerful way would be to fill the drop down with a *value* resulting from the evaluation of a more complex expression.

If we extended the *Gender* table like this:

```
DefineObjectInstance { className = "StaticTable",

  name    = "Gender",
  source  = "inline",

  rows = {
                {value="M1",text="Male", agerange="16-25"},
                {value="M2",text="Male", agerange="26-35"},
                {value="M3",text="Male", agerange="36-45"},
                {value="F1",text="Female", agerange="16-25"},
                {value="F2",text="Female", agerange="26-35"},
                {value="F3",text="Female", agerange="36-45"},
             },
  };
```

We could change *onFillDropDown* to say:

ROW.text + "s aged " + ROW.agerange

The drop down would end up with entries that look like this:

Males aged 16-25
Males aged 26-35
Males aged 36-45
Females aged 16-25
Females aged 26-35
Females aged 36-45

Note that executing custom code is a relatively expensive process. This event is fired once for each entry in the dropdown. If you have a lot dropdowns and/or large

dropdowns this can have an impact on screen rendering time. The eXtension recognises the special case of an empty script (the default) and returns the value of ROW.text without having to execute any custom code. Consider carefully if you need to use this event, or if you can change the structure of your table so that ROW.text already contains the correct data.

***onSelectValue* event**:

The code in this event determines what drop down entry to set as the selected. It can be something simple like:

ROW.value

If the expression does not return a value of type *Boolean,* the default logic of comparing the value is returned with the associated field's value.

An example of an expression returning a value of type *Boolean* would be:

(ROW.value == "Hello")

Note that executing custom code is a relatively expensive process. This event is fired once for each entry in the dropdown. If you have a lot dropdowns and/or large dropdowns this can have an impact on screen rendering time. The eXtension recognises the special case of an empty script (the default) and returns the value of ROW.value without having to execute any custom code. Consider carefully if you need to use this, or if you can change the structure of your table to avoid it.

***onSelectValueChanged* event**:

This event fires when an entry in the drop down changes. The default behaviour here is

FIELD.setValue(ROW.value)

In other words, set the value of the field (which is referred to as FIELD) associated with the drop down to the value of the selected ROW.

***additionalEntries*:**

Lets you add entries in addition to those added by the specified method in *dataSourceType.*

The easiest and preferred way of adding a blank entry at the top of the drop down would be to add it to your static table or fixed values with blank values.

Another way would be to use *additionalEntries*.

This property works by calling an internal method named `addTableROW`. It receives 2 parameters:

`oRow`	`Object` - represents the ROW. It must be structurally the same as any of your other ROWs. In other words it must have the same properties.
`bAtBegin`	`Boolean` - Indicates whether to add the entry at the beginning or at the end of the drop down. Defaults to `false`

For example, to add an entry at the top of the drop down source by the *Gender* table you would this in your additionalEntries property:

```
var oGenderROW = {value: "", text: "Select an entry from this list", agerange:"" }
this.addTableROW(oGenderROW, true); /* Note the this qualifier */
```

Note that when the drop down is associated to a field, the field will be set to the "" when this entry is selected unless you add logic to the *onSelectValue* and *onSelectValueChanged*.

See Also

Dropdowns in Tutorial 3 - Advanced Screen Enhancement

Static Tables that load from static data in Tutorial 6 - Tables and XML Documents

Static Tables that load from a database file in Tutorial 6 - Tables and XML Documents

XML Documents in Tutorial 6 - Tables and XML Documents

Example – Dynamically refreshing a drop down without server interaction in **Tutorial 11 - FAQ and** Examples

Example – Two level drop down in Tutorial 11 - FAQ and Examples

Example – Using a drop down as subfile option field in Tutorial 11 - FAQ and Examples

axElementBasicProperties - 5250 Screen Element Basic Properties

Description

Every 5250 screen element has a fundamental and common set of properties, regardless of any further visualization that is applied to it.

5250 screen elements exist naturally on the 5250 screen because of the way it has been designed and the DDS used to display it.

Useability

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

Properties

Name	Description and Comments	JavaScript	Shipped default
style	Additional style properties that should be applied to the screen element.	Yes. Must return a valid style object.	None.
tabIndex	The tab index that should be applied to this element relative to all other elements on the screen.	No.	0
visible	Indicates whether this element is visible.	Yes.	True.
enabled	Indicates whether this element is enabled.	Yes.	True.
tooltip	The tool tip (or hint) that should appear when the user hovers the mouse over this screen element.	Yes. Must return a valid string.	None.
applyToInputOutput	Indicates whether this eXtension should be applied automatically to both input and output element	Yes	False.

Notes, Comments and Warnings

None.

Examples

(1). On the System i Main Menu, *option 8 – Problem handling* has its **tooltip** property set to *Please screen print and problem report details that you have and e-mail them to support@mycompany.com for further assistance*.

When the user hovers the mouse over the option the tooltip appears, like this:

```
     6. Communications

     8. Problem handling
     9. Display
    10. Information Assistant options
    11. System i Access tasks

    90. Sign off

Selection or command
===>
```

Please screen print and problem report details that you have and e-mail them to support@mycompany.com for further assistance.

axFKeyPanel – Insert a function key panel into screen

Description

A typical 5250 screen shows function keys in two rows at the bottom of the screen. aXes emulates the screen by displaying function keys as buttons in the same place. The aXes Function Keys Panel is a separate HTML panel that sits adjacent to a terminal server panel and contains the function keys displayed as buttons. The Enter key button is always available and the screen in the terminal server panel determines the other function keys available in the function keys panel. The terminal server panel includes no function keys when the function keys panel is active

Useability

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

Properties

Name	Description and Comments		JavaScript	Shipped default
position	Right	A vertical function keys panel will appear at the right of the terminal server panel	No	right
	Left	A vertical function keys panel will appear at the left of the terminal server panel		
	Top	A horizontal function keys panel will appear above the terminal server panel.		
	Bottom	A horizontal function keys panel will appear below the terminal server panel.		
horizontalWidth	The width of a horizontal function keys panel is fixed at the same width of the terminal server panel.		No	180
verticalHeight	The height of a vertical function keys panel is fixed at the same height of the terminal server panel.		No	40
resizePane			No	False
buttonWidth	The button width applies to all buttons		No	150
buttonHeight	The button height applies to all buttons		No	30
buttonPadding	The button padding applies to all buttons		No	5
buttonStyle			Yes. Must return a valid style object.	Flat
label	Text	Text is the button label extracted from the terminal server screen without the function key identification (i.e. Exit without the F3). Cancel and Exit are	No	Text Only

		examples of setting "text" as the label property value.		
	Function key	Function key shows only the function key identification. F12 and F3 are examples of setting "function key" as the label property value.		
	Both	The both value shows the button label from the terminal server screen and the function key. F12=Cancel and F3=Exit are examples of setting the "both" as the label property value.		
autoGenerateFunctionKeyScript onItemSelection			No Yes	True

Notes, Comments and Warnings

The function key panel extension works on TS2.

AutoGUI options for TS1 developer are not available, as the function key panel extension does not work on TS1.

Function key panel if configured as autoGUI options, dynamically inserts this extension into every available screen. Same properties appear on autoGUI options.

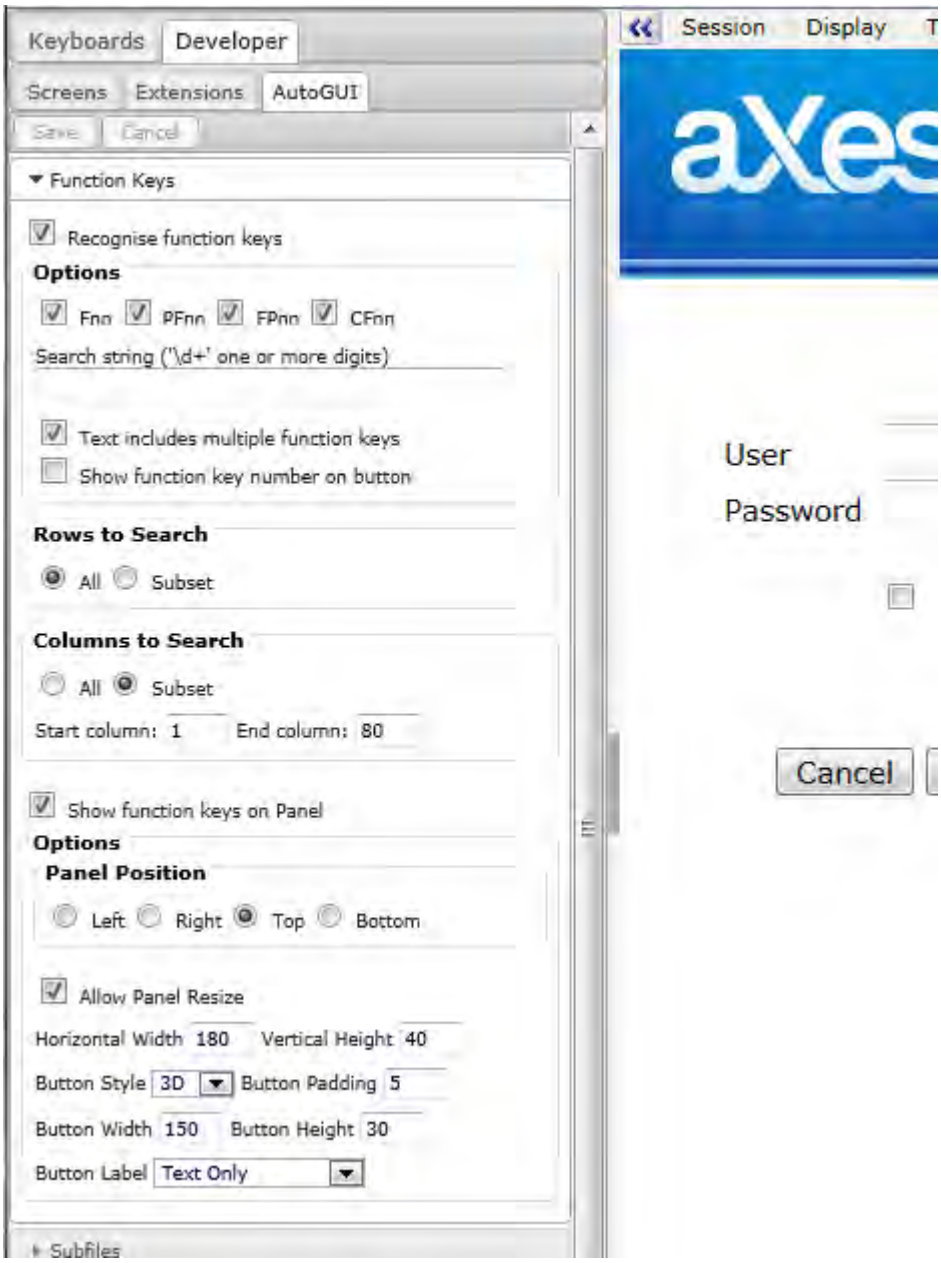
When using the function keys panel extension and autoGUI on customized screens, the function key panel extension takes priority and overrides autoGUI settings.

Button Label options to select text only, function key only and both text and function key overrides the “Show function key number on button” options.

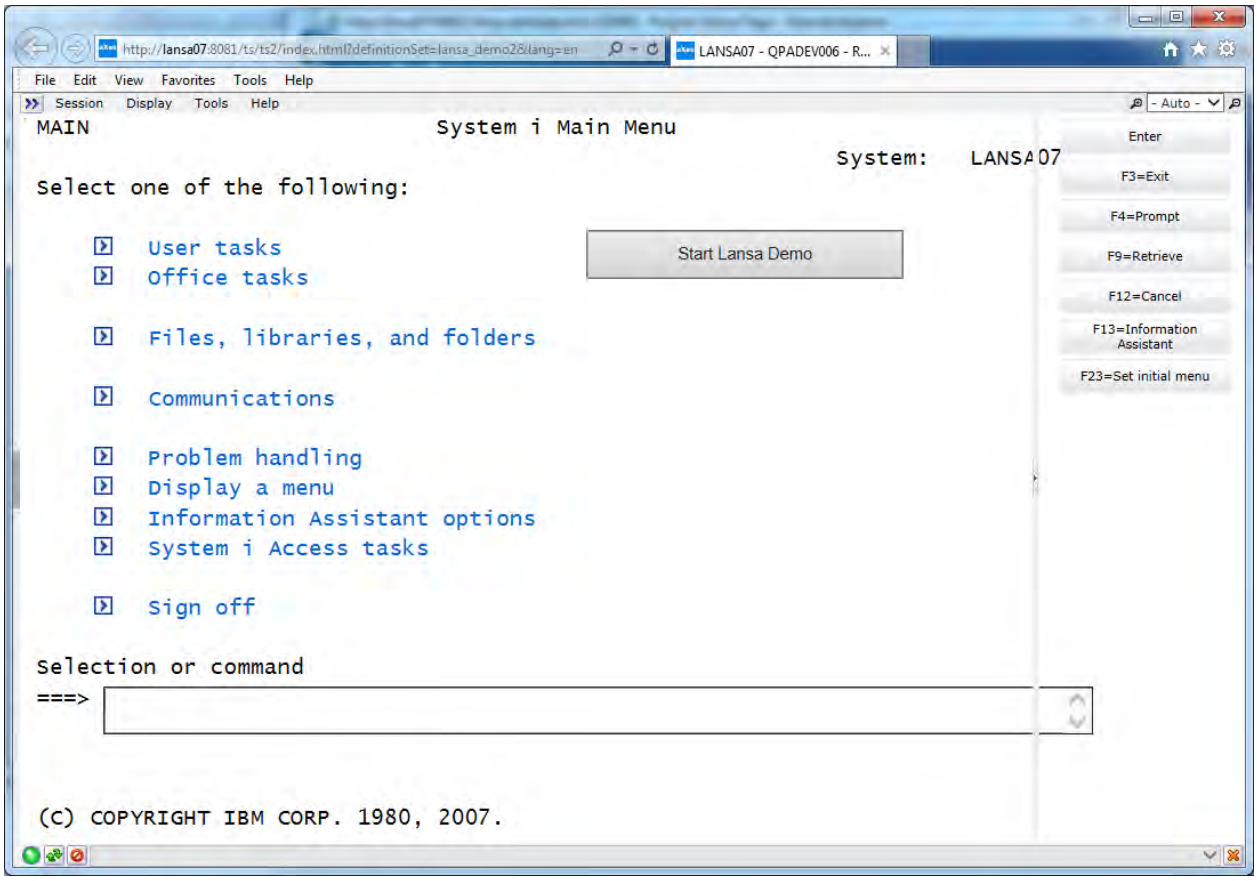
Screens containing selectable items, with select type “button”, will have the buttons positioned in the function keys panel. Screens with pop up windows that contain function keys retain the function key buttons and the buttons do not appear in the function keys panel.

AutoGUI Options

AutoGui options is accessible only on TS2 developer tools. Start TS2 developer tools within the development session or add “&dev” to URL of TS2 this will enable developer mode and AutoGui tab will be accessible within the developer tab panel. Check Show function keys on Panel to enable the options.
See screenshot below.


Keyboards
Developer
Screens
Extensions
AutoGUI
Save
Cancel
Function Keys
Recognise function keys
Options
Fnn
PFnn
FPnn
CFnn
Search string ('\d+' one or more digits)
Text includes multiple function keys
Show function key number on button
Rows to Search
All
Subset
Columns to Search
All
Subset
Start column: 1
End column: 80
Show function keys on Panel
Options
Panel Position
Left
Right
Top
Bottom
Allow Panel Resize
Horizontal Width 180
Vertical Height 40
Button Style 3D
Button Padding 5
Button Width 150
Button Height 30
Button Label Text Only
Subfiles
Session
Display
User
Password
Cancel

Function key panel
TS2 user mode



axFKeyPanel – Insert a function key panel into screen

Description

A typical 5250 screen shows function keys in two rows at the bottom of the screen. aXes emulates the screen by displaying function keys as buttons in the same place. The aXes Function Keys Panel is a separate HTML panel that sits adjacent to a terminal server panel and contains the function keys displayed as buttons. The Enter key button is always available and the screen in the terminal server panel determines the other function keys available in the function keys panel. The terminal server panel includes no function keys when the function keys panel is active

Useability

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

Properties

Name	Description and Comments		JavaScript	Shipped default
position	Right	A vertical function keys panel will appear at the right of the terminal server panel	No	right
	Left	A vertical function keys panel will appear at the left of the terminal server panel		
	Top	A horizontal function keys panel will appear above the terminal server panel.		
	Bottom	A horizontal function keys panel will appear below the terminal server panel.		
horizontalWidth	The width of a horizontal function keys panel is fixed at the same width of the terminal server panel.		No	180
verticalHeight	The height of a vertical function keys panel is fixed at the same height of the terminal server panel.		No	40
resizePane			No	False
buttonWidth	The button width applies to all buttons		No	150
buttonHeight	The button height applies to all buttons		No	30
buttonPadding	The button padding applies to all buttons		No	5
buttonStyle			Yes. Must return a valid style object.	Flat
label	Text	Text is the button label extracted from the terminal server screen without the function key identification (i.e. Exit without the F3). Cancel and Exit are	No	Text Only

		examples of setting "text" as the label property value.		
	Function key	Function key shows only the function key identification. F12 and F3 are examples of setting "function key" as the label property value.		
	Both	The both value shows the button label from the terminal server screen and the function key. F12=Cancel and F3=Exit are examples of setting the "both" as the label property value.		
autoGenerateFunctionKeyScript onItemSelection			No Yes	True

Notes, Comments and Warnings

The function key panel extension works on TS2.

AutoGUI options for TS1 developer are not available, as the function key panel extension does not work on TS1.

Function key panel if configured as autoGUI options, dynamically inserts this extension into every available screen. Same properties appear on autoGUI options.

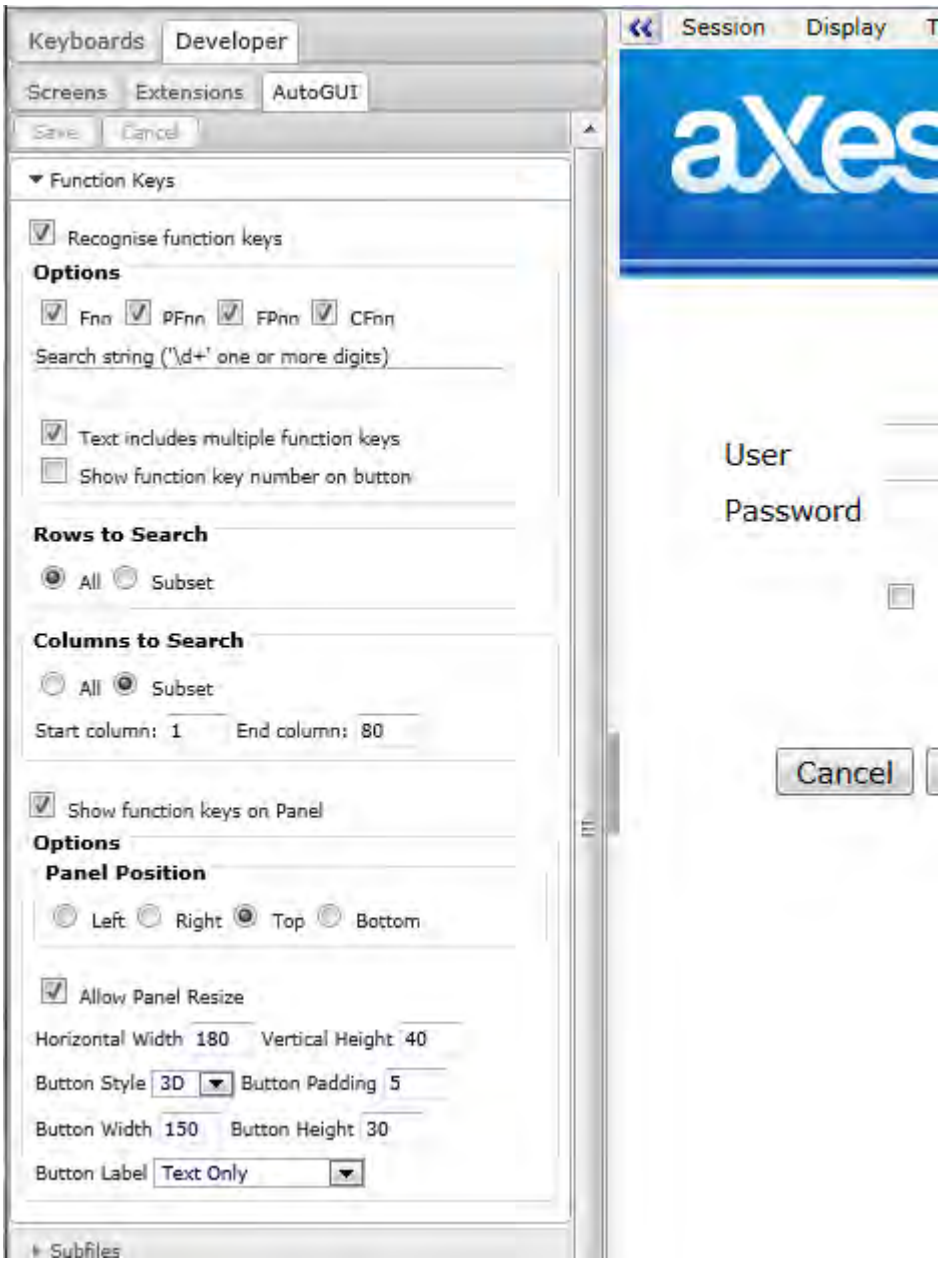
When using the function keys panel extension and autoGUI on customized screens, the function key panel extension takes priority and overrides autoGUI settings.

Button Label options to select text only, function key only and both text and function key overrides the “Show function key number on button” options.

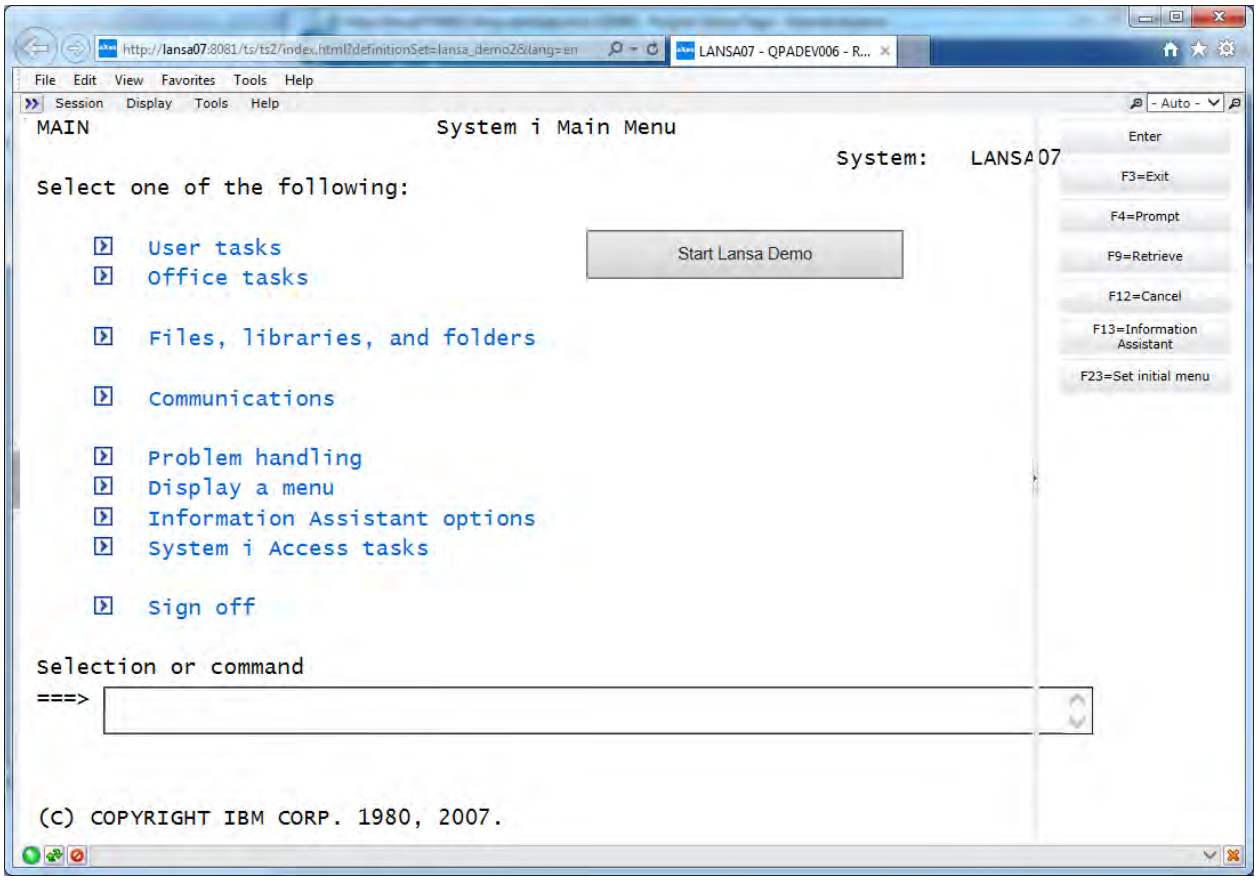
Screens containing selectable items, with select type “button”, will have the buttons positioned in the function keys panel. Screens with pop up windows that contain function keys retain the function key buttons and the buttons do not appear in the function keys panel.

AutoGUI Options

AutoGui options is accessible only on TS2 developer tools. Start TS2 developer tools within the development session or add “&dev” to URL of TS2 this will enable developer mode and AutoGui tab will be accessible within the developer tab panel. Check Show function keys on Panel to enable the options.
See screenshot below.


Keyboards
Developer
Screens
Extensions
AutoGUI
Save
Cancel
Function Keys
Recognise function keys
Options
Fnn
PFnn
FPnn
CFnn
Search string ('\d+' one or more digits)
Text includes multiple function keys
Show function key number on button
Rows to Search
All
Subset
Columns to Search
All
Subset
Start column: 1
End column: 80
Show function keys on Panel
Options
Panel Position
Left
Right
Top
Bottom
Allow Panel Resize
Horizontal Width 180
Vertical Height 40
Button Style 3D
Button Padding 5
Button Width 150
Button Height 30
Button Label Text Only
Subfiles
Session
Display
User
Password
Cancel

Function key panel
TS2 user mode



axFrame – Insert a frame into the screen

Description

axFrame allows a frame to be added to a 5250 screen.

The frame is typically used to display a web page from a corporate or external web site.

The frame can be made to fade away when the user moves the mouse out of it to another 5250 screen area - revealing what is visible beneath it. This allows for significant reuse of 5250 screen real estate.

There may be multiple frames on a 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
style	The style to be applied to the frame.	Yes	
src	The URL that the frame is to be loaded from. The load process, like all web pages, is asynchronous. The frame will not attempt to display itself if this property is set to an empty string.	Yes	/ts/blank_page.html
intFadeInterval	The initial fade interval. The use of fading is a way to preserve and reuse 5250 screen real estate. Refer to the examples following. This specifies, in milliseconds, the initial time taken for the frame to fade by 10%. When the frame has faded to zero it is made invisible and can no longer be accessed. The default value of 600 indicates that it takes 6 seconds (600 milliseconds X 10 fades) for the frame to become invisible. Set this property to zero to turn off frame fading. Frame fading is initiated when the user moves the mouse out of the frame area. Frame fading stops and the fade effect is removed if the user moves the mouse back over the frame any point before it becomes invisible.	Yes	600

subsFadeInterval	The subsequent fade interval. This specifies, in milliseconds, the subsequent time taken for the frame to fade by 10%. The default value of 150 indicates that it takes 1.5 seconds (150 milliseconds X 10 fades) for the frame to become subsequently invisible. Set this property to zero to turn off frame fading.	Yes	150

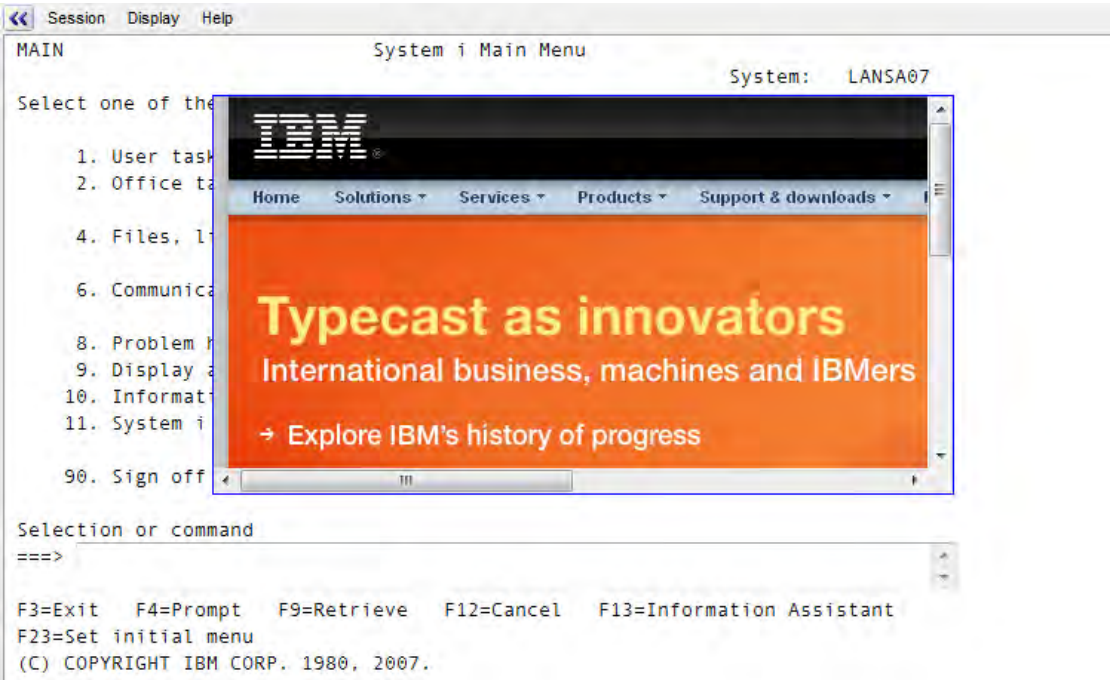
Examples

(1). Add the IBM web site to the System i Main menu.

A frame is added to the System i Main Menu as a new element with these properties:

Frame	
style	border:1px solid blue;
src	http://www.ibm.com
initFadeInterval	0
subsFadeInterval	0

The resulting display looks like this:



---

(2). Fade the IBM web site to the System i Main menu.

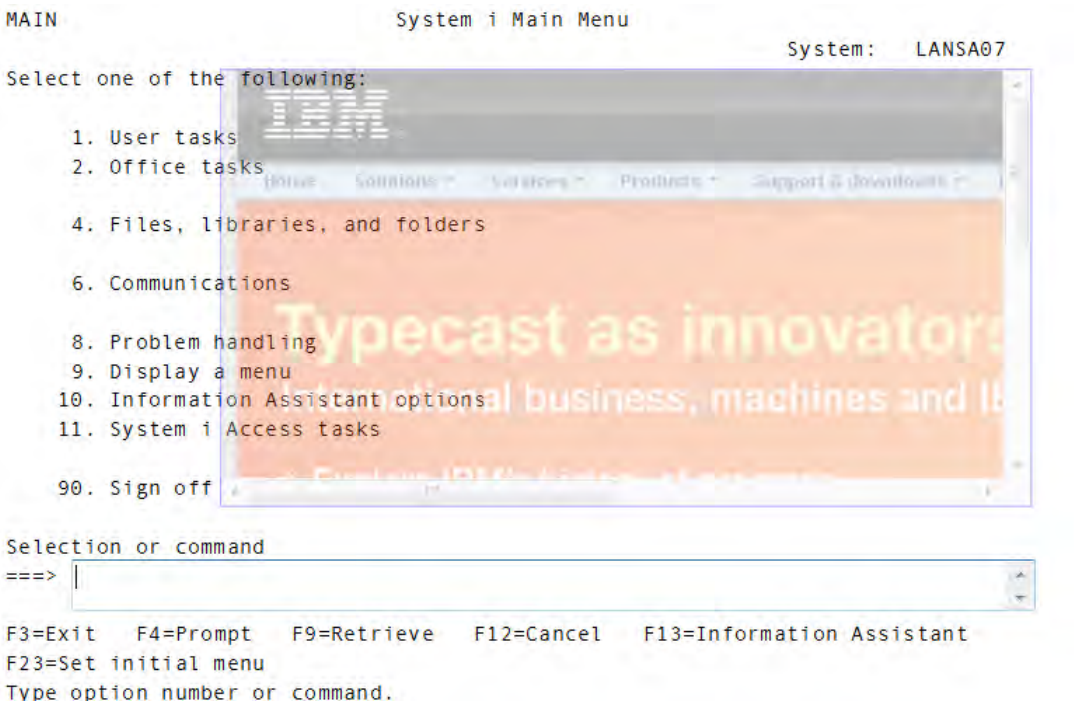
The frame in example 1 is changed to have these properties:

Frame	
style	border:1px solid blue;
src	http://www.ibm.com
initFadeInterval	1000
subsFadeInterval	50

These indicate that when the frame is displayed initially it will fade away in about 10 seconds (1000 milliseconds is defined as the time for each 10% of the fading) unless the user positions the mouse over it.

If they subsequently move the mouse off the frame it will fade away in about 0.5 seconds (50 milliseconds is defined as the time for each 10% of the fading).

The display now looks like this:



The frame initially fades away over 10 seconds. If the user moves the mouse over the frame the fading stops and the fading effect is removed from the frame. When the user subsequently moves the mouse out of the frame it fades away almost instantly.

---

(3). Reusing 5250 screen real estate

The command input field on the System i Main menu is named *CommandLine*.

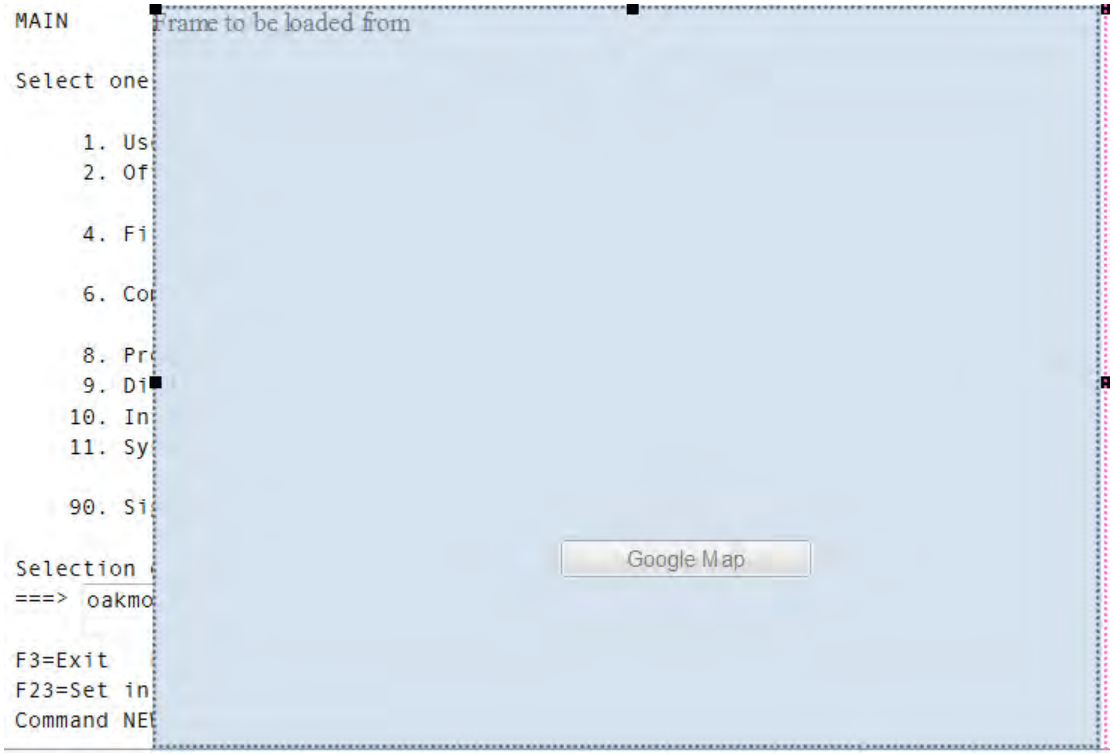
A frame is then added to the System i Main Menu with these properties:

Properties	
Basic	
name	GoogleMap
style	left:90px;top:40px;width:550px;h
type	Input
tabIndex	0
visible	True
enabled	True
tooltip	
Frame	
style	border:1px solid orange;
src	
initFadeInterval	2000
subsFadeInterval	50

Significantly, the frame:

- is named ***GoogleMap***.
- has no src (source) value specified.
- has a long initial total fade time of 20 seconds.
- has a short subsequent total fade time of 0.5 seconds.

The frame is sized to use a very large portion of the 5250 screen, like this:



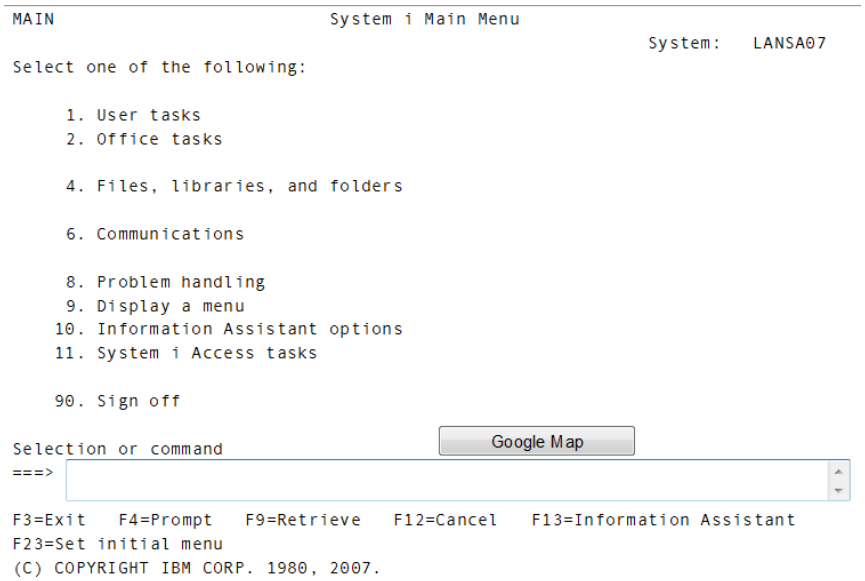
Next a button is added as new element with caption *Google Map* and this *onClick* script:

```
/* Get the value typed into the command line */

var cl = FIELDS("CommandLine").getValue();

/* If it is blank/null then show an error */

if (cl == "")
{
   window.alert("Enter the location details for the map");
}

/* Display the Google map details using the command */
/* details as the requested address details         */

else
{
   var gm = FIELDS("GoogleMap");
   var url = "http://maps.google.com/maps?q=" + cl;
   gm.setProperty("src",url);
   gm.refresh();
}
```

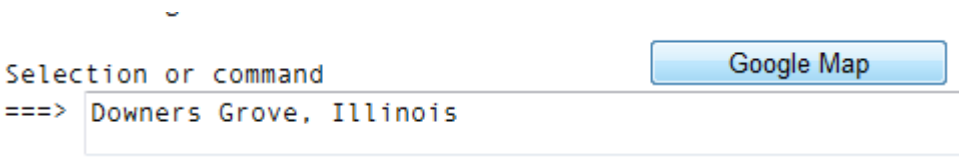
This script takes the value from the CommandLine and passes it to Google Maps.

In use, this example works something like this:

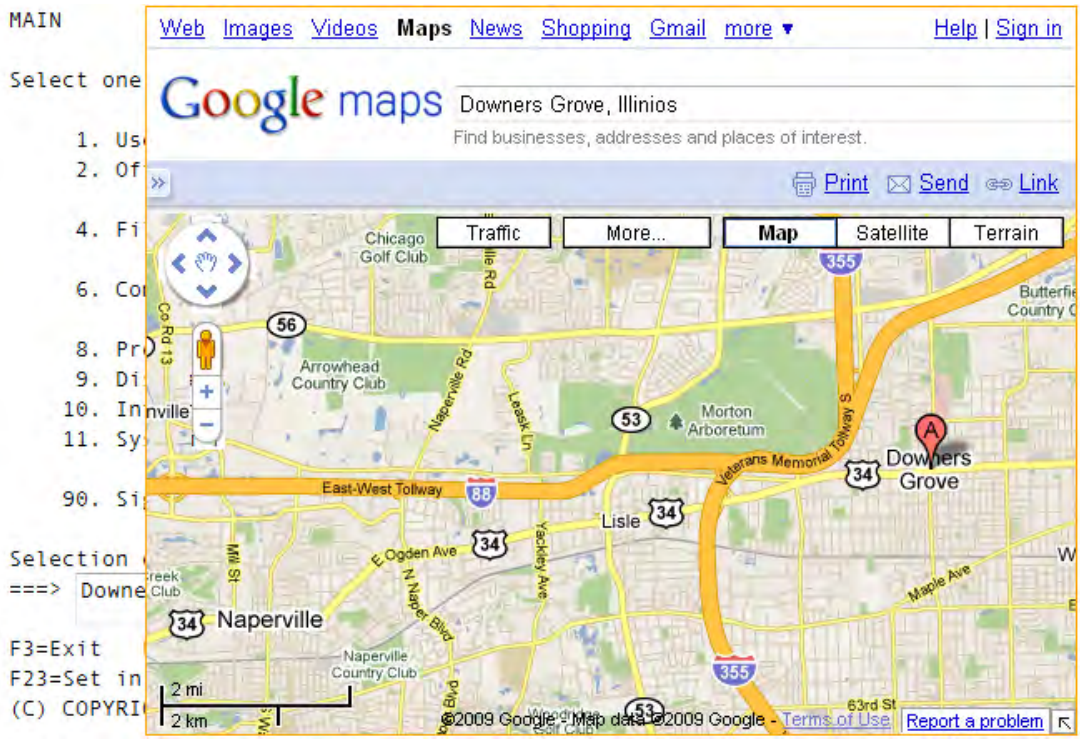
Initially the frame does not appear because it has a blank/null src property:



The user enters an address and clicks the Google Map button:



The frame is now activated because it has a *src* property, so it displays the Google map details:



The frame will stay on display as long as the user keeps the mouse over it.

Once they move the mouse outside it, back to the 5250 screen, it will fade away very quickly and become invisible - allowing the user to perform another Google map search or to resume using the normal 5250 options on the System i Main menu.

---

(4). Introductions and Welcomes

In this example a simple and visually crude "corporate home page" HTML document named ***MyCompany.html*** is added to the aXes ts/screens folder.

Its source code is like this:

```
<html>
<head>
</head>
<body style="font-family:Verdana;background-color:#FFFF99;">
<p style="text-align:center;">Welcome to ACME</p>
<dir>
<li style="margin-top:20px;"><b>Company HR Policy</b></li>
<div style="margin-left=40px;">Click in here to read about ..... etc, etc, etc, etc</div>
<div style="margin-left=40px;">Click in here to read about ..... etc, etc, etc, etc</div>
<div style="margin-left=40px;">Click in here to read about ..... etc, etc, etc, etc</div>
<div style="margin-left=40px;">Click in here to read about ..... etc, etc, etc, etc</div>
<li style="margin-top:20px;"><b>Upcoming Company Social Functions</b></li>
<div style="margin-left=40px;">Click in here to read about ..... etc, etc, etc, etc</div>
<div style="margin-left=40px;">Click in here to read about ..... etc, etc, etc, etc</div>
<div style="margin-left=40px;">Click in here to read about ..... etc, etc, etc, etc</div>
<div style="margin-left=40px;">Click in here to read about ..... etc, etc, etc, etc</div>
<li style="margin-top:20px;"><b>Vist the Company Facebook to get to know your Peers</b></li>
<div style="margin-left=40px;">Click in here to read about ..... etc, etc, etc, etc</div>
<div style="margin-left=40px;">Click in here to read about ..... etc, etc, etc, etc</div>
<div style="margin-left=40px;">Click in here to read about ..... etc, etc, etc, etc</div>
<div style="margin-left=40px;">Click in here to read about ..... etc, etc, etc, etc</div>
</DIR>
</body>
</html>
```

Hint: Set user *PUBLIC to have *R rights only for any new documents you add to aXes folders.

It looks like this when displayed in a browser page:

Welcome to ACME

- **Company HR Policy**
  Click in here to read about ..... etc, etc, etc, etc
  Click in here to read about ..... etc, etc, etc, etc
  Click in here to read about ..... etc, etc, etc, etc
  Click in here to read about ..... etc, etc, etc, etc

- **Upcoming Company Social Functions**
  Click in here to read about ..... etc, etc, etc, etc
  Click in here to read about ..... etc, etc, etc, etc
  Click in here to read about ..... etc, etc, etc, etc
  Click in here to read about ..... etc, etc, etc, etc

- **Vist the Company Facebook to get to know your Peers**
  Click in here to read about ..... etc, etc, etc, etc
  Click in here to read about ..... etc, etc, etc, etc
  Click in here to read about ..... etc, etc, etc, etc
  Click in here to read about ..... etc, etc, etc, etc

Hint: We very strongly recommend that you consult with a professional graphics designer if you intend to build a corporate home page.

Next a property is added to the USERENV object to count how often the System i Main menu has been displayed in the current session:

```
var USERENV =
{
    mainMenuUseCount : 0, /* Count how many times the main menu is used */
```

Hint: You will need to stop and restart your development session to pick up the modified USERENV.

Next a frame is added to the System i Main menu with these properties:

Basic	
name	Introduction
style	left:20px;top:20px;width:61
type	Input
tabIndex	0
visible	True
enabled	True
tooltip	
Frame	
style	border:1px solid gray;
src	
initFadeInterval	600
subsFadeInterval	150

Significantly, the frame:

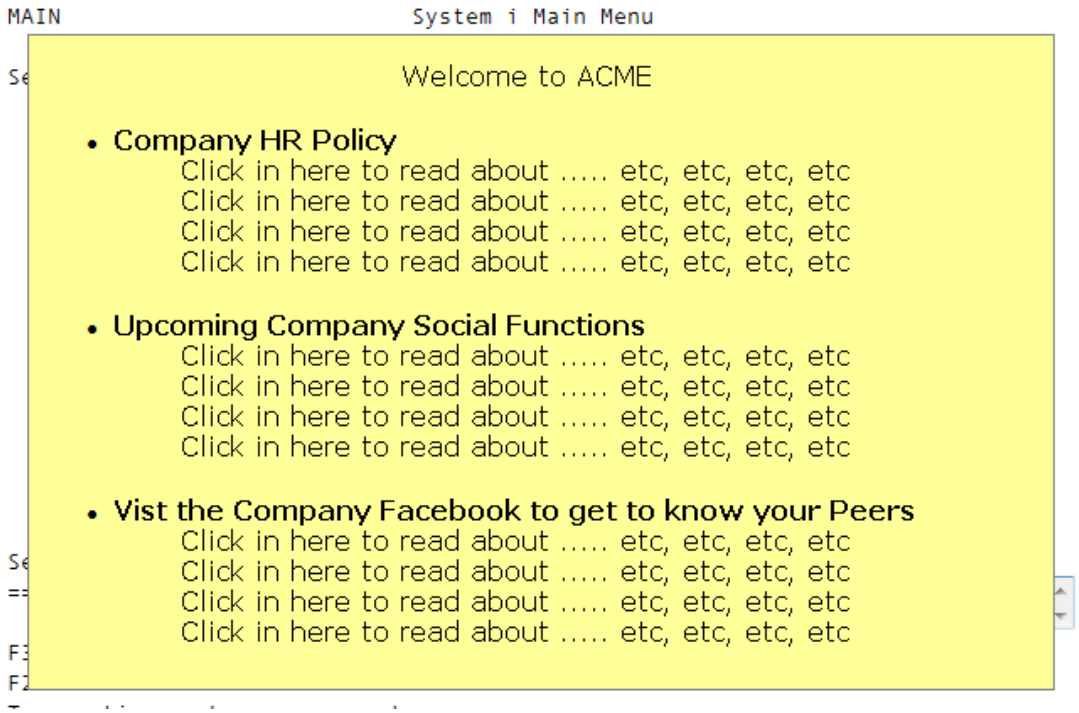
- is named ***Introduction***.
- has no src (source) value specified.
- has a long initial total fade time of 6 seconds.
- has a short subsequent total fade time of 1.5 seconds.

Finally, this script is a added to the ***onArrive*** property of the System i Main menu screen itself:

```
/* Increment count of times menu used */

USERENV.mainMenuUseCount++;

/* If this is the first time show the introduction */

if (USERENV.mainMenuUseCount == 1)
{
   var intro = FIELDS("Introduction")
   intro.setProperty("src","/ts/screens/myCompany.html");
   intro.refresh();
}
```

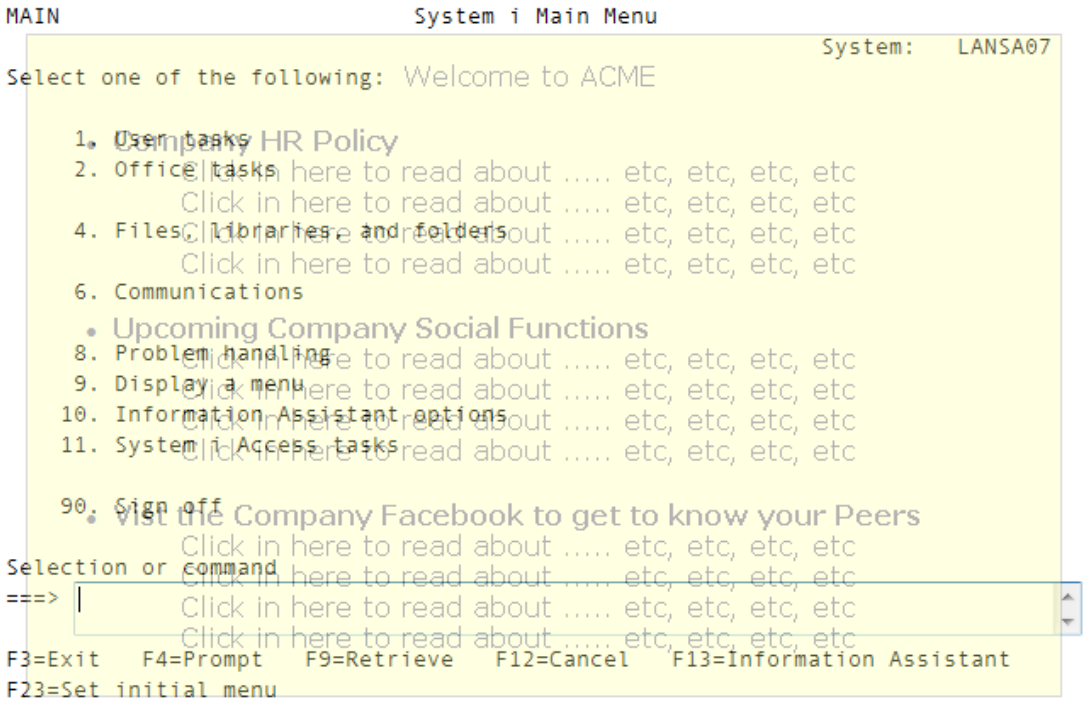
This script is counting how many times the System i Main Menu is displayed. It shows the introduction frame only the first time that the screen arrives.

So when the user signs on and the System i Main Menu displays for the first time they see this “corporate home page” frame overlaying the 5250 screen:



As long as they keep the mouse over the frame they can work within it.

Once they move the mouse outside the frame it fades away in about 1.5 seconds.



If the mouse is not over frame on the initial display the frame would fade away much more slowly - in about 6 seconds.

Subsequent redisplays of the System i Main Menu do not show the frame at all because the src property is not set to anything by the System i Main Menu's ***onArrive*** script.

These types of corporate home pages are useful to keep employees up to date.

Additionally, they can be changed externally at any time in just a few seconds - without any changes being required to the 5250 application or to any aXes customizations.

Additionally they easily link off to other HTML pages, web sites, etc.

axGoogleChart – Insert a Google Chart into screen

Description

axGoogleChart allows to embed of a Google Chart into your screen.

By using this extension you will be subject to the Google Terms of Service as outlined in http://code.google.com/apis/chart/terms.html

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
sourceParmString	The source of the parameters to add to the google chart URL.	No	Table
tableName	For **sourceParmString** = table, supply the table name where to get the parameter URL information from. This must be the same name as the one specified in the getTable() call – see **onLoadChart** and examples.	No	Blank
userString	For **sourceParmString** = User String, supply a valid string to append to the Google Chart URL as parameters.	No	Blank
appendString	For **sourceParmString** = table, append this string to the URL parameters.	No	Blank
onLoadChart	Code to execute prior to invoking the Google Chart URL.	Yes	Blank
type	The type of Google Chart. Maps to the **cht=** Google chart parameter.	No	Blank
colors	The colors to apply to the chart. Maps to the **chco**= Google chart parameter and **must be specified according to its rules**.	No	Blank
legends	Chart legends. Maps to the **chdl=** Google chart parameter and **must be specified according to its rules**.	No	Blank
axis	Chart axes to show. Maps to the **chxt=** Google chart parameter and **must be specified according to its rules**.	No	x
title	The title of the Google Chart.	No	Blank

Notes, Comments and Warnings

To use this extension you **must** be familiar with Google charts.

Information about Google charts can be found at: `http://code.google.com/apis/chart`

This extension provides a mechanism to build the URL parameter string to append to the Google Chart URL. In other words, given the Google Chart URL

`http://chart.apis.google.com/chart`

this extension appends the `?<parameters>` to it.

As such it relies on the functionality provided by the Google Chart API.

There are 2 ways to build the parameters based on the value of the **sourceParmString** property.

Setting the **sourceParmString** to **User String** requires the user to enter the entire string to append to the URL in the **userString** parameter.
The only property that is evaluated when choosing **User String** is **userString.**

When **sourceParmString** is set to **Table** the user must create and fill a table using the **onLoadChart** property and specify its name in the **tableName** property. The table row object must conform to this structure:

Property Name	Type	Mandatory	Description
chartData	String	Yes	The data used by Google to draw the chart. Google Parameter: **chd** **Note: this extension implements only chd=t**
dataSet	String	No	A string to identify a dataSet. Required only when the chart is made of more than one set of data.
chartLabel	String	No	A label for the chartData

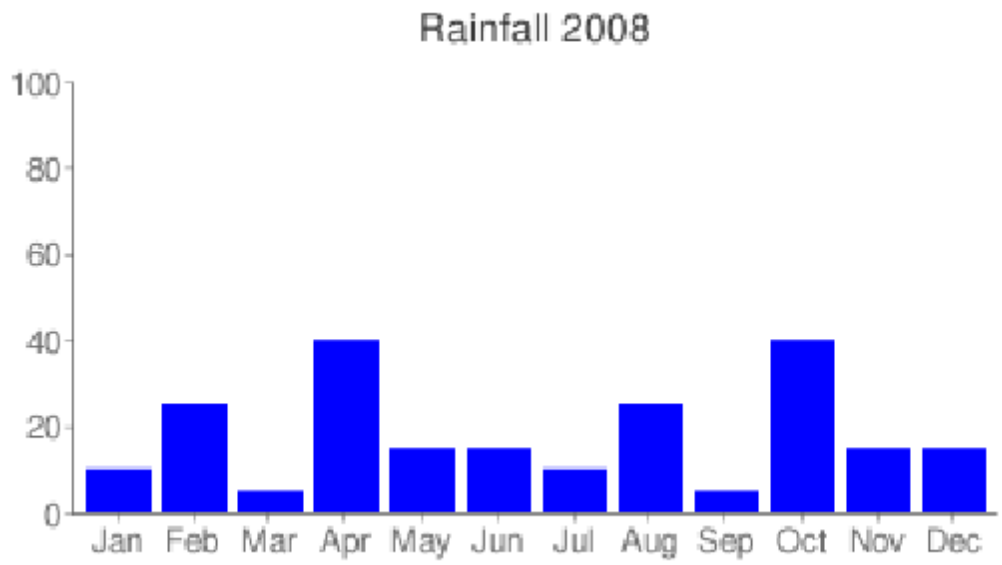
Here is some sample code to put into the onLoadChart property. This code creates a table called `Rainfall` by month …

```
var t1 = TABLEMANAGER.getTable("Rainfall");

t1.insertChild ({chartData:"10",chartLabel:"Jan"});
t1.insertChild ({chartData:"25",chartLabel:"Feb"});
t1.insertChild ({chartData:"5",chartLabel:"Mar"});
t1.insertChild ({chartData:"40",chartLabel:"Apr"});
t1.insertChild ({chartData:"15",chartLabel:"May"});
t1.insertChild ({chartData:"15",chartLabel:"Jun"});
t1.insertChild ({chartData:"10",chartLabel:"Jul"});
t1.insertChild ({chartData:"25",chartLabel:"Aug"});
t1.insertChild ({chartData:"5",chartLabel:"Sep"});
t1.insertChild ({chartData:"40",chartLabel:"Oct"});
t1.insertChild ({chartData:"15",chartLabel:"Nov"});
t1.insertChild ({chartData:"15",chartLabel:"Dec"});
```

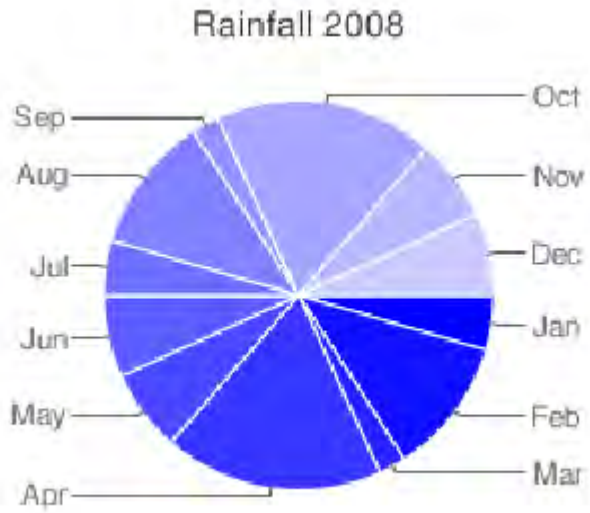
Together with these Chart properties …

tableName	Rainfall
type	bvs
colors	0000FF
axis	x,y
title	Rainfall+2008

produce this chart:



To produce a Pie chart with the same data set the Type property to **p**



For a line chart set the Type property to **lc**

Using more than one data set.

To show more than one data set you must identify the row being inserted as belonging to a specific set. To do this use the optional **dataSet** property.

We can extend the above example by adding the Rainfall for the year 2007. First we would have to add the **dataSet** to the existing 2008 data:

```
t1.insertChild ({dataSet:"A",chartData:"10",chartLabel:"Jan"});
t1.insertChild ({dataSet:"A",chartData:"25",chartLabel:"Feb"});
t1.insertChild ({dataSet:"A",chartData:"5",chartLabel:"Mar"});
t1.insertChild ({dataSet:"A",chartData:"40",chartLabel:"Apr"});
t1.insertChild ({dataSet:"A",chartData:"15",chartLabel:"May"});
t1.insertChild ({dataSet:"A",chartData:"15",chartLabel:"Jun"});
t1.insertChild ({dataSet:"A",chartData:"10",chartLabel:"Jul"});
t1.insertChild ({dataSet:"A",chartData:"25",chartLabel:"Aug"});
t1.insertChild ({dataSet:"A",chartData:"5",chartLabel:"Sep"});
t1.insertChild ({dataSet:"A",chartData:"40",chartLabel:"Oct"});
t1.insertChild ({dataSet:"A",chartData:"15",chartLabel:"Nov"});
t1.insertChild ({dataSet:"A",chartData:"15",chartLabel:"Dec"});
```

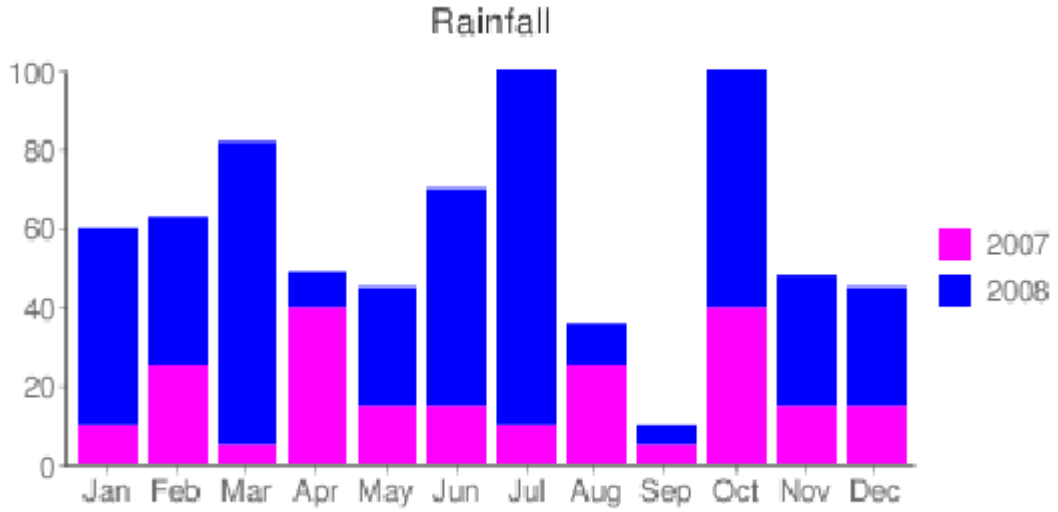
Next we add the second set using a different identifier than the "A" used above. Note also that we don't to specify chart labels:

```
t1.insertChild ({dataSet:"B",chartData:"50"});
t1.insertChild ({dataSet:"B",chartData:"38"});
t1.insertChild ({dataSet:"B",chartData:"77"});
t1.insertChild ({dataSet:"B",chartData:"9"});
t1.insertChild ({dataSet:"B",chartData:"30"});
t1.insertChild ({dataSet:"B",chartData:"55"});
t1.insertChild ({dataSet:"B",chartData:"90"});
t1.insertChild ({dataSet:"B",chartData:"11"});
t1.insertChild ({dataSet:"B",chartData:"5"});
t1.insertChild ({dataSet:"B",chartData:"66"});
t1.insertChild ({dataSet:"B",chartData:"33"});
t1.insertChild ({dataSet:"B",chartData:"30"});
```

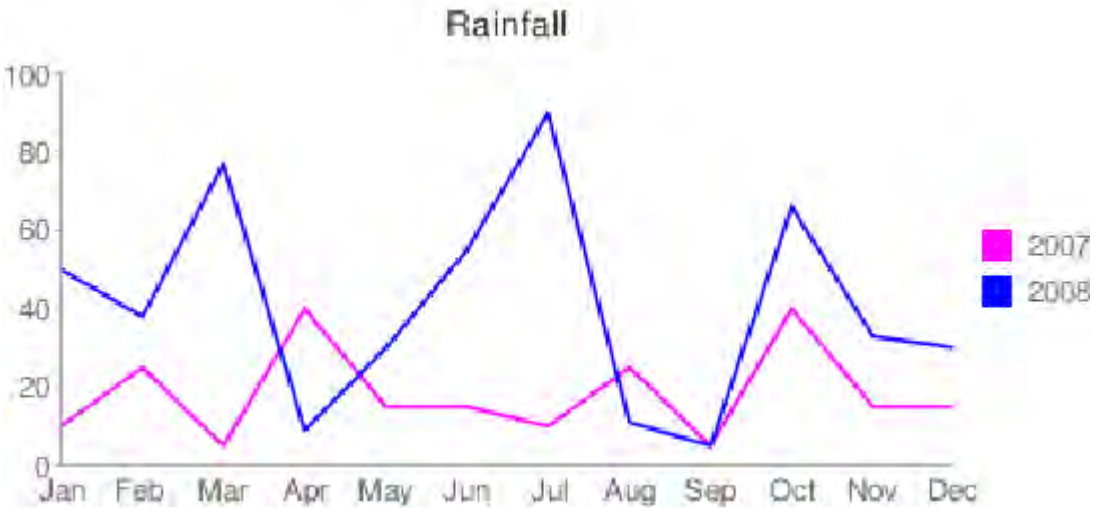
Using the **legends** property we can add a legend for the 2007 and 2008 sets, change the **title** and add a color to the **colors** property for the second set like this:

colors	FF00FF,0000FF
legends	2007|2008
title	Rainfall

Result for the Bar Chart (**type** bvs):



Result for the Line Chart (**type** lc):



Pie charts are not suitable to display more than one data set.

See Also

Example – Dynamic Google Chart 1 in Tutorial 11 - FAQ and Examples

Example – Dynamic Google Chart 2 in Tutorial 11 - FAQ and Examples

axGraphPaper – Place Graph Paper on Screen

Description

axGraphPaper draws graph paper over or under the current 5250 screen.

It is designed to assist aXes developers in laying out screens by providing a layout grid for them to work from.

axGraphPaper only functions in design mode.

Useability

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

Properties

Name	Description and Comments	JavaScript	Shipped default
lineColor	The color of the graph paper lines. Specify as a valid HTML color.	No.	DarkGray
lineStyle	The style of the graph paper lines - solid, dotted or dashed.	No.	dotted
graphSize	The size of the graph paper squares in pixels. Default is 10. Must be in range 5 through 200.	No.	10
graphStyle	The style of the graph paper lines. Specify as horizontal, vertical or both.	No.	Both
graphLocation	The location of the graph paper lines. Specify as at the front or at the back of the 5250 display.	No.	Front

Notes, Comments and Warnings

Theoretically, this eXtension should be removed from customized screens before they are deployed to production systems. No harm will come from not doing this as the graph paper eXtension will only display itself when executed in design mode.

See Also

Using Screen Graph Paper

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

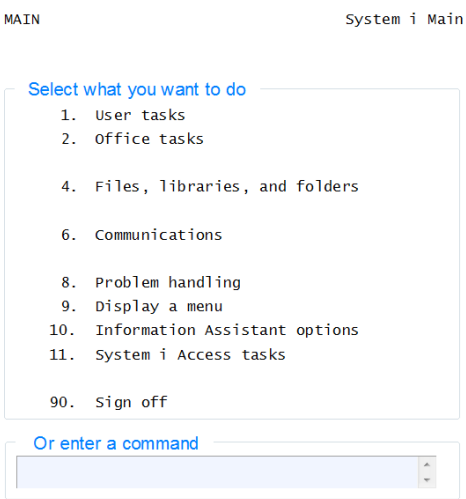
Name	Description and Comments	JavaScript	Shipped default
caption	Specifies the caption or title to be used for the group box.	Yes. Must return a string.	axGroupBox
style	Additional style properties that should be applied to the group box and its caption.	Yes. Must return a valid style object.	None.
look	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.

Notes, Comments and Warnings

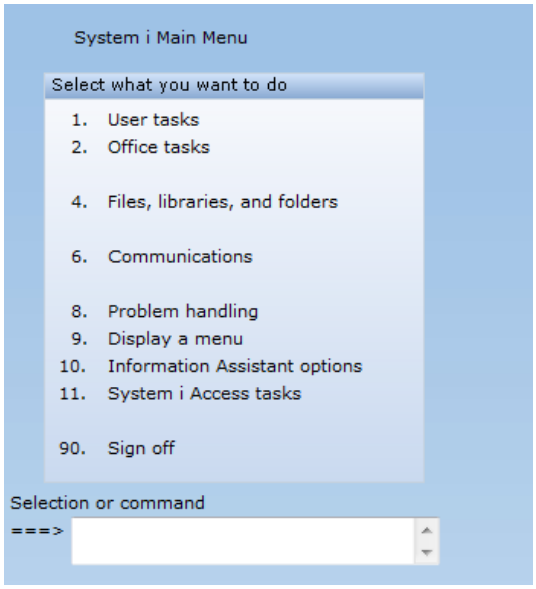
None.

Examples

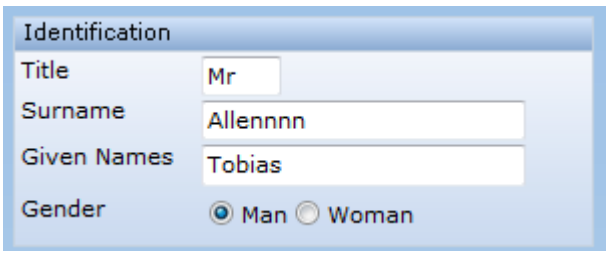
(1). This is an example of two group boxes being used to enhance the OS?400 main menu …



(2). Here is the same example using the modern group box look



(3). Here are some examples of using the modern group box look to group fields:



See Also

Simple Customization in Tutorial 2 - Basic Screen Enhancement

Group Boxes in Tutorial 3 - Advanced Screen Enhancement

Understanding Screen Sizing and the Thin Red Line in Tutorial 10 - 5250 Screen Styling.

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?

axHTML – Insert raw HTML into screen

Description

axHTML allows a raw HTML string to be inserted into a 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
html	Specifies a string or JavaScript code that defines the HTML element definition(s) to be inserted into the screen. For example: Hello or <table><tr><td>123</td></tr><tr> td>456</td></tr></table>.	Yes. Must return a string.	None.
containerStyle	Additional style properties that should be applied to the container of the HTML elements defined by the html property.	Yes. Must return a valid style object.	None.

Notes, Comments and Warnings

None.

Examples

(1). This simple string …

```
<p style="color:red;font-size:48pt;">Hello World</p>
```

should produce a display like this …

Hello
World

---

(2). This executable JavaScript …

```
var oDate   = new Date();
var sResult = "<div>Time is " + oDate.getTime().toString() + "</div>";
ENV.returnValue = sResult;
```

should produce a display like this …

Time is 1247727497119

axHyperLink – Insert a hyperlink into screen

Description

axHyperLink allows a hyperlink to be inserted into the 5250 screen.

Useability

Context	Supported
Input screen fields	Yes
Output screen fields	Yes
Input subfile fields	Yes
Output subfile fields	Yes
New screen elements	Yes

Properties

Name	Description and Comments	JavaScript	Shipped default
textBeforeLink	The text to appear before the hyperlink.	Yes. Must return a string.	Blank
caption	The text to appear in the hyperlink.	Yes. Must return a string.	"Link"
textAfterLink	The text to appear after the hyperlink.	Yes. Must return a string.	Blank
style	CSS to apply to the hyperlink.	No.	Cursor:hand
mouseOverStyle	CSS to apply to the hyperlink while the mouse is hovered over it.	No.	Blank
onClick	Event that fires when clicking on the hyperlink.	Yes .	window.open('http://www.axeslive.com/' , 'aXes')"
mouseOverColor	CSS color to apply to the hyperlink on the mouseover event. Deprecated. Use mouseOverStyle instead.	Yes. Must return a string.	Blank

Notes, Comments and Warnings

The textBeforeLink and textAfterLink properties allow you to create a link that is only a part of the whole field such as "Click here to register."

See Also

Hyperlinks in Tutorial 3 - Advanced Screen Enhancement

Example – Using a Hyperlink to select a subfile entry in Tutorial 11 - FAQ and Examples

axImage – Insert image into screen

Description

axImage allows an image 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
style	CSS to apply to the hyperlinkg.	No	Blank
imagePath	Path to locate the image file	No	`ts/screens/<private folder>`
imageName	Name of the image file	No	`Blank`
onClick	Event that fires when clicking on the image.	Yes	Blank
onMouseOver	Event that fires when mousing over the image.	No	Blank
onMouseOut	Event that fires when mouse just went out of the image.	No	Blank

See Also

Images in Tutorial 3 - Advanced Screen Enhancement

axInputboxStyle1 – Insert an HTML input element

Description

aXInputboxStyle1 allows a user to add an HTML input element which can be a password, text, file, color, email, or a number

Useability

Context	Supported
Input screen fields	Yes
Output screen fields	Yes
Input subfile fields	Yes
Output subfile fields	Yes
New screen elements	Yes

Properties

Name	Description and Comments	JavaScript	Shipped default
value	Indicates the value that a user entered.	No	Blank
style	CSS style to be applied.	Yes. Must return a valid style object.	None.
selectContentOnFocus	When set to true, the whole content will be highlighted when the element receives the focus.	Yes. Must return a Boolean.	True.
inputType	List of HTML input element that can be selected. Values are color, text, password, file, email and number.	Yes	Blank
onChange	Optional. JavaScript code to be executed when the element changes state from unchecked to checked and vice versa.	Yes.	None.
onFocus	Optional. JavaScript code to be executed when the element receives the focus.	Yes.	None.
onBlur	Optional. JavaScript code to be executed when the element loses the focus.	Yes.	None.

Notes, Comments and Warnings

Currently, for aXes v2.2.0, password is the only supported input type.

axJQueryTheme – Change existing jQuery theme

Description

axJQueryTheme allows Developers to change existing jQuery themes in TS2 using JQuery theme options available and from jQuery website.

Useability

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

Properties

Name	Description and Comments	JavaScript	Shipped default
jqueryTheme	Select from jqueryTheme available options.	No.	DefaultTheme

Notes, Comments and Warnings

None.

TS2 by default is using the jQueryUI smoothness theme. By adding this extension to customize screens, Developers can now change the jQueryUI theme using available options currently shipped with aXes. It does not limit Developer to use available options; it is also possible for the Developer to add additional themes that are available from jQueryUI websites.

How to Add additional jQueryUI theme.

Visit http://jqueryui.com/ click the themes sections, customized or download any existing theme available.

1. Extract the zip files and look for the css folder containing the theme that we will be adding to aXes.

2. Copy the folder containing the selected theme.
   example:
   eggplant

3. Add a prefix “jQtheme_” to the selected theme folder
   Example:
   jQtheme_eggplant

4. Copy the jQtheme_<theme name> folder to /ifs/axes/ts/ts2/css

5. Make sure to set the authorities of all the file inside the jQtheme_<theme name> to *PUBLIC *R authorities. See notes below

Note:
Use the WRKLNK command in your IBM i to ensure that the new file only has *PUBLIC *R authorities.

See Also

N/A

axKeypad – provides a jQuery keypad

Description

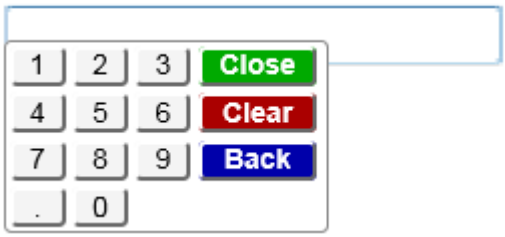
This is a text element which attaches a default jQuery keypad with a decimal character.

Properties

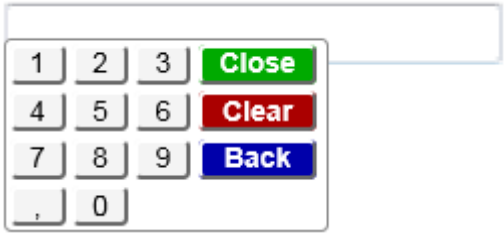
Name	Description and Comments	JavaScript	Shipped default
value	Initial element value to be displayed.	Yes. Must return a string.	None.
style	Additional style properties that should be applied to the element.	Yes. Must return a valid style object.	None.
maxInputLength	If specified, this will set the maximum number of characters that can be entered.	Yes. Must return a number.	0
onFocus	Optional. JavaScript code to be executed when the element receives the focus.	Yes.	None.
onBlur	Optional. JavaScript code to be executed when the element lost the focus.	Yes.	None.

Notes, Comments and Warnings

When language is set to non-French, the following keypad eXtension with decimal point character will be displayed:



And if set to French, the following keypad eXtension with the comma character as the decimal point will be displayed:



axLabel – Insert label into screen

Description

axLabel allows a text label 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
text	Specifies a string or JavaScript code that text to be inserted into the label.	Yes. Must return a string.	axLabel
style	Additional style properties that should be applied to the container of the label.	Yes. Must return a valid style object.	None.
onClick	Optional JavaScript code to be executed when the user clicks on the label	Yes.	None.

Notes, Comments and Warnings

None.

Examples

(1). If this string is used as **text** ….

```
Hello World
```

should produce a display like this …

Hello World

---

(2). If this **style** is added to the preceding example …

```
color:red;
```

it should produce a result like this …

Hello World

---

(3). If this **onClick** code is added style is added to the preceding example …

```
window.alert("The label was clicked");
```

then clicking on the label should produce a result like this:

(4). If the IBM i Main Menu screen has been identified, and the command entry field has been named "Command", then these axLabels are added to it:

Text	onClick
Click here for output queue QPRINT	FIELDS("Command").setValue("WRKOUTQ QPRINT"); SENDKEY("Enter");
Click here for job queue QBATCH	FIELDS("Command").setValue("WRKJOBQ QBATCH"); SENDKEY("Enter");

Then the command entry screen should look something like this …

```
MAIN                                System i Main Menu
                                                          System:      LANSA07
Select one of the following:

      1.  User tasks                          Click here for output queue QPRINT
      2.  Office tasks                        Click here for job queue QBATCH

      4.  Files, libraries, and folders

      6.  Communications
```

When you click on the labels the commands WRKOUTQ QPRINT, or WRKJOB QBATCH are entered, and the enter key is sent.

In effect these labels act as command short cuts by allowing commonly used commands to be executed with a single click. This adds value to the application by making it easier to use.

See Also

Moving Things Around and Changing Labels and Captions in Tutorial 3 - Advanced Screen Enhancement

axMessageHandler – Read and Display Messages from the 5250 Screen

Description

axMessageHandler allows messages to be read from the 5250 screen and displayed in a control. If there is more than one message, that is the more messages indicator is found on the message line, the Next Message button is enabled. Hitting the Next Message button will read and display the next message in the message handler.

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
lineNumber	The number of the line on the 5250 screen that displays messages.	No	22
moreIndicator	The list of markers that indicate that there are more messages.	No	"+"
handlerStyle	The style of the Message Handler.	No	blank
prevOnClick	The JavaScript code to execute when the previous message button is clicked.	Yes	`SET5250CURSORPOS(this.getPropertyValue("lineNumber"),1); SENDKEY("PageUp");`
nextOnClick	The JavaScript code to execute when the next message button is clicked.	Yes	`SET5250CURSORPOS(this.getPropertyValue("lineNumber"),1); SENDKEY("PageDown");`

Examples

Insert a Message Handler into the *Maintain Employee Information* screen of the aXes Demo HR system.

Step 1: Open the *Projects Home Page*. Select your project and then use the *Work as Developer* option and sign on to a 5250 session.

Step 2: Start the shipped aXes demo system by entering the commands:

ADDLIBLE AXESDEMO

CALL XHRRPGTRN

The resulting 5250 screen 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
 Exit    Add    Cancel
```

Step 3: Start editing the screen.

Step 4: Add an axMessageHandler eXtension.

This is done by first adding a new element to the screen.

Make the new element into a *Message Handler* by checking that option.

Then size and position the message handler as required.

Step 5: Select the message text on line 22 and uncheck it's Default Visualization option so that it no longer appears.

Save your changes.

The modified screen 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

F3=Exit  F6=Add  F12=Cancel
```

axPopupWindow — Window Settings

Description

This eXtension allows you to customize the position, size and style of a 5250 window. This eXtension applies to the TS2 engine only.

Properties

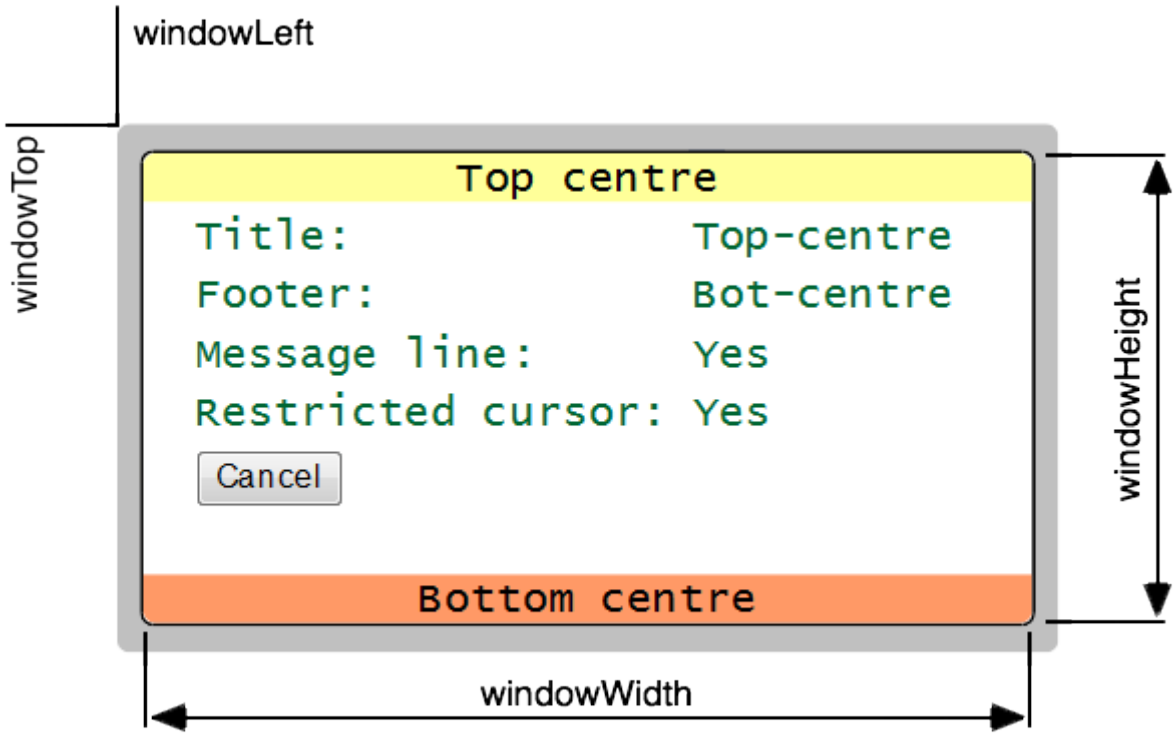
Name	Description and Comments	JavaScript	Shipped default
style	Additional style properties that should be applied to the window.	Yes. Must return a valid style object.	None.
backgroundStyle	Additional style properties that should be applied to the window.	Yes. Must return a valid style object.	None.
windowTop	The position, in pixels, of the top of the window. A negative value indicates that the position specified by the host application should be used.	Yes.	-1
windowLeft	The position, in pixels, of the left edge of the window. A negative value indicates that the position specified by the host application should be used.	Yes.	-1
windowWidth	The width of the window in pixels. A negative value indicates that the width specified by the host application should be used.	Yes.	-1
windowHeight	The height of the window in pixels. A negative value indicates that the height specified by the host application should be used.	Yes.	-1

Notes, Comments and Warnings

TS2 windows are constructed using two boxes. A gray, slightly transparent (on browsers that support it) box provides a clearly visible border that separates the window from the background. Inside it is another box that provides a solid border to the window, has a white background, and contains any headers or footers specified by the host application.

To maintain backward compatibility with TS1 windows, the contents of the window are not placed inside the background box. They are placed inside the gray window box and the background is then positioned behind them. This is important to understand when applying styles. Inherited styles like font settings applied to the background will not be inherited by the window contents.

The windowTop and windowLeft properties define the position of the top left corner of the window. The windowWidth and windowHeight properties define the size of the content area inside all borders.



Also see PopUp Windows and Screen Rendering in aXes-TS2.

axQuickPick – Insert quick pick menu into screen

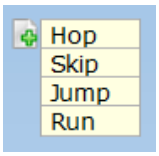
Description

axQuickPick may be used in two different ways:

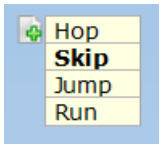
(1). The first is when the **image** property is used with a new screen element or with an existing 5250 screen element *whose default visualization has been unchecked/removed.*

axQuickPick displays an image on the 5250 screen, like this example:

As the user mouses over the image a quick pick menu appears, like this example:



As the user mouses over the menu items the font weight changes:

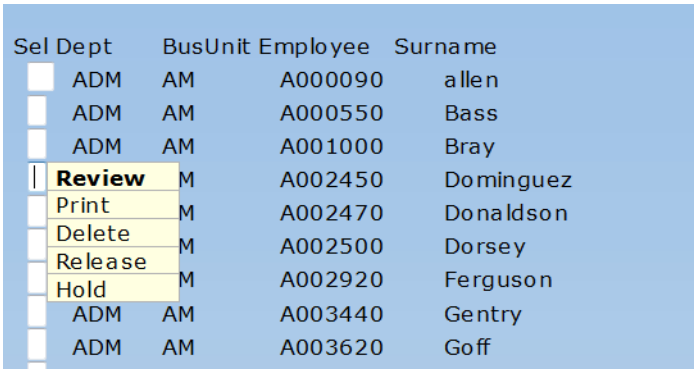


When the user clicks on a menu item an associated script is executed.

(2). The second is when the **image** property is not used - typically only with existing 5250 elements *whose default visualization has <u>not</u> been unchecked/removed*:

axQuickPick does not display an image. Instead it underlines the associated 5250 screen element's default visualization as the mouse passes over it, presenting the quick pick menu near the visualized field.

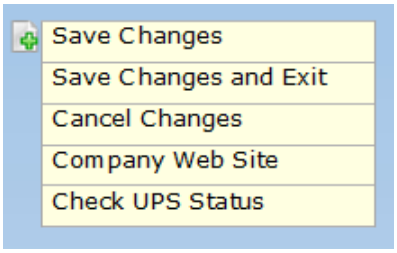
This is an example of a quick pick menu that has been associated with the "Sel" (selection) column in a subfile. The quick pick menu appears beside the selection field.



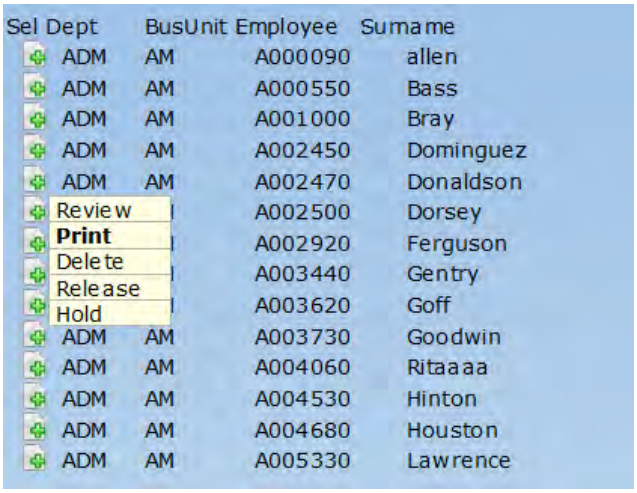
Likewise, here a quick pick menu has been associated with the Employee column in a subfile. As the user moves the mouse down the column the quick pick menu follows - indicating which employee it is currently associated with by underlining it:

Quick pick menus may be used as new screen elements in place of push buttons.

For example a quick pick menu could contain these items:



Quick pick menus may also useful for selecting items from subfiles. For example. here the selection field in a subfile is visualized as a quick pick menu indicating what action should be taken against an employee:



Useability

Context	Supported
Input screen fields	Yes
Output screen fields	Yes
Input subfile fields	Yes
Output subfile fields	Yes
New screen elements	Yes

Properties

Name	Description and Comments	JavaScript	Shipped default
image	The image to be displayed. The quick pick menu is displayed as the mouse passes over this image. Typically the image property is only used with new screen elements or with existing 5250 screen elements whose *Default Visualization* has been unchecked/removed. Typically the image property is not used (ie: blanked out/removed) with existing 5250 screen elements whose *Default Visualization* has not been removed.	Yes. Must return a string.	axquickpick.gif
imageSize	The size of the image that is to be displayed. Choose from 16x16, 32x32 or 64x64.	Yes. Must return "S" (16x16), "M" (32x32) or "L" (32x32).	16px x 16px
items	The captions of the item(s) that are to appear on the quick pick menu.	No.	None.
itemStyle	The style of the item(s) on the quick pick menu. The height must be specified, be in pixels (px) and in the range 5 to 500. The width must be specified, be in pixels (px) and in the range 5 to 500.	Yes. Must return a style object.	height:15px width:80px background:infobackground, border-bottom:1px solid activeborder padding-left:4px
containerStyle	The style of the container (DIV) of the quick pick menu items.	Yes. Must return a style object.	cursor:pointer background:infobackground border-top:1px solid activeborder border-left:1px solid

			activeborder, border-right:1px solid activeborder
containerLocation	The displayed location of the quick pick menu relative to the image. Chose from Above and Left, Above and Right, Below and Left or Below and Right.	Yes. Must return on o f the strings "AL", AR", "BL" or "BR".	Below and Right.
containerVOffset	Additional specific vertical offset to be applied to the quick pick menu container. Default is zero. Specify as an integer in the range -256 to +256. Value specifies the number of pixels.	Yes. Must return an integer.	0
containerHOffset	Additional specific horizontal offset to be applied to the quick pick menu container. Default is zero. Specify as an integer in the range -256 to +256. Value specifies the number of pixels.	Yes. Must return an integer.	O
onItemSelection	The script to be executed when the user clicks on a quick pick menu item. Special scripting values ENV.itemNumber and ENV.itemCaption are accessible to this script. They may be used to identify which menu item was clicked.	Yes.	None/

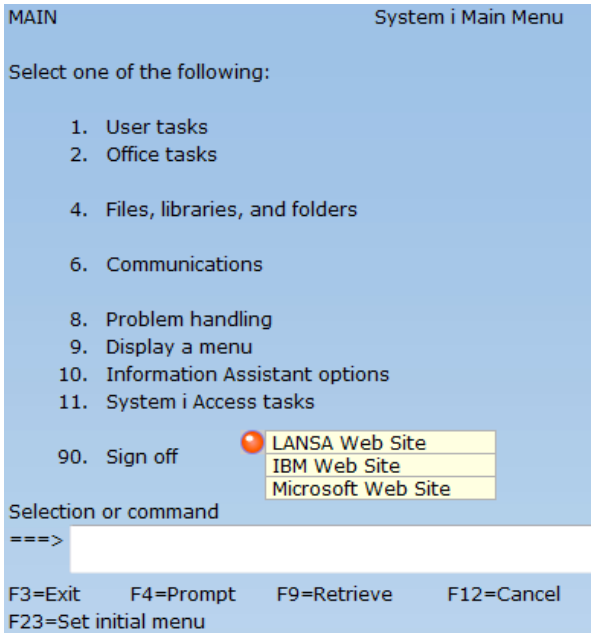
Notes, Comments and Warnings

None.

Examples

(1). Used as way to a launch other web sites

A quick menu with an orange ball image is added to the IBM i Main Menu:



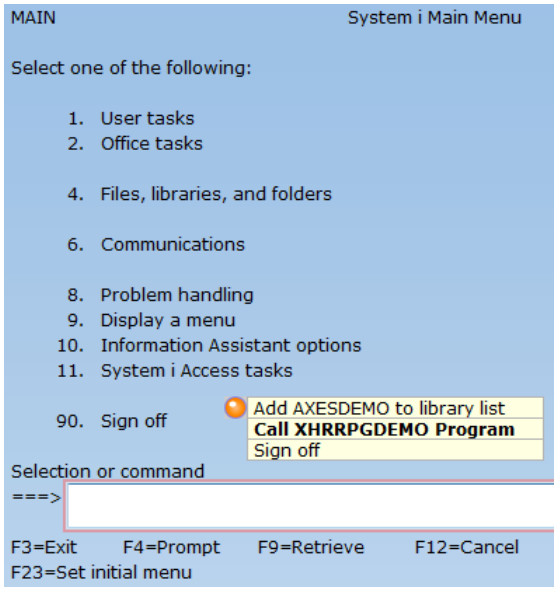
The **onItemSelection** script allows the LANSA, IBM and Microsoft web sites to be viewed in another window:

```
switch (ENV.itemNumber)
{
   case 1:
        window.open("http://www.lansa.com","_blank");
        break;
   case 2:
        window.open("http://www.ibm.com","_blank");
        break;
   default:
        window.open("http://www.microsoft.com","_blank");
        break;
}
```

---

(2). Used as way to minimize keystrokes for common actions

The preceding example is changed to:



and the **onItemSelection** script was changed to:

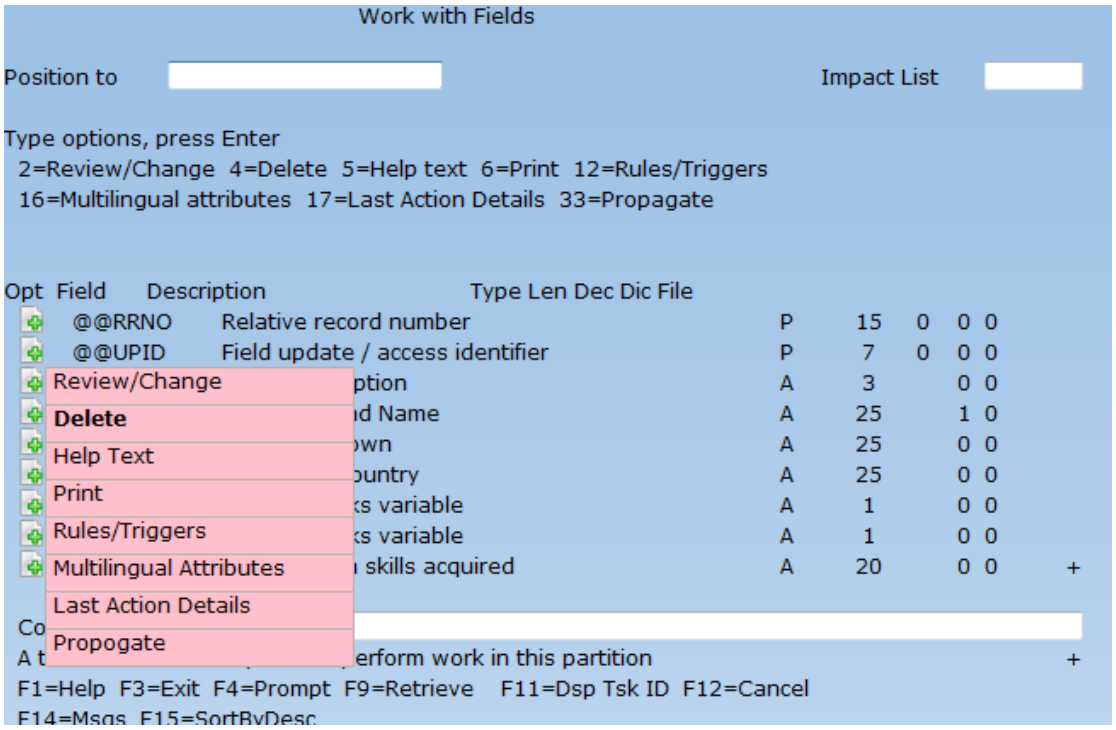
```
var command = "";

switch (ENV.itemNumber)
{
   case 1:
        command = "ADDLIBLE AXESDEMO";
        break;
   case 2:
        command = "CALL XHRRPGTRN";
        break;
   default:
        command = "SIGNOFF";
        break;
}

FIELDS("CommandLine").setValue(command);
SENDKEY("Enter");
```

---

(3). Used to select subfile entries

Here a classic 5250 "Work With" style screen has an option column (titled "Opt") in which action values can be entered.

The subfile option column was named *SFLOption* and replaced by the axQuickPick eXtension:



The **onItemSelection** script was set to:

```
var sOption = "";

/* Set the correct subfile value for the menu item number */

switch (ENV.itemNumber)
{
   case 1: sOption = "2"; break;
   case 2: sOption = "4"; break;
   case 3: sOption = "5"; break;
   case 4: sOption = "6"; break;
   case 5: sOption = "12"; break;
   case 6: sOption = "16"; break;
   case 7: sOption = "17"; break;
   case 8: sOption = "33"; break;
   default: break;
}

/* If an option was set, put into the 5250 subfile field */
/* and press the enter key                               */

if (sOption != "")
{
   FIELD.setValue(sOption);
   SENDKEY("Enter");
}
```

See Also

Dynamically Changing Themes in Tutorial 10 - 5250 Screen Styling

axRadioButton – Insert radio buttons into screen

Description

axRadioButton allows the creation of a set of radio buttons.

Useability

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

Properties

Name	Description and Comments	JavaScript	Shipped default
radioButtonStyle	CSS to apply to the radio buttons	No	Blank
captionStyle	CSS to apply to the text of the radio buttons	No	Blank
orientation	Layout the radio buttons horizontally (**orientation = H**) or vertically (**orientation = V**)	No	V (vertical)
dataSourceType	Indicate where the source data is obtained from.	No	fixedValues
tableName	For **Static Table** data source type, this must be the value set for the name = property. For **XML File** or **Fixed values**, specify a name to identify the data set so that the table is filled only once per session. For **Dynamic Table** this property is ignored and sqlQueryName is used instead.	No	Blank
onCreateRadioButton	JavaScript code to execute to set the radio button text.	Yes	ROW.text
xmlFileName	Name of XML file when the source type is XML.	No	Blank
xmlFileLocation	Path to locate the xmlFileName when the source type is XML.	No	*DEFAULT which is equal to /ts/screens/<private folder>
fixedValues	For **Fixed values** data source type you must enter the value pairs used to fill the radio button	No	No
onSelectRadio	Determines the radio button entry to select.	Yes	ROW.value
onRadioButtonClicked	Event fired when clicking on a Radio Button	Yes	FIELD.setValue(ROW.value)
sqlQueryName	For **Dynamic Table** data source type the value set for the name = property	No	Blank
sqlVariables	For **Dynamic Table** data source the variables used in your query.		ENV.SQL.SQLVariableLibrary = USERENV.dftSQLDataLibrary
keepLastKey	For **Dynamic Table** data source. If this option is true, a dynamic table will only be	No	false

	reloaded if the key changes.		
onChange	Optional. JavaScript code to be executed when the element changes state from unchecked to checked and vice versa.	Yes.	None.
onFocus	Optional. JavaScript code to be executed when the element receives the focus.	Yes.	None.
onBlur	Optional. JavaScript code to be executed when the element lost the focus.	Yes.	None.

Notes, Comments and Warnings

The *dataSourceType* property determines the method used to populate the radio button.

Except for *Dynamic Table* you should specify a *tableName.*

Fixed Values: use this setting to hard code the value and text of the radio buttons.

Static Table: static tables are loaded once from the server at start up. The name of the static table must be the name specified in *tableName.*

XML File: the data to fill the table is obtained from an XML file located by default in /ts/screens/<private folder>. To specify another location change the *xmlFileLocation* property.

Dynamic Table: use this option carefully as it is the only one where the data is loaded at every screen arrival with an expected performance impact in the rendering of the screen. When you use this option you must specify the name of the query in *sqlQueryName* as defined in your dynamic tables file. You can use the keepLastKey option to improve performance, if multiple requests using the same query key are commonly made.

What is a ROW.

The radio button makes use of the aXes table structures. It means that the data to create the radio button is stored in memory in a table-like structure consisting of rows and columns. For example, consider a table defined like this:

```
DefineObjectInstance { className = "StaticTable",

  name    = "Gender",
  source  = "inline",

  rows = {
               {value="M",text="Male"},
               {value="F",text="Female"},
            },
  };
```

(Note that for such a table you would most likely set the *dataSourceType* to *Fixed Values* and type in the values manually).

At start up aXes populates the tables with one called *Gender*. It will have 2 rows, each **ROW** consisting of columns *value* and *text.* Using the Gender table, *ROW.value* would yield the current row's *value*= value and *ROW.text* the current row's *text*= value.

There is no constraint on how you name the row columns. If you use other names you must remember to change the default values to reflect the column names.

For example, if your file or fixed values define your columns like this:

```
  rows = {
                {codigo="M",descripcion="Masculino"},
                {codigo="F",descripcion="Femenino"},
            },
  };
```

The default values or script code should refer to *ROW.codigo* and *ROW.descripcion*.

***onCreateRadioButton* event**:

This event creates the radio button and sets its displayable text. It will fire (following our table example) whenever a screen (*any screen)* that has a radio button whose *tableName* was set to *Gender* arrives.

It will locate the *Gender* table and iterate through all its entries.

For each entry it will create a radio button using the value of the *text* column of the Gender table as the label. The result in our example will be two radio buttons: one with caption Male and the other with caption Female.

This is the simplest way to use *onCreateRadioButton.*

A more powerful way would be to create the radio buttons with a *value* resulting from the evaluation of a more complex expression.

If we extended the *Gender* table like this:

```
DefineObjectInstance { className = "StaticTable",

  name    = "Gender",
  source  = "inline",

  rows = {
                {value="M1",text="Male", agerange="16-25"},
                {value="M2",text="Male", agerange="26-35"},
                {value="M3",text="Male", agerange="36-45"},
                {value="F1",text="Female", agerange="16-25"},
                {value="F2",text="Female", agerange="26-35"},
                {value="F3",text="Female", agerange="36-45"},
            },
  };
```

We could change *onCreateRadioButton* to say:

ROW.text + "s aged " + ROW.agerange

which would create this set of radio buttons:

Males aged 16-25
Males aged 26-35
Males aged 36-45
Females aged 16-25
Females aged 26-35
Females aged 36-45

onRadioButtonClicked

***onSelectRadio* event**:

The code in this event determines which radio button to set as the selected (checked). It can be something simple like the default value:

ROW.value

If the expression does not return a value of type *Boolean,* the default logic of comparing the value is returned with the associated field's value.

An example of an expression returning a value of type *Boolean* would be:

(ROW.value == "Hello")

***onRadioButtonClicked* event**:

This event fires when a radio button is clicked. The default behaviour here is

FIELD.setValue(ROW.value)

In other words, set the value of the field (which is referred to as FIELD) associated with the radio button to the value of the selected ROW.

See Also

Radio Buttons in Tutorial 3 - Advanced Screen Enhancement

axScreenBasicProperties - Screen Level Basic Properties

Description

Screen level basic properties apply to an entire screen.

Properties

Name	Description and Comments	JavaScript	Shipped default
name	The name of the screen	No	None.
style	Style elements that should be applied the body of the screen.	Yes. Must return a valid style object.	None.
useTerminalStyles	Indicates whether the normal aXes CSS style sheets should be applied to this screen.	No.	False.
firstElementToFocus	Indicates which element on the screen will get the focus when the screen arrives: Options: Follow the original 5250 screen. Set focus on element with the lowest tab index.	Yes	Follow original 5250 screen
limitedCursorSupport	aXes emulates a 5250 screen cursor. When a screen is customized by eXtensions most of the time cursor use does not make any sense anymore as the fields would have been moved around. There might be occasions where only a very minor modifications are made to the screen and cursor is essential for the operation of that screen. In this case limitedCursorSupport can be turned on (set to true). The cursor support is limited in a sense that: • The cursor positioning is based on the original position of the 5250 elements, so if you move the element the cursor will not have any awareness of it. • The cursor will always be visible even when it is in an edit box.	No.	False.
disableTerminalKeys	Indicates whether terminal keys (for example function keys F1 – F24) should be disabled. When terminal keys are disabled, physical keys typed in by users are ignored by AXES. Keys submitted programmatically using SENDKEY function will still be processed by AXES.	No.	False.
disableAutoGUI	Indicates whether AutoGUI is enabled for the screen. This property is only valid for screens generated with the aXes-TS2 rendering engine. AutoGUI is always disabled for customized screens in aXes-TS.	No.	False.
keepPopupLocation	Where a customized 5250 screen is a 5250 pop-up window you may use this property to maintain its 5250 location (if set to true) or to cause it to be moved to the top left of the display area (if set to false). This does not apply to the aXes-TS2 rendering engine. The user can drag the window out of the way instead.	No.	True.
keepLegacy5250FFW	Indicates whether to apply the legacy 5250 Field Format Word to input elements so that they emulate the behaviour of a 5250 terminal.	Yes.	False.
onArrive	JavaScript code that is to be executed every time the 5250 screen arrives.	Yes.	None.

onLeave	JavaScript code that is to be executed every time the 5250 screen departs. ENV members: • **ENV.key**: indicates what key triggered the event. Please refer to the key symbolic names table in the examples section. Return value: Set to false to prevent the screen from leaving.	Yes.	None.

Notes, Comments and Warnings

You should not use alerts and any other operation that halts or interrupts the execution of JavaScript code during the arrival or departure of a screen.

Departure code may not be executed in an abnormal termination.

Examples

(1). Specify this code in the **onArrive** property:

```
TRACE("**** MY ONARRIVE CODE IS EXECUTING. ****");
```

Specify this code in the **onLeave** property:

```
TRACE("**** MY ONLEAVE CODE IS EXECUTING. ****");
```

Now execute your aXes application with trace mode turned on.

You should see trace statements indicating that your code has been executed.

(2). Disabling certain function keys

Specify this code in the **onLeave** property to disable the F1 & F4 keys:

```
if(ENV.key == "F4" || ENV.key == "F1")
{
    ENV.returnValue = false;
}
```

The following table lists the symbolic names for the keys (the symbolic names are the values that are to be used for comparison with ENV.key):

Key	ENV.key value
F1 – F24	F1 – F24
Enter	ENTER
Roll Down (Page Up)	ROLLDOWN
Roll Up (Page Down)	ROLLUP
Roll Left	ROLLLEFT
Roll Right	ROLLRIGHT
Clear Screen	CLEARSCREEN
PA1	PA1
PA2	PA2
PA3	PA3
Help	HELP
Print	PRINT

See Also

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? in Tutorial 11 - FAQ and Examples

axScreenDataViewer – Display data from another screen

Description

Use of this eXtension enables the display of individual field data or a subfile data from one 5250 screen to another screen.

Properties

Name	Description and Comments	JavaScript	Shipped default
caption	Specifies the text that is to appear at the top of the element. Applicable only when the reference is a subfile and not an individual field.	Yes. Must return a string.	None.
referenceName	Refers to the logical name of the reference data.	Yes. Must return a string.	None.
containerStyle	Additional style properties that should be applied to the element. Overrides values whenever there is a previous setting.	Yes. Must return a valid style object.	None.
onFocus	Optional. JavaScript code to be executed when the element receives the focus.	Yes.	None.
onBlur	Optional. JavaScript code to be executed when the element lost the focus.	Yes.	None.

Notes, Comments and Warnings

There are two user scripts needed to successfully display data from one screen to another screen:

1) SAVESCREENDATA and
2) DISPLAYSCREENDATA.

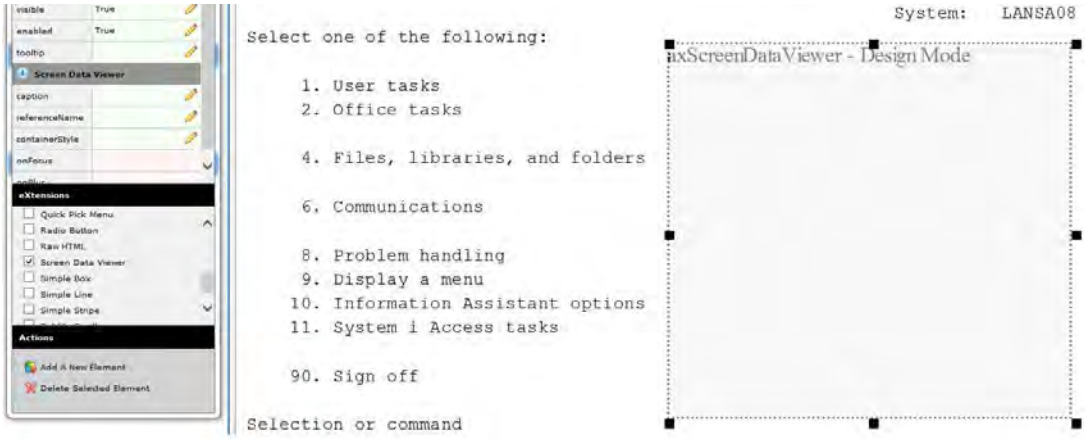
SAVESCREENDATA(<fieldName>);
where <fieldName> is the name of a reference field assigned by the user.

SAVESCREENDATA(<referenceName>, <rowCount>, <colName1>, … <colNameN>);
<referenceName> - the logical name of a subfile assigned by the user.
<rowCount> - the number of subfile rows to be displayed.
<columnName> - the name of the subfile column assigned by the user.

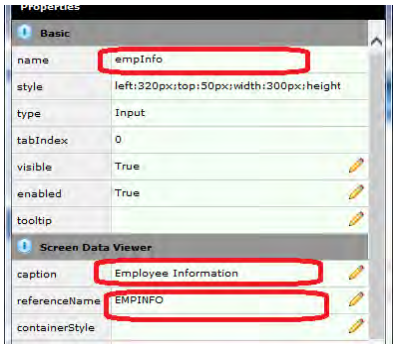
DISPLAYSCREENDATA(<element name>);
<element name> - the name of the Screen Data Viewer element assigned by the user.

Examples

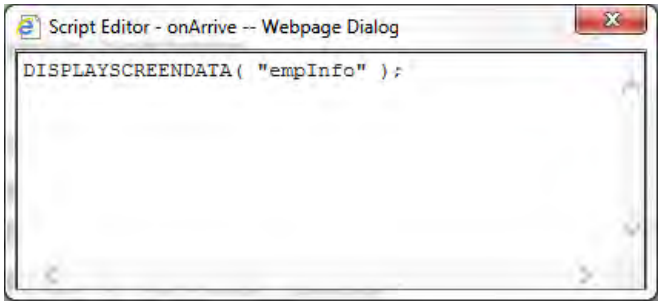
(1). Add a Screen Data Viewer element to a primary screen.



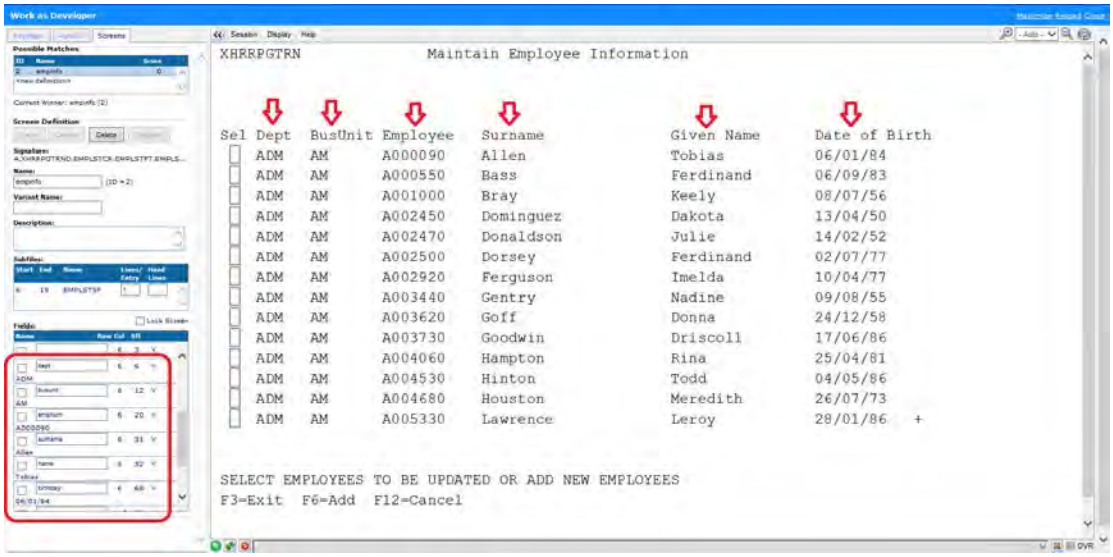
(2). Define element properties.



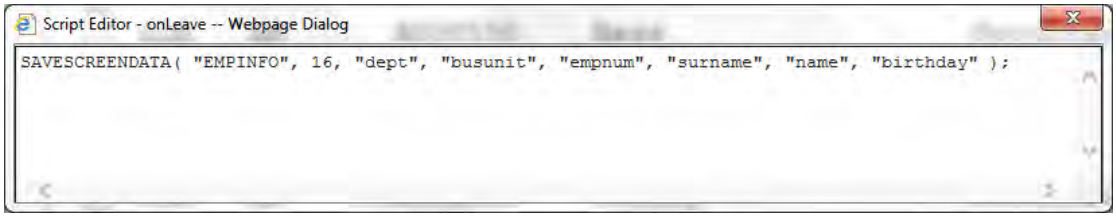
(3). Put the DISPLAYSCREENDATA script onto the onArrive dialog.



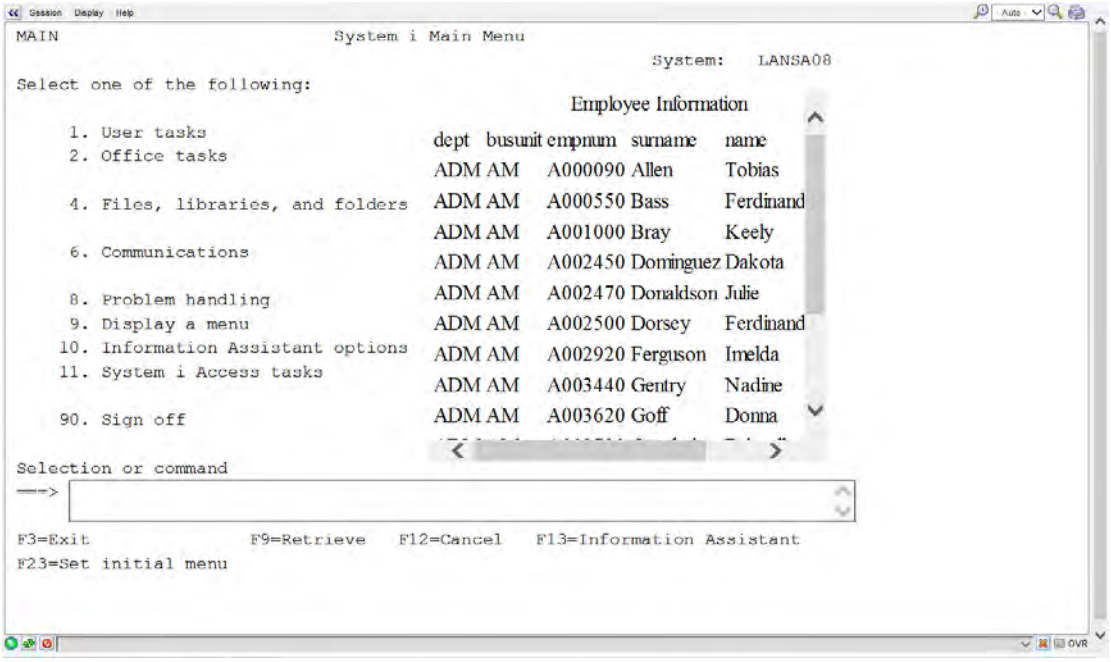
(4). Go to the screen where the subfile resides. Mark the columns to be referenced.



(5). Put the SAVESCREENDATA script onto the onLeave dialog.



(6). Go back to the primary screen and see the result.



axScreenSize - Auto Zoom Screen Size

Description

The Auto Zoom feature in TS1 calculates the zoom amount based on the default screen size (which is derived from the 5250 terminal size which is normally 80 rows x 24 columns). When larger real estate area is required for a screen, it is necessary to specify the width and height of the screen to help the layout engine calculate the correct zoom. This eXtension allows you to define this size.

The TS2 engine calculates the zoom based on the actual contents of the screen. So it may not be necessary to specify a size. But, if you want to override the default calculations for a particular effect, or to force consistency with TS1 this eXtension will allow you to override the calculated height and width.

In TS2, this eXtension will also allow you to override the zoom setting used by the layout engine.

Properties

Name	Description and Comments	JavaScript	Shipped default
width	Width of the screen (in pixels). If 0 the default screen width will be used.	Yes.	0
height	Height of the screen (in pixels). If 0 the default screen height will be used.	Yes.	0
zoom	Sets the zoom value for the current screen. Allowable values are: -2: Fit. This causes the screen to automatically zoom to fit the window. This is a one-off operation. The zoom will not adjust when the window is resized. -1: Auto. This causes the screen to automatically zoom to fit the window. The zoom will continue to adjust when the window is resized. 0: Do nothing. This leaves the zoom setting untouched. 5-200: Fixed value. This sets the zoom to the specified percentage. This is available in TS2 only.	Yes.	0
lockZoom	Locks the zoom control in the user interface, preventing the user from changing it. This is available in TS2 only.	Yes.	False

See Also

Understanding Screen Sizing and the Thin Red Line in Tutorial 10 - 5250 Screen Styling

axSimpleBox – Insert a simple box into screen

Description

axSimpleBox allows a simple 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
style	Additional style properties that should be applied to the container of the label.	Yes. Must return a valid style object.	None.

Notes, Comments and Warnings

None.

Examples

(1). If this is used as **style** and the box is sized and located ….

```
border:1px solid red;background:buttonface;
```

it should produce a display like this when used on 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
```

---

(1). If this is used as **style** and the box is sized and located ….

```
background:orange;
```

it should produce a display like this on 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

      6.  Communications

      8.  Problem handling
      9.  Display a menu
```

---

axSimpleLine – Insert a simple line into screen

Description

axSimpleLine allows a simple line 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
orientation	The intended orientation of the line. Specify as horizontal or vertical.	Yes. Must return "H" or "V".	Horizontal
lineStyle	The style of the line. Specify as dotted, dashed, solid or double.	Yes. Must return "dotted", "dashed", "solid" or "double".	Solid
lineColor	The color of the line. Must be a valid HTML color. For example: Green or #FFF5F6.	Yes. Must return a valid color.	Black
lineThickness	The thickness of the line in pixels. Must be specified as Npx where N is in the range 1 to 256.	Yes. Must return a valid thickness string in format "Npx".	1px

Notes, Comments and Warnings

None.

Examples

(1). If these properties are used and the line is sized and located ….

orientation	Horizontal
lineStyle	Solid
lineColor	Green
lineThickness	2px

it could produce a display like this on the IBM 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
```

---

(2). If these properties are used and the line is sized and located ….

orientation	Vertical
lineStyle	Dashed
lineColor	Red
lineThickness	8px

---

it could produce a display like this on 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

      6.  Communications

      8.  Problem handling
      9.  Display a menu
     10.  Information Assistant options
     11.  System i Access tasks

     90.  Sign off

Selection or command
```

axSimpleStripe – Insert a simple stripe into screen

Description

axSimpleLine allows a simple stripe to be inserted into the 5250 screen. Stripes are intended to be used to enhance the appearance of 5250 screen bodies, subfile headings and subfile bodies.

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
context	The context of the stripe. Specify as screen panel, subfile heading, subfile body, mini-panel heading or mini-panel body. This determines the way that the stripe will be colored and bordered. See notes following regarding how colors and borders are determined.	Yes. Must return "PH" (Screen Panel), "SH" (Subfile heading) or "SB" (subfile body).	Screen panel
orientation	The orientation of any color gradient that may be applied to the stripe. Specify as horizontal or vertical.	Yes. Must return "H" or "V".	Vertical

Notes, Comments and Warnings

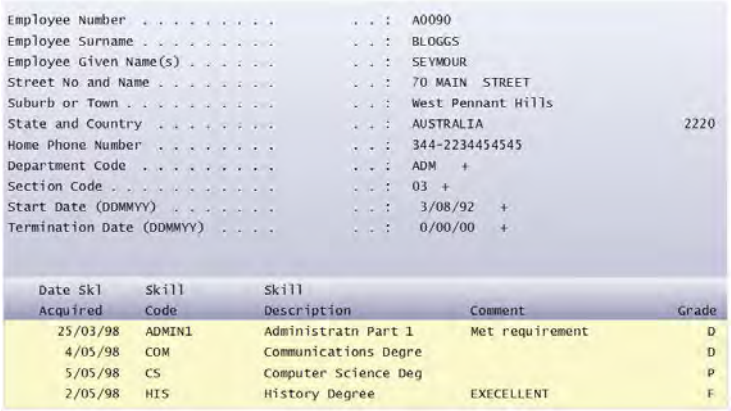
When the selection of the context is made, this in effect chooses a scheme for producing the stripe. The scheme is defined in the USERENV object by the aSimplestripeStyles object array. This array defines the start and end colors and the border used by the stripe. By altering your USERENV object's definition you can change the way that all stripes are visualized.

Examples

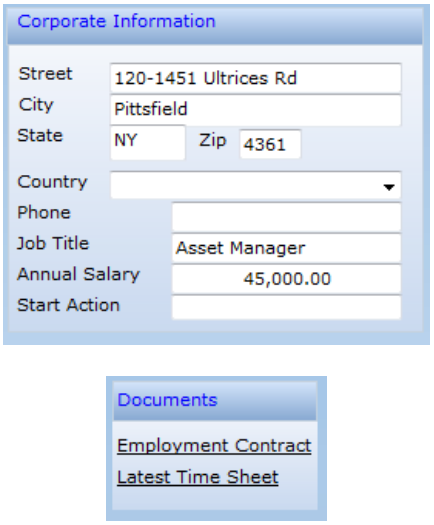
(1). By using 3 stripes this 5250 screen can be altered from this …

```
Employee Number . . . . . . . . .        . . :  A0090
Employee Surname . . . . . . . . .       . . :  BLOGGS
Employee Given Name(s) . . . . . .       . . :  SEYMOUR
Street No and Name . . . . . . . .       . . :  70 MAIN  STREET
Suburb or Town . . . . . . . . . .       . . :  West Pennant Hills
State and Country . . . . . . . .        . . :  AUSTRALIA                    2220
Home Phone Number . . . . . . . .        . . :  344-2234454545
Department Code . . . . . . . . .        . . :  ADM   +
Section Code . . . . . . . . . . .       . . :  03  +
Start Date (DDMMYY) . . . . . . .        . . :   3/08/92   +
Termination Date (DDMMYY) . . . .        . . :   0/00/00   +

  Date Skl     Skill        Skill
  Acquired      Code        Description            Comment                Grade
    25/03/98   ADMIN1       Administratn Part 1    Met requirement          D
     4/05/98   COM          Communications Degre                            D
     5/05/98   CS           Computer Science Deg                            P
     2/05/98   HIS           History Degree        EXECELLENT               F
```

to this:



(2). Here a mini-panel heading and mini-panel bodies have been used to visually group sets of related fields together ….



See Also

Adding a Stripe in Tutorial 3 - Advanced Screen Enhancement

axSpinEdit – Insert a Spin Edit

Description

Use this extension to add a spin edit box into a 5250 screen. The end-user can use this extension to change a field value by scrolling a list of enumerated values. This is an extended Dropdown element.

Useability

Context	Supported
Input screen fields	Yes
Output screen fields	Yes
Input subfile fields	Yes
Output subfile fields	Yes
New screen elements	Yes

Properties

Name	Description and Comments	JavaScript	Shipped default
dropDownStyle	CSS to apply to the element	No	Blank
dataSourceType	Indicates where the source data is obtained from.	No	fixedValues
visibleLines	Indicates the number of visible lines. When a selected value is greater than 1, the element acts as a spin edit element. When the value is 1, it will behave as a drop down element.	No	2
multipleSelect	Allows selection of multiple data or single data.	No	false
lowerRange	The lowest value in the element list.	No	Blank
upperRange	The highest value in the element list.	No	Blank
tableName	For **Static Table** data source type, this must be the value set for the name = property.	No	Blank
onFillDropDown	Script codes to execute during the filling up of element list.	Yes	Blank (i.e. no script to execute – interpreted as ROW.text)
xmlFileName	Name of XML file when the source type is XML.	No	Blank
xmlFileLocation	Path to locate the xmlFileName when the source type is XML.	No	*DEFAULT which is equal to /ts/screens/<private folder>
fixedValues	For **Fixed values** data source type, you must enter the pair of data values that will appear on the element list.	No	No
additionalEntries	Use this property to add one or more entries to the spin edit element. Note that if you use this option with static tables, dynamic tables or XML documents, the values you are adding are inserted into the underpinning table – which subsequently causes them to appear element list. This may impact other eXtensions or scripting that also use the same underpinning table.	Yes	No

onSelectValue	Data to be displayed as a result of the script code execution during the selection of value.	Yes	Blank (i.e. no script to execute – interpreted as ROW.value)
onSelectedValueChanged	Script code to execute when the current value has changed.	Yes	FIELD.setValue(ROW.value)
sqlQueryName	For **Dynamic Table** data source type the value set for the name = property	No	Blank
sqlVariables	For **Dynamic Table** data source the variables used in your query.		ENV.SQL.SQLVariableLibrary = USERENV.dftSQLDataLibrary

Notes, Comments and Warnings

***dataSourceType*:**

Determines the method used to populate the element. The options are:

Fixed Values: use this setting for a small number of values and the values don't change.

Static Table: static tables are loaded once from the server at start up. The name of the static table must be the name specified in *tableName.*

Static tables are very efficient and they can contain hard coded values or can be loaded by executing SQL commands.

XML File: the data to fill the table is obtained from an XML file located by default in /ts/screens/<private folder>. To specify another location change the *xmlFileLocation* property.

Dynamic Table: use this option carefully as it is the only one where the data is loaded at every screen arrival with an expected performance impact in the rendering of the screen. When you use this option you must specify the name of the query in *sqlQueryName* as defined in your dynamic tables file.

This value indicates that the data should come from a dynamically created table.

- A dynamic table is (re)loaded every time a screen is displayed.
- Dynamic tables are expensive to use compared to static tables.
- Only use them when the table content needs to vary according to some dynamic criteria (eg: current company, current product, etc).
- Dynamic tables can only be loaded by SQL commands.

You can use the keepLastKey option to improve performance, if multiple requests using the same query key are commonly made.

Should I use a Static or a Dynamic Table?
You do not have to use a dynamic table just because you want to execute an SQL command to load the table. You can load static tables from SQL commands as well.
You can use the highly efficient static tables in ways you may not know about.
Imagine a scenario where a currency table's content needs to change according to a company number (ie: You can only use certain currencies for certain companies).
Your immediate reaction may be to just use a dynamic table - passing the company number as an SQL variable to control which currencies are selected by the SQL command that is executed to fill the dynamic table.

This will work, but compared to using a static table this approach is expensive to execute.
You can improve this approach in several ways by using static tables instead:
Option 1: Create a single Static Table named Currency:
It would contain all the currency details for all the companies - and the associated company number as an extra column. When filling the element list, you can use scripting in the *onFillDropDown* property to omit table entries that are not valid for the current company by returning ENV.returnValue as null (meaning no visible entry is to be added to the element list).
Option 2: Create multiple Static Tables named Currency_CC:
Where _CC is the company number (say).
Script the element's *tableName* property to dynamically change.
You are using static tables - but now you are dynamically changing the table name - instead of the table content.
Note: This approach also works well with XML sourced tables. For example, you could also create an RPG program that runs every night (say).
It would read corporate "static" data files and flatten them out into a highly structured series of XML documents - where the name of the document (ie: table) is meaningful.
Option 3: Use start up or log on logic to load the "correct" set of static table data
When there is a clear point in the application where the user logs on and selects a company to work with - you can script an onArrive or onLeave events to force all static tables to load or reload.
Where they are loaded from SQL you can pass variables to the SQL executed to load them (eg: the selected company number).
Now the static table Currency would only ever contain the currencies for the currently selected company.
All the "users" of the Currency static table - like drop downs - no longer have to worry about the content - it will always be correct for the currently selected company.

What is a ROW

The element list makes use of the aXes table structures. It means that the data to populate the list is stored in memory in a table-like structure consisting of rows and columns. For example, consider a table defined like this:

```
DefineObjectInstance { className = "StaticTable",

  name    = "Gender",
  source  = "inline",

  rows = {
                {value="M",text="Male"},
                {value="F",text="Female"},
             },
  };
```

(Note that for such a table you would most likely set the *dataSourceType* to *Fixed Values* and type in the values manually).

At start up aXes populates the tables with one called *Gender*. It will have 2 rows, each **ROW** consisting of columns *value* and *text.* Using the Gender table, *ROW.value* would yield the current row's *value*= value and *ROW.text* the current row's *text*= value.

There is no constraint on how you name the row columns. If you use other names you must remember to change the default values to reflect the column names.

For example, if your file or fixed values define your columns like this:

```
  rows = {
                {codigo="M",descripcion="Masculino"},
                {codigo="F",descripcion="Femenino"},
            },
  };
```

The default values or script code should refer to *ROW.codigo* and *ROW.descripcion*.

***onFillDropDown* event**:

This event populates the element list. It will fire (following our table example) whenever a screen (*any screen)* that has the element whose *tableName* was set to *Gender* arrives.

It will locate the *Gender* table and iterate through all its entries.

For each entry it will add an entry where the visible text in the list is equal to the value in *text* column of the Gender table. The result in our example will be a spin edit element with 2 entries: Male and Female.

This is the simplest way to use *onFillDropDown.*

A more powerful way would be to fill the list with a *value* resulting from the evaluation of a more complex expression.

If we extended the *Gender* table like this:

```
DefineObjectInstance { className = "StaticTable",

  name    = "Gender",
  source  = "inline",

  rows = {
                {value="M1",text="Male", agerange="16-25"},
                {value="M2",text="Male", agerange="26-35"},
                {value="M3",text="Male", agerange="36-45"},
                {value="F1",text="Female", agerange="16-25"},
                {value="F2",text="Female", agerange="26-35"},
                {value="F3",text="Female", agerange="36-45"},
            },
  };
```

We could change *onFillDropDown* to say:

ROW.text + "s aged " + ROW.agerange

The element list would end up with entries that look like this:

Males aged 16-25
Males aged 26-35
Males aged 36-45
Females aged 16-25
Females aged 26-35
Females aged 36-45

Note that executing custom code is a relatively expensive process. This event is fired once for each entry in the element list. If you have a lot spin edit elements and/or large lists this can have an impact on screen rendering time. The eXtension recognises the special case of an empty script (the default) and returns the value of ROW.text without having to execute any custom code. Consider carefully if you need to use this event, or if you can change the structure of your table so that ROW.text already contains the correct data.

***onSelectValue* event**:

The code in this event determines what entry in the element list to set as the selected. It can be something simple like:

ROW.value

If the expression does not return a value of type *Boolean,* the default logic of comparing the value is returned with the associated field's value.

An example of an expression returning a value of type *Boolean* would be:

(ROW.value == "Hello")

Note that executing custom code is a relatively expensive process. This event is fired once for each entry in the list. If you have a lot spin edit elements and/or large lists, this can have an impact on screen rendering time. The eXtension recognises the special case of an empty script (the default) and returns the value of ROW.value without having to execute any custom code. Consider carefully if you need to use this, or if you can change the structure of your table to avoid it.

***onSelectValueChanged* event**:

This event fires when the current element value changes. The default behaviour here is

FIELD.setValue(ROW.value)

In other words, set the value of the field (which is referred to as FIELD) associated with the element list to the value of the selected ROW.

***additionalEntries*:**

Lets you add entries in addition to those added by the specified method in *dataSourceType.*

The easiest and preferred way of adding a blank entry at the top of the element list would be to add it to your static table or fixed values with blank values.

Another way would be to use *additionalEntries*.

This property works by calling an internal method named `addTableROW`. It receives 2 parameters:

`oRow`	`Object` - represents the ROW. It must be structurally the same as any of your other ROWs. In other words it must have the same properties.
`bAtBegin`	`Boolean` - Indicates whether to add the entry at the beginning or at the end of the element list. Defaults to `false`

For example, to add an entry at the top of the element list source by the *Gender* table you would this in your additionalEntries property:

```
var oGenderROW = {value: "", text: "Select an entry from this list", agerange:"" }
this.addTableROW(oGenderROW, true); /* Note the this qualifier */
```

Note that when the list is associated to a field, the field will be set to the "" when this entry is selected unless you add logic to the *onSelectValue* and *onSelectValueChanged*.

See Also

Dropdowns in Tutorial 3 - Advanced Screen Enhancement

Static Tables that load from static data in Tutorial 6 - Tables and XML Documents

Static Tables that load from a database file in Tutorial 6 - Tables and XML Documents

XML Documents in Tutorial 6 - Tables and XML Documents

Example – Dynamically refreshing a drop down without server interaction in **Tutorial 11 - FAQ and** Examples

Example – Two level drop down in Tutorial 11 - FAQ and Examples

Example – Using a drop down as subfile option field in Tutorial 11 - FAQ and Examples

axSubfileEnhancer – Enhance Subfile Appearance

Description

axSubfileEnhancer automatically adds simple stripes to the header and body of a subfile on a screen. The stripes are intended to enhance the appearance of the subfile. The subfile defaults to the first subfile on the screen, but this can be overridden to any other subfile on the screen, or moved elsewhere on the screen, using the start and end row properties.

The subfile enhancer is sized by using the row / column properties of the extension.

Useability

Context	Supported
Input screen fields	No
Output screen fields	No
Input subfile fields	No
Output subfile fields	No
Whole screen	Yes

Properties

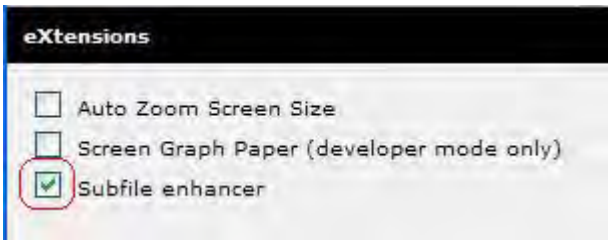
Name	Description and Comments	JavaScript	Shipped default
subfileName	The subfile to have its appearance enhanced. "First" means detect the first subfile on the screen. It can be set to any of the subfiles on the screen.	Yes. Must return "FIRST" or a subfile record format name.	First
headerHeightRows	The number of rows in the subfile header.	Yes. Must be 1 - 24	1
bodyFirstRow	The first row of the subfile body. If 0, this means automatically determine the first row of the subfile body.	Yes. Must be 1 - 24	0
bodyLastRow	The last row of the subfile body. If 0, this means automatically determine the last row of the subfile body.	Yes. Must be 1 - 24	0
highlightRows	Indicates that a stripe be created for each row of the subfile rather than for the whole body.	Yes.	False.
alternateColors	Indicates that rows should be highlighted with alternate colors (ignored if highlightRows is False).	Yes.	False.
linesPerRow	The number of screen lines that represent a row in the subfile. A value of 0 tells the eXtension to attempt to automatically determine the value.	Yes.	0
columnFirst	The first column of the subfile	Yes	1
columnLast	The last column of the subfile	Yes	79
Style	The style applied to the stripes	No	

Notes, Comments and Warnings

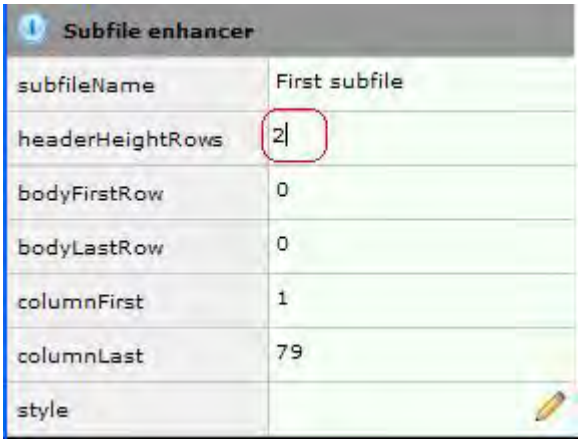
This extension is intended for straightforward subfile modification. (For more complex scenarios, use two Simple Stripe extensions)

Examples

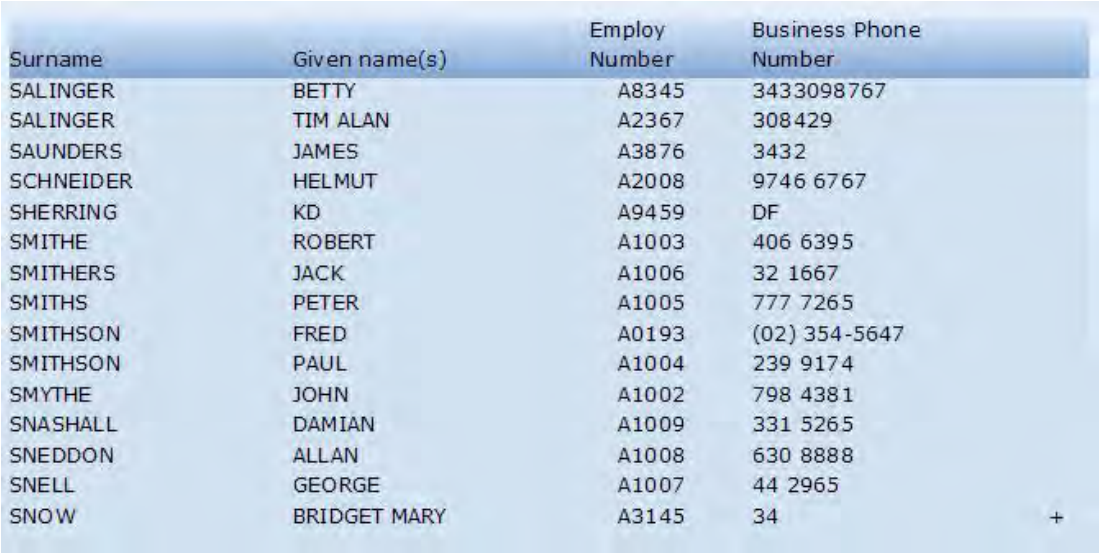
(1). To use the subfile enhancer, edit the screen's properties and click on the screen anywhere outside an element, and check the subfile enhancer checkbox:



Then set the headerHeightRows property to the required value for your subfile



Using the default settings and changing the headerHeightRows to 2, the subfile's appearance can be changed to this:



Surname	Given name(s)	Employ Number	Business Phone Number
SALINGER	BETTY	A8345	3433098767
SALINGER	TIM ALAN	A2367	308429
SAUNDERS	JAMES	A3876	3432
SCHNEIDER	HELMUT	A2008	9746 6767
SHERRING	KD	A9459	DF
SMITHE	ROBERT	A1003	406 6395
SMITHERS	JACK	A1006	32 1667
SMITHS	PETER	A1005	777 7265
SMITHSON	FRED	A0193	(02) 354-5647
SMITHSON	PAUL	A1004	239 9174
SMYTHE	JOHN	A1002	798 4381
SNASHALL	DAMIAN	A1009	331 5265
SNEDDON	ALLAN	A1008	630 8888
SNELL	GEORGE	A1007	44 2965
SNOW	BRIDGET MARY	A3145	34

axSubfileScroller – Insert Subfile Scroll buttons into a 5250 Screen

Description

axSubfileScroller allows a subfile scrolling control 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
markerFieldName	The name of the field that contains the string that indicates the subfile extends to another page	No	Blank
scrollerStyle	CSS to apply to the subfile scrolling buttons.	No	Height = 17px Width = 17px
scrollMarkers	List of strings that indicates that the subfile extends to another page	No	"+", "More..."
pgUpImagePath	Path to locate the page up image file	No	`/ts/skins/images`
pgUpImageName	Name of the page up image file	No	`pageup.gif`
pgDnImagePath	Path to locate the page down image file	No	`/ts/skins/images`
pgDnImageName	Name of the page down image file	No	`pagedown.gif`
pgUpOnClick	Event that fires when clicking on page up.	Yes	SENDKEY("PageUp");
pgDnOnClick	Event that fires when clicking on page down.	Yes	SENDKEY("PageDown");

Examples

Insert subfile scroll buttons into the *Maintain Employee Information* screen of the aXes Demo HR system.

Step 1: Open the *Projects Home Page*. Select your project and then use the *Work as Developer* option and sign on to a 5250 session.

Step 2: Start the shipped aXes demo system by entering the commands:

ADDLIBLE AXESDEMO

CALL XHRRPGTRN

The resulting 5250 screen 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 +

Step 3: Start editing the screen and click on the "+" sign and name that field PageMarker.

Step 4: Add an axSubfileScroller eXtension.

This is done by adding a new element to the screen.

Make it into a *Subfile Scroller* by checking that option.

Change the scroller's *markerFieldName* property to "PageMarker".

Then move and position the scroller as required.

Step 5: Select the PageMarker field and uncheck its Default Visualization option so that its contents no longer appear.

Save your changes.

The modified screen 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	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

Change Scroller Button Images

This example assumes that an axSubfileScroller eXtension has been added to an aXes screen as in the previous example.

Step 1: Open the *Projects Home Page*. Select your project and then use the *Work as Developer* option and sign on to a 5250 session.

Step 2: Start the shipped aXes demo system by entering the commands:

ADDLIBLE AXESDEMO

CALL XHRRPGTRN

Step 3: Start editing the screen and click on the Subfile Scroller eXtension to highlight it and bring up its properties into the aXes Designer window.

Step 4: Click on and change the ***pgUpImageName*** property to movepageup.jpg. Then click on and change the ***pgDnImageName*** property to movepagedown.jpg.

Step 5: Click on the ***scrollerStyle*** property and change the ***height*** and ***width*** style attributes so that the scroller buttons will be large enough to display the new images. Use 22px for each attribute.

Save your changes.

The modified screen 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	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

axTimer – Insert a timer into the screen

Description

axTimer allows a timer to be added to the screen. The timer is not visible at run time. When the timer interval has passed, the timer ticks and executes the script specified.

The timer can be set to tick repeatedly or just once, or by setting the interval to 0, it can be set to not tick at all.

A timer would normally be added as a new element to a screen. When creating the new element you will need to manually uncheck the default visualization.

A timer will be invisible unless the screen is being edited.

There can be multiple timers on a screen.

Useability

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

Properties

Name	Description and Comments	JavaScript	Shipped default
tickInterval	This is the time between ticks, in milliseconds	Yes	0
onTick	This is the script that will be executed when the timer ticks	Yes	alert('timer tick occurred');
keepTicking	This indicates whether the timer will tick once and stop (false) or tick repeatedly (true)	Yes	true

Warnings

When creating a new element, you will need to check the timer extension and uncheck the default visualization extension.

Examples

(1).

<u>Add a timer that ticks once, after 10 seconds, and issues an alert</u>

Edit a screen and add a new element to it, and make it a Timer

Set its properties to:
tickInterval: 10000

onTick: `alert('10 seconds has elapsed');`
keepTicking : false

Save the screen. After 10 seconds an alert will appear:

"10 seconds has elapsed"

If you click ok on the alert, the timer will not tick again, because property keepTicking is false.

(2).

Add a timer that checks every 10 seconds whether the user has entered a value, and stops checking when a value has been entered.

Find and edit a screen with an input field on it.

Name the input field "myField".

Add a new element to the screen and make it a timer.

Set its properties to:
tickInterval: 10000
onTick:

```
alert('timer 10 seconds tick occurred');

var F = FIELDS("myField");
if ( F.getValue() != "")
{
alert("setting my tick interval to 0");

FIELD.setProperty("tickInterval", 0);
FIELD.refresh();
}
```

keepTicking : true

Save the screen and ensure the field named myField is blanked out.

Every 10 seconds an alert will appear: "timer 10 seconds tick occurred".

Type in a value into the field named "myField".

The next alert will be:
"timer 10 seconds tick occurred"

followed by:
"setting my tick interval to 0"

Then there will be no more ticks, because the timer's interval has been set to zero.

(3).

Add a timer that changes the style of an input field, after 2 seconds.

Find and edit a screen with an input field on it.

Change the field to use the Default Visualization *and* the Timer extension.

Set the input field's properties as follows:

tickInterval: 2000
onTick:

```
var oStyle = null;
oStyle = {"color":"red", "background":"blue"};

FIELD.setProperty("axdv.style" , oStyle );
FIELD.refresh();
```

keepTicking: false

Note that the style that is changed is the style of the default visualization, and this is distinguished from the Basic style by using "axdv.style"

axUserElementBasicProperties - New Screen Element Basic Properties

Description

Every new screen element has a fundamental and common set of properties, regardless of any further visualization that is applied to it.

New screen elements do not exist naturally on the 5250 screen.

They have been added to the 5250 screen layout by an aXes developer.

Useability

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

Properties

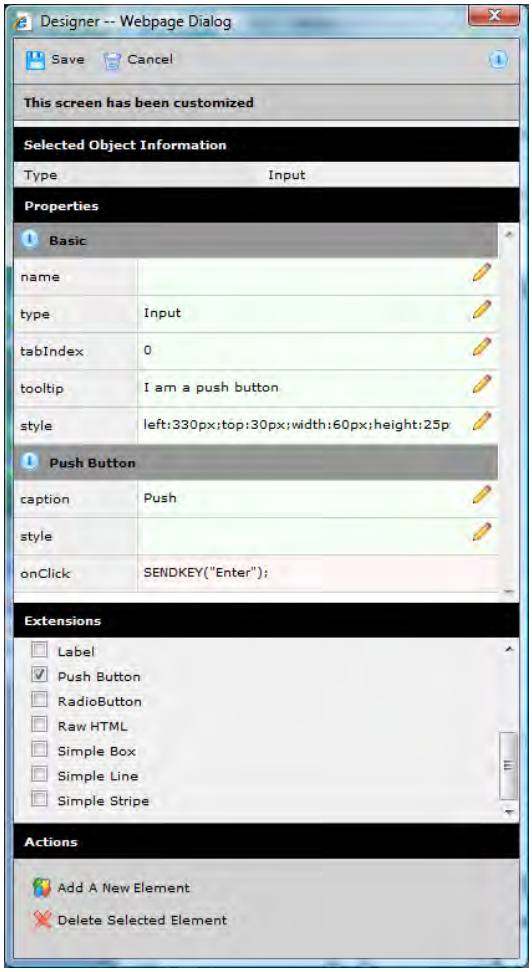
Name	Description and Comments	JavaScript	Shipped default
name	The symbolic name of the new screen element. Just as 5250 screen elements may have names, new screen elements may have names.	No.	None.
type	The type of the new screen element. May be input or output. The use of this property on new screen elements is rare.	No.	Input.
tabIndex	The tab index that should be applied to this element relative to all other elements on the screen.	No.	0
tooltip	The tool tip (or hint) that should appear when the user hovers the mouse over this screen element.	Yes. Must return a valid string.	None.
style	Additional style properties that should be applied to the screen element.	Yes. Must return a valid style object.	None.
visible	Indicates whether this element is visible.	Yes.	True.
enabled	Indicates whether this element is enabled.	Yes.	True.

Notes, Comments and Warnings

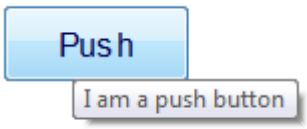
None.

Examples

(1). If a new element is added to a screen and visualized as a pushbutton, its property sheet may look like this, with the **tooltip** property set to *I am a push button*:



When the user hovers the mouse over the push button the tooltip appears:



Documentation Library

Extensions Problem Isolation Techniques

Checking IBM i PTF levels

Checking Axes Server Job Logs

Using Fiddler

Checking IBM i history logs

Checking Axes CERN logs

Checking IFS File Authority

Checking Customized IBM i Logon Screens

Checking for network/browser compression issues

Checking IBM i PTF levels

To use Axes some IBM supplied IBM i PTFs are mandated:

IBM i Version	PTFs required
V5R3M0	**SI15896** – CPF8902 occurs. Unable to create new virtual devices **SI22468** – Virtual terminal feedback not available when QSECURITY =30 o
V5R4M0 **V5R4M5**	**SI35079** - QTVOPNVT ignores initial pgm, menu, curlib with Key 8. This PTF also includes and supersedes SI22470.

- To check if a PTF is installed use the *DSPPTF* command.
- Also check that it has been applied temporarily or permanently.
- After applying a PTF stop and restart all Axes server jobs.

Checking Axes Server Job Logs

Often checking the job logs of Axes server jobs can reveal information about a problem:

Does this facility need to be installed separately? If yes, where do you get it from?	No.
How you turn it on / start it?	It is always on.
How you turn it off / stop it?	It cannot be turned off.
Are there performance ramifications in using it?	No.
How is it typically used to isolate problems?	Use the ***WRKSBSJOB* AXES** command to display all Axes server jobs active in the AXES subsystem. Use option *5 - Work With*, then option *10 - Display job log, if active, on job queue, or pending*. On the resulting *Display Job Log* screen, use *F10=Display detailed messages* and then scroll through all the message displayed. Look for messages of interest that indicate warning or error situations. Use F1=Help on any message to display more details the time and date it was issued, which is often useful in associating a message with a problem. Repeat this procedure for all jobs active in the AXES subsystem.
What sort of information might you be asked to send to Axes support?	If an Axes server job contains messages of interest, record the job name, user and number from the top of the screen (eg: job AXES, user AXES, job number 041940). Use the ***DSPJOBLOG*** command to print details of all the messages associated with the Axes server job: Eg: DSPJOBLOG JOB(041940/AXES/AXES) OUTPUT(*PRINT) Locate the report (spool file) produced and send it to Axes support as a simple text document. Please avoid sending screen prints.

Using Fiddler

Fiddler runs on your PC. It monitors all traffic between your web browser and the Internet. When captured and saved this traffic can be a useful source of problem resolution information.

Does it have to be installed separately? If yes, where do you get it from?	Yes. Fiddler can be downloaded from http://www.fiddler2.com/fiddler2/
How you turn it on / start it?	Run the Fiddler application on the same PC as the browser. To start capturing traffic, select *Capture Traffic* from the *File* menu.
How you turn it off / stop it?	Deselect *Capture Traffic* from the *File* menu or close the Fiddler application.
Are there performance ramification in using it?	If run for long periods of time it may develop some memory or disk space problems. Normally you would only run it while reproducing a particular problem.
How is it typically used to isolate problems?	Fiddler is a proxy that sits between the browser and the server and captures all traffic between the two. It makes it possible to see exactly what is, or isn't, being sent between the two. In addition, the sessions can be saved for later playback making it possible to reproduce and diagnose a problem remotely, without needing a VPN or host connection.
What sort of information might you be asked to send to Axes support?	You may be asked to send a Fiddler log. This would involve starting Fiddler and then turning on *Capture Traffic* just before reproducing the problem. After the problem has been reproduced you save the traffic details to a Session Archive file (*.saz) by selecting *File* > *Save -> All Sessions*. Send the resulting .saz file to Axes support.

Checking IBM i history logs

All IBM i servers maintain a history log called QHST. In sites with suspect communication links QHST will often record TCP/IP level drop outs and recoveries. If an aXes system is exhibiting intermittent or random issues then the root problem may be traceable to TCP/IP level communications issues. In such situations you need to rectify these basic TCP/IP level issues before aXes will function reliably. For example, people using a wireless link or with a loose or failing LAN connection may experience problems, as will people whose network routers play up or are re-booted.

Does it have to be installed separately? If yes, where do you get it from?	No
How you turn it on / start it?	It is always turned on.
How you turn it off / stop it?	It is always turned on.
Are there performance ramification in using it?	No.
How is it typically used to isolate problems?	Use the IBM i command DSPLOG and scroll to messages issued around the time of the aXes issue. Look for any messages related to communication errors.
What sort of information might you be asked to send to Axes support?	Screen prints of the DSPLOG command showing messages issued at or around the time of the aXes issue.

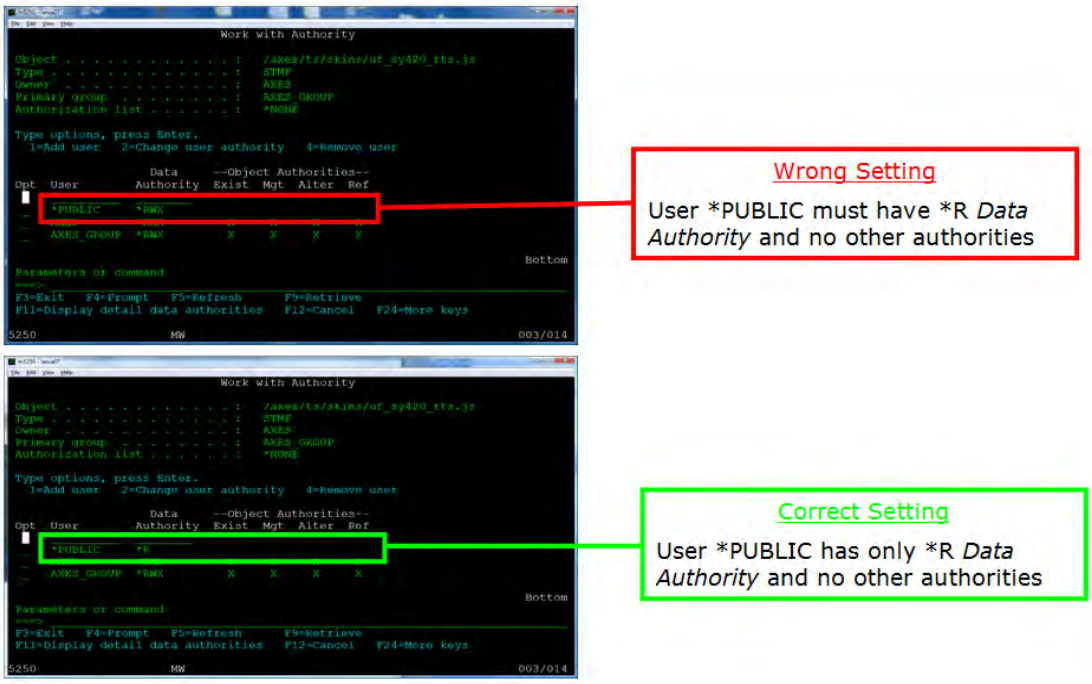
Checking Axes CERN logs

Axes can produce CERN logs. These may reveal information about a problem. It is most useful in situations where you cannot install Fiddler onto the problem PC.

Does it have to be installed separately? If yes, where do you get it from?	No
How you turn it on / start it?	Edit /axes/configs/aXesW3.conf by using the WRKLNK command: ▪ Uncomment this directive by deleting the # in front of it: **#CernLog=1** ▪ Uncomment the directive by deleting the # in front of it: **#LogPath=/axes/logs/** ▪ Save the changes to /axes/configs/aXesW3.conf ▪ Stop and restart the axes server jobs
How you turn it off / stop it?	Comment the directives that were uncommented in the previous point. Stop restart the Axes servers.
Are there performance ramification in using it?	Yes, but they are not very heavy, so it is okay to leave CERN logging on in multi-user production environments for short time periods. Large amounts of trace information may be produced.
How is it typically used to isolate problems?	Entries in the log file typically contain information about the success or failure of http transactions. Typically you would search for bad return codes from http requests like 401, 403, etc.
What sort of information might you be asked to send to Axes support?	Log files are located in the directory specified in the LogPath directive in file /axes/configs/aXesW3.conf. By default this is /axes/logs Log files follow the naming pattern: **cernYYYYMMDD_hhmmss** where the date time refers to when the server was last stopped. To obtain a log file for a transaction that occurred between the time the server was LAST started and now you will have to stop the server to allow for the log file to be created. If requested send the CERN logs files to Axes support.

Checking IFS File Authority

In general Axes files stored on the IBM i IFS should be set so that user *PUBLIC has *Data Authority* *R only and no other authorities. When files do not follow this rule they may not be served, causing problems in client side applications. Use the IBM i *WRKLNK* command to review and alter IFS file authorities. For example:



Checking Customized IBM i Logon Screens

Axes imposes some limitations on how customized IBM i sign-on screens are formatted. A problem is typified by this error message: **CPF1296 – Signon information required**. (assuming that all mandated PTFs have been applied).
To verify that a custom sign-on is causing an issue with Axes follow these steps:

> Note: Do not use *LIBL or omit library names from any of the following commands. If you do not specify an exact library name you risk checking the wrong thing.

- Find out the name of the subsystem that Axes 5250 sessions are attempting to log into. Typically this is QINTER, but it may be different at some sites.
- Use a **DSPOBJD *all/xxxxxx *sbsd** command to identify how many subsystems named xxxxxx exist. Typically only one exists. If multiple exist you need to determine which one is being used when you attempt to sign on.
- You should now have a single subsystem name (eg: QINTER) and library (eg: QSYS).
- Next determine the exact details of the customized logon screen by using the command **DSPSBSD llllll/xxxxxxx** where llll is the library name and xxxxxx is the subsystem name. Then use menu option *1- Operational attributes* and record the sign-on display file and library name, like in this example:

```
Sign-on display file . . . . . . . . . . :   QDSIGNON
  Library  . . . . . . . . . . . . . . . :     QSYS
```

- Now used the command **DSPFFD llllll/dddddd** where llll is the display file library and dddddd is the display file name, like this example – DSPFFD qsys/qdsignon.
- Now examine the 2nd and 3rd displayed pages displayed by the DSPFFD command and fill in this table from the details:

Field	Row in display file	Column in display file	Visual Order of field (working from top to bottom, left to right)
USERID			
PASSWRD			
PROGRAM			
MENU			
CURLIB			

√ For example, the DSPFFD results for the shipped IBM sign-on screen QSYS/QDSIGNON yields this table:

Field	Row in display file	Column in display file	Visual Order of field (working from top to bottom, left to right)
USERID	6	53	**1**
PASSWRD	7	53	**2**
PROGRAM	8	53	**3**
MENU	9	53	**4**
CURLIB	10	53	**5**

If the display file you are checking does not yield a display order of exactly 1 -> 5 for fields USERID -> CURLIB then it may need to be altered to work with Axes.

√ Finally, check that *all* of these fields, and *only* these fields are defined in the display file: SYSNAME, SBSNAME, DEVNAME, USERID, PASSWRD, PROGRAM, MENU, CURLIB, QSNERROR, COPYRIGHT and UBUFFER.

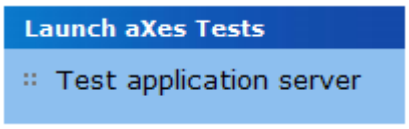
Checking for network/browser compression issues

aXes installs many files on the IFS in compressed format. It delivers them to the browser in this compressed form. Sometimes networks and/or browsers are unable to handle the compressed format correctly (the circumstances in which this problem occurs are arcane).

A compressed file problem is typified by a JavaScript error that says something like "Line 1 character 1 is invalid" - because the compressed file cannot be processed by the JavaScript interpreter.

You can confirm you have this problem by doing this simple test:

Go to the aXes home page and use this option:

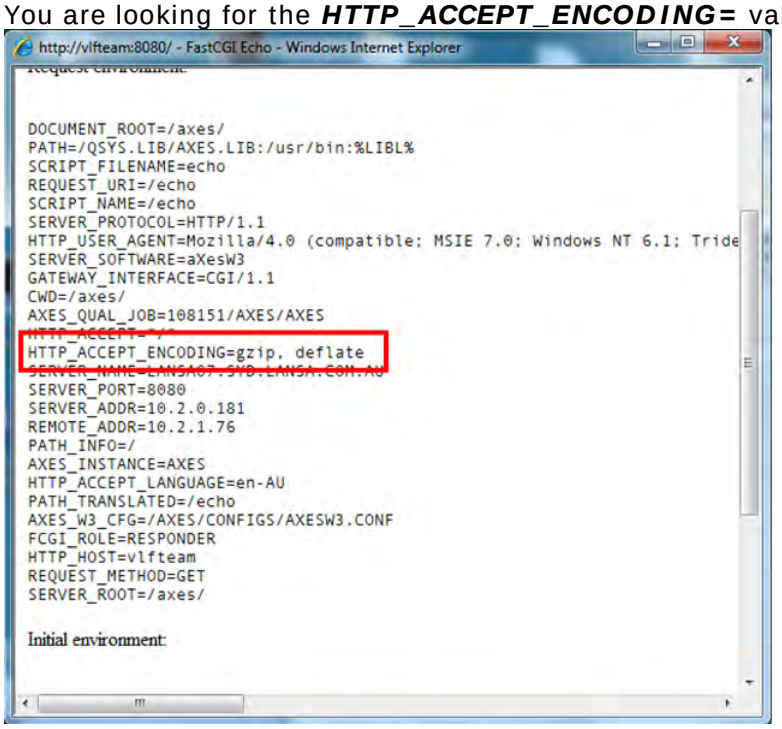


If you cannot even display the home page use a URL formatted like this instead:

http://<aXes host and port>**/echo**

Scroll through the resulting display.

You are looking for the ***HTTP_ACCEPT_ENCODING=*** value - like this:



If the HTTP_ACCEPT_ENCODING= option is not displayed -OR- if it does not say gzip, deflate then you have this compression problem.

If you have the problem then you should use the aXes DECOMPRESS command against the various aXes IFS folders, the most significant one being the <axes root folder>/ts folder. After decompressing you must clear your browser caches.