Arterial XML eBusiness Server

Development Kit (ADK)

5th March, 2005

IBM TOWER
LEVEL 3, 60 CITY ROAD
SOUTHGATE 3006
VICTORIA, AUSTRALIA

VOICE 613 9421 5911
FACIMILE 613 9428 3729

WWW.ARTERIALSOFTWARE.COM
INFO@ARTERIALSOFTWARE.COM

aXes Development Kit (ADK)

aXes Development Kit (ADK)

aXes Development Kit (ADK)

The aXes Terminal Server Development Kit (ADK) is available to partners and customers to assist in the customization of aXesTS. Contact Arterial Software for access to the download site.

Customization guidelines

Following these guidelines will make your customization a straightforward process. While it is not necessary to follow them strictly, they will prevent you from creating JavaScript and authority errors that may be hard to trace.

Developer prerequisites

This guide assumes that the developer undertaking the customization will have a solid understanding of HTML and Cascading Style Sheets (CSS). Knowledge of JavaScript and XML will be beneficial.

Backing up files

Before you make a change to any file on the iSeries, be sure to make a backup copy of that file.

Restricted files

You should avoid changing any of the files in the /ts/ directory with the exception of the three keyboard map files: AS400KeyMap.xml; DefaultKeyMap.xml; and WindowsKeyMap.xml. These files are dealt with in “Customizing keyboard maps”.

You can modify any of the files in /ts/skins/ and /ts/skins/images.
The files in /ts/test/ are for the ADK only, and do not effect aXes TS. They are for offline testing purposes only.

Modifying files on the iSeries

Before testing your files live on your iSeries, it is wise to create a test directory from which your customised files can be served. This will force you to create backups of all the files in the /axes/ directory as well as prevent files updated by Program Temporary Fixes (PTFs) from overwriting your changes.

Create a directory on the IFS called /axestest/ and copy the /ts/ directory and all of its subdirectories from the /axes/ directory.

Then change the following directives in /axes/configs/aXesTS.conf:

TSPath=/axestest/ts

AdminPath=/axestest/ts/admin

Restart the aXes server and your aXes TS files should now be served out of your new directory.

When you want to test your changes online, you should transfer ONLY the files you have modified to their respective directories on the iSeries. Do not transfer anything you have changed in the /ts/test/ directory.

If you make a change to a file in the /ts/skins/ or /ts/skins/images/ directories, you do not have to restart the aXes servers for that change to take effect. You will have to restart the servers for changes made to files in the /ts/ directory.

Applying PTFs

Some PTFs that you apply to aXes may effect files that you have customised. It is up to you to determine if this is the case by reading the PTF’s cover letter carefully. If there is such an occurance, you will need to copy your changes from the customised file into the updated PTF file.

In all cases with PTFs, if a file under /ts/ has changed, and you are not running TS from the product directory (/axes/) then you will need to copy the files changed in the PTF from the install directory to your running directory (/axestest/ if you used the example above).

File authorities

Files that are transferred to the host MUST have their authority set to *PUBLIC *R or they will not be served to the client. A 403 permission error is the likely result of a file with the incorrect authority.

aXes Development Kit (ADK)

Links in skins

If you add HTML links such as onclick events or <a> elements to skin files, they should always be set to open the linked document in a new browser window. If a document is opened in the window where a live aXesTS session is running, the session will disappear and the user will be required to reconnect to that session from the login screen.

Internet Explorer settings

Internet Explorer's advanced settings (internet options/advanced) should be set to the default with one exception. Be sure that the option "Display a notification about every script error" is checked. It is important that you are aware of JavaScript errors while you are customizing the skin and script files.

aXes Development Kit (ADK)

Customization overview

The ADK enables you to work on the customization of aXes TS skin files, style sheets and even JavaScript files without having to use the iSeries. Changes you make to files in the /ts directory can be displayed using the ADK as if they were being served lived on the internet. This will save you a lot of development time and effort.

How do I install the ADK?

Make sure you download the latest version of the ADK before you begin the installation. When PTF's are released that effect aXesTS, a new version of the ADK may also be released. You can check the date of creation of your version of the ADK by reading the source of /ts/skins/ADK.html. If a PTF has been released for a later date then your ADK may need to be updated.

The ADK zip file MUST be unzipped in your PC's root directory (c:/ for example). Extracting the files will create a directory called /ts, so be sure that there is no directory called /ts in your root directory before unzipping the file.
What's in the ADK?

The ADK contains all of the stream files (with the exception of the help and administration files) that are used on the client PC to transform each XML terminal transaction that is received from the host iSeries.

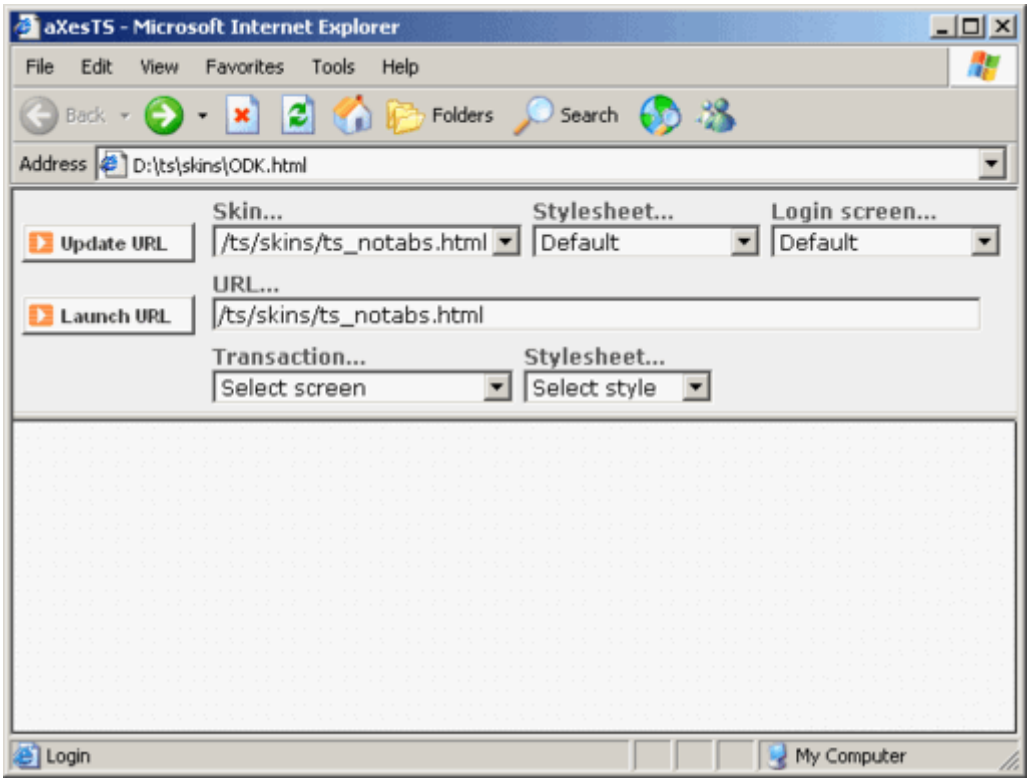
The directory structure is as follows:

/ts/	- Files used to transform XML transactions. Keymap files.
/ts/skins/	- Files used to create the aXesTS skin. Style sheets and login screen.
/ts/skins/images/	- Image files used by the style sheets and skins.
/ts/test/	- Files used by the ADK. Saved transactions.
/ts/test/samples/	- Files used by modified HTML transactions.
/ts/test/example/	- Files used in the "Companyone" worked example.

How do I start the ADK?

Open the file c:/ts/skins/ADK.html in Internet Explorer. If the ADK has been installed correctly, you will see the following screen loaded in the browser.

aXes Development Kit (ADK)



If the screen doesn't load properly, you have probably unzipped the ADK into a subfolder on your hard drive. The ADK needs to be unzipped in the root directory. Make sure that there is not a directory on your hard drive called /ts before unzipping the ADK file.

How do I use the ADK?

After you have loaded the ADK into Internet Explorer, you can load your choice of skin into the bottom (design) frame by entering the correct URL and pressing the "Launch URL" button.

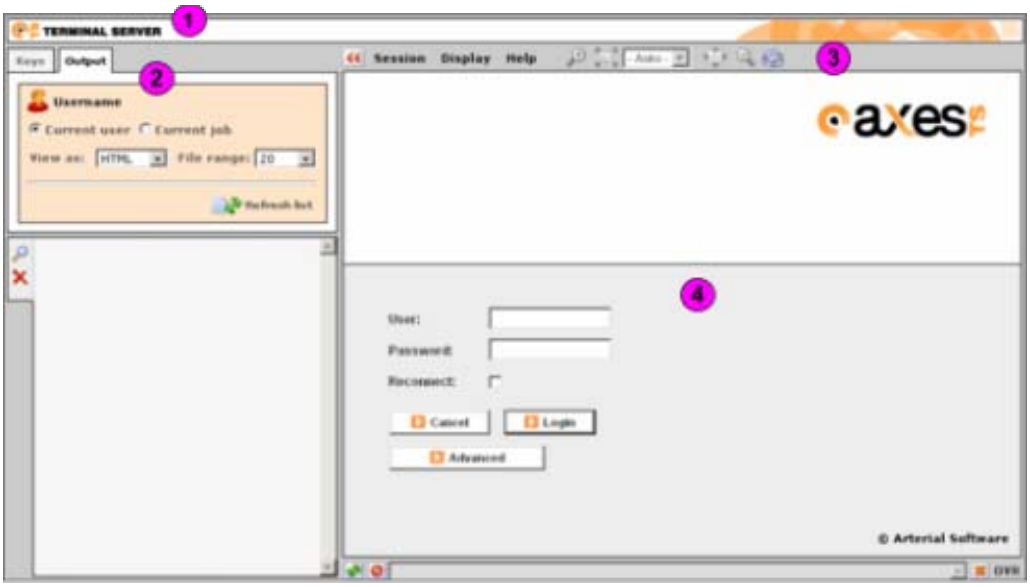
To assist you in choosing the right skin, use the three drop down lists at the top of the screen. You can add values to these lists once you have created custom skin files.

After choosing a skin, you can also choose a style sheet and login screen combination to load at the same time. When you have made your choices, press the "Update URL" button, and the URL entry field will be updated with your choices. You can also manually enter URL's into the URL field.

What is a skin?

An aXesTS skin is a collection of HTML files governed by one main file (ts.html by default) that determines how the terminal will be displayed to the user. The main file contains a set of frames that each represent parts of the aXesTS screen (see below).

aXes Development Kit (ADK)



The main components of the skin are:

- The header bar
- The tabs frame
- The main frame
- The terminal frame

The only frame that must be present for an aXesTS session to begin is the main frame as it contains the terminal frame, where the aXes host transactions are processed.

The skin is also where default values are set for the:

- Login screen file name
- Default style sheet
- Default keyboard map
- Default text size option
- Insert/Overwrite mode option

Skin files include, but are not limited to HTML, CSS, JS, XML and WAV file types.

What is a transaction?

ADK's transactions are local copies of actual, live terminal screens. For example, after logging in to a live aXes TS session you may see the "Main Menu" screen. The data for this screen has been sent to your browser as an XML transaction that is interpreted by the terminal frame. This XML data can be copied to a local file for use with the ADK. In this way, you can copy every screen of your application onto your PC to view offline.

See "How do I save a transaction?" for more about transactions.

To view a transaction, after loading a skin, simply choose an option from the "Transaction..." drop down list. If a skin has not been loaded a warning message will appear.

What is a style sheet?

There are two types of style sheets that affect aXesTS: style sheets that are embedded in HTML documents with <link> tags; and style sheets that aXesTS uses to display transactions. Both have the file extension ".CSS".

aXes Development Kit (ADK)

Embedded style sheets are common in web design and you should be aware of them before attempting to customize any aXes component. They are used to control the colors of the skin backgrounds and fonts etc.

The aXesTS terminal frame applies a style sheet's rules to the transactions it receives from the iSeries. These style sheets, which can be changed at any time during an aXes TS session, control various aspects of what is shown in the terminal frame.

These aspects include the:

- Font
- Font size
- Text colors
- Background color
- Entry field colors
- Borders
- Background image
- Cursor color

You can specify a terminal style sheet in the ADK's URL using the drop down list in the top row, or you can change the style sheet after you have loaded a skin using the style sheet selection list in the bottom row.

What's different about the login screen?

aXesTS login screens are XML documents that are loaded by the terminal frame before a connection to your iSeries application has been established. It is similar to a normal transaction, but you can redesign it to suit your company's corporate image.

Each login screen contains its own style sheet definitions, so changing the current terminal style sheet will have no effect on the login screen.

You can specify a default login screen by customizing the skin, or you can add it to the launch URL using the "Login screen…" drop down box in the top row of the ADK.

What can I put in the URL?

The URL entry field can be used to emulate the URL that you would use to start a live TS session. For example, a URL to start a live session may be "http://ipaddress:port/ts/skins/ts.html". In the ADK, this URL would be represented as "/ts/skins/ts.html". So when you are finished developing the offline version of your customized skin and have uploaded the files to the host, you can use the URL from the edit field to start a live session that uses your changes.

As well as specifying style sheets and login screens in the URL, you can set login screen variables.

The variables that can be set are:

- user= [the username you want to set]
- reconnect= [true or false]
- device= [a device name]
- program= [program to load]
- menu= [menu to load]
- curlib= [current library name]
- pwd = [password]
- encrypt = [whether the specified password has been encrypted, true or false]
- signon = [auto signon with specified username and password, true or false]

You set these values by adding them to the end of the URL in the following fashion.

/ts/skins/ts.html?user=wally&device=dev01

or

/ts/skins/ts.html?login=axes_custom.xml&user=wally&menu=abc123

Enter some values such as these in the URL edit field and press the “Launch URL” button to see the results! (You may have to press the “Advanced” button on the login screen to see the fields you have changed.)

What does the Encrypt Password button do?

If you specify the pwd login screen variable, for example “/ts/skins/ts.html?user=wally&pwd=mypass” and you press the “Encrypt Password” button, the password wil be encrypted and encrypt=true will be appended to the URL. In this case, the resulting URL would be similar to:

/ts/skins/ts.html?user=wally&pwd=tnEOGHt2/VlbZt4/eSI6zITlxI0=&encrypt=true

You can use this URL in a live TS session, and the password wil lbe decrypted automatically. Using this in conjunction with the “signon=true” login variable will start a session automatically. However, aXes does not recommend including passwords in URLs as they are not secure. They can be useful for guest passwords and demo sites where security is not a concern.

What does the terminal frame hover text mean?

When you hover the cursor over objects in the terminal frame, hover text will show up next to the cursor. The text shows the CSS class name of the object under the cursor. This class name can be used to help customize the terminal style sheets as it lets you easily see what class all of the objects on your screen are, without reading the awkward XML documents.

For example in the “Main menu” transaction, the hover text for the word “Gryphon” is “.Text PRE .x20”.

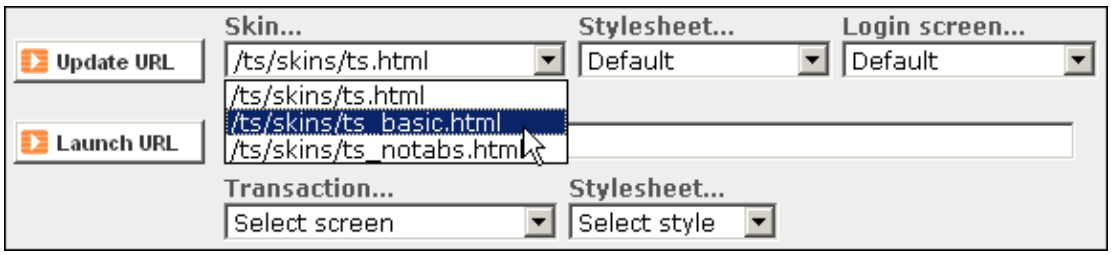
Note that the login screen has its own style sheet rules and the hover text on this screen is irrelevant.

See “Customizing terminal style sheets” for more information.

Working with the ADK

When you make a copy of a skin, style sheet or login screen you will need to change the ADK so that the copy you are working on is displayed in the ADK lists. Otherwise you will have to manually enter the URL of the new files each time the ADK is refreshed.

The file that you need to modify to add new filenames to the ADK drop down lists is /ts/test/command.html.



How do I add a file to the ADK?

Open command.html in a text editor and find the <select> group for the type of file you are adding.

For example, if you have created a copy of the file ts.html named ts_custom.html, you need to add the following entry in the <select id="skin"> group:

<option value="/ts/skins/ts_custom.html" >/ts/skins/ts_custom.html</option>

Now when you refresh the browser, the option will appear in the "Skin…" drop down list.

This example applies to skins, login screens, style sheets and transactions. Style sheets should be added to both the id="cssref" group and the id="stylesheet" group.

How do I save a transaction?

You can save transactions from your application to work with offline. This will help you when you are customizing your terminal style sheet.

To save transactions to your PC follow these instructions:

1. Start an aXesTS session and navigate to the application screen that you want to work with offline.
2. From the "Help" menu, select "Diagnostics" and then "View Transaction".
3. Press the "Copy to clipboard" button.
4. Paste the clipboard contents into a new text editor document.
5. Save the document as "myTransaction.xml" in the /ts/test/ directory on your PC.

You can now add the transaction to the "Transaction…" drop down list using the instructions in "How do I save a transaction?".

aXes Development Kit (ADK)

Customizing the login screen

Most customers will want to redesign the default TS login screen to suit their company's image. This is accomplished by copying and modifying the file /ts/skins/axes_login.xml.

What's in the file?

axes_login.xml contains all of the JavaScript, CSS style definitions and HTML required to display the file. While it is an XML file, the terminal frame treats it as though it was an HTML file. Any HTML tags you put in-between the <HTML> and </HTML> tags will be included in the resulting login page.

What can I change?

You can add HTML between the first and last tags and add or change style properties between the <style></style> tags. You can reorder the existing HTML tags, or make them invisible to the user.

For example, if you don't want your users to have access to the advanced login fields, you can hide the "Advanced" button by changing the style property for the button from:

```
#advButton {
       width: 150;
       }
```

to

```
#advButton {
       width: 150;
       display: none;
       }
```

You should not remove any of the existing entry fields from the HTML. If you don't want them to be displayed on the screen use the "display: none" style sheet rule. If you remove an entry field you may encounter JavaScript errors.

Should I modify the JavaScript?

Unless you understand the JavaScript in the default file, you should leave it alone. Do not remove any of the script or you may encounter JavaScript errors.

aXes Development Kit (ADK)

Customizing terminal style sheets

The terminal style sheets effect the visual aesthetics of your host application screens. The default style sheets are:

- /ts/skins/axes_default.css
- /ts/skins/axes_green.css
- /ts/skins/axes_white.css
- /ts/skins/axes_custom.css

All of these style sheets are applied after the initial style sheet, /ts/axes.css is loaded. axes.css should not be modified or renamed. However, you can make a copy of it in the /ts/skins/ directory if you would like to use it as a starting point for your customization.

Where do I start?

The easiest way to understand terminal style sheets is to make small changes and see how they effect the transactions in the ADK.

Start by copying and renaming an existing style sheet (such as axes_default.css) and adding the new filename to the ADK's command.html (see "How do I add a file to the ADK?").

Load a skin using the new style sheet, then select the "Main menu" transaction. You can also save you own transactions to work with offline (see "How do I save a transaction?").

Now that you have the transaction loaded in the ADK, hover the mouse cursor over the text "AS/400 Main Menu" on the screen. The hover text that appears will tell you that object's CSS class name. In this example, ".Text PRE .x22". This means that the object you are hovering over is a ".Text PRE" object with a specific class of ".x22".

To change how that object is displayed, you need to find the reference to that object in the current style sheet. In axes_default.css, this style definition is provided in two places as follows.

```
.Text PRE,
.InField INPUT,
.InField TEXTAREA,
.SelMenu,.SelField,.SelList,.SelButton, .SelPullDown,.WinTitle,
.Cursor {
    font:10pt "Lucida Console", "Courier New", monospace;
    font-weight: normal;
    }
```

This definition sets the font family, font weight and font size for all of the text elements on the screen.

```
.x22, .Infield .x22 {color:#990099;}
```

This definition sets the color for ".x22" attributes to "#990099". The ".Infield .x22" class value specifies that ".x22" attributes will have the same appearance whether or not they are text objects or input field objects. This definition could also have been written as:

```
.Text PRE.x22, .Infield .x22 {color:#990099;}
```

If you now change the color value above to #009900 (or green) and refresh the ADK, you will see that all elements on the transaction screen with ".x22" attributes are displayed in green.

If you hover over an element that shows a class name such as ".InField INPUT .x24 (.x3a)", the value inside the brackets is the focus attribute. This means the class of the object changes to that value when the object receives the focus. For example, clicking in an entry field.

What fonts can I use?

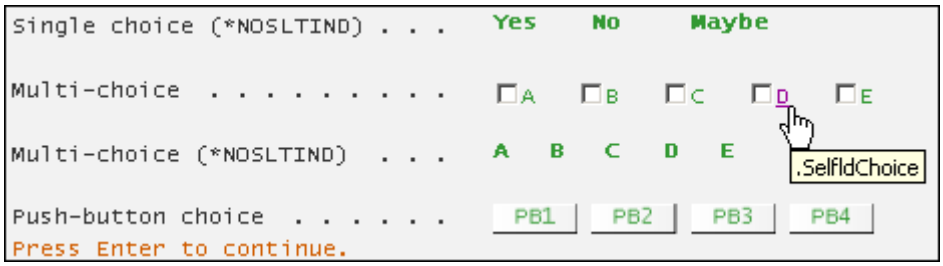
Because of the nature of the iSeries, each character should to take up the same amount of space on the screen so that each line of text aligns with the line beneath it. For that reason we recommend that

you use a fixed-width font such as “Courier New” or “Lucida Console” for all of the elements on the screen.

If you do not use a fixed width font, the cursor may not position itself correctly over each character. If you have a custom application where cursor positioning and alignment is of no concern, then you can use any standard web safe font.

What about DDS keyword elements?

GUI elements that appear in transactions such as push buttons and radio boxes have been included in your application using DDS keyword definitions (see the “DDS keyword lists” transaction). In the terminal style sheets they are defined in the same way as normal text objects, with the addition of “Hover”, “X” (selected) and “Focus” attributes.



The ADK hover text will show you the base class name of the object you want to change, and from there you can figure out which additional attributes are available for that object by inspecting the axes_default.css values.

For example, if you want to change the values of a “.MenuChoice” menu object, you can also set the values of the “.MenuChoiceX”, “.HoverMenuChoice” and “.MenuChoiceFocus” attributes among others.

What about embedded HTML?

If HTML elements have be embedded in your applications using the HTML DDS keyword, the styles of the elements can be controlled through the terminal style sheets.

For example, in the ADK transaction “HTML Keyword example”, an HTML header is embedded using the <H2> element. If we add the following line to the end of our style sheet:

```
H2{ font-family: arial; color: red}
```

then that texts properties will change accordingly.

The blocks of HTML themselves are contained within <DIV> elements that have the class name “.HTMLKW”.

Do I have to specify styles for every object?

All of the available properties are initially set in axes.css. If you look in axes_green.css for example, you will see that it only specifies a font. Therefore, you only need to add style properties for those elements you wish to change.

How do I set a style sheet to be the default?

To make your new style sheet the default that is loaded by the skin. You need to change the value of the default style sheet in ts_frame.html. See “Setting default values” for more information.

How do I add a style sheet to the “Select Style sheet” popup menu?

First read the instructions below in “6.3 Modifying the main frame and menus”.

To add a new style sheet to the list in the popup menu, find the block. Then duplicate one of the existing <a> blocks within it and change the following values:

```
<a class="popup_item" href="#" hidefocus="true" onfocus="this.blur()"
      onclick="parent.SetTermStyle('myStyle.css')">My Style</a>
```

Replace **myStyle.css** with the filename of your new style sheet. The value in “**My Style**” will now appear in the drop down list.

aXes Development Kit (ADK)

Customizing the skin

The most important rule in customizing aXesTS skins is to only modify copies of the original files. When the time comes to upload the files to the host, only upload files that you have changed, then you can still use the default skin files to start a session if there is a problem with the customized files.

Make sure to read the section “Customization guidelines” at the beginning of this document before attempting to modify any files.

What can customization achieve?

Once you have aXesTS installed and running successfully on your iSeries you can use the ADK to create a new look for your installation. Using HTML, CSS and JavaScript you can:

- Put your own logo in the header.
- Add or remove tabs from the tab panel.
- Add menu items and headings.
- Change text and background colors.
- Embed aXesTS in an existing HTML page.
- Change default options.
- Add functionality with scripts that access databases.

Where do I start?

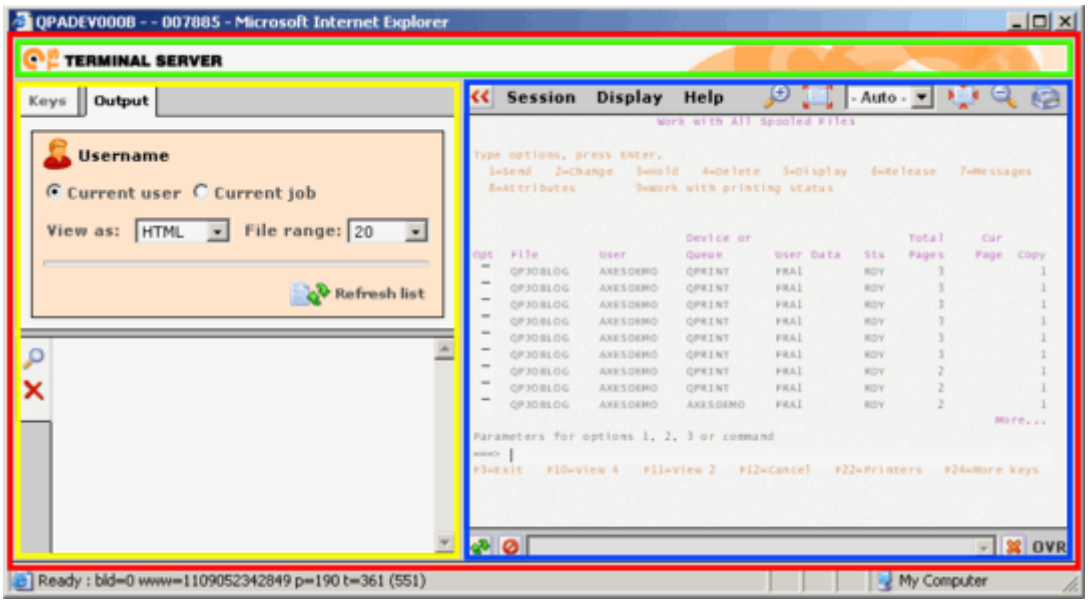
The most important file in the customization process is /ts/skins/ts.html. The first step in any customization should be to duplicate this file and rename it. For the purpose of this document we will assume the copy of the document has been renamed “custom_skin.html”.

Add the file to the ADK “Skin…” drop down list using the instructions in “How do I add a file to the ADK?”.

aXes Development Kit (ADK)

What's in the skin?

"ts.html" contains the framesets that position all of the frames in the browser window as shown below.



- ts.html - header.html - tabs.html - ts_frame.html

The hierarchy of skin files is as follows:

ts.html	**header.html**	Contains logos and other non-essential items.
	vspacer.html	Provides padding to the tabs frame.
	tabs.html	Contains other sets of frames in a tab format.
	ts_frame.html	Contains the menus and terminal frame.

The only file in the framework shown above that is required to successfully start an aXesTS session is ts_frame.html. It contains the terminal frame and scripts required by the TS server. If necessary, you could create a skin that didn't include any of the other files listed above, as long as it contained a frame with this file in it. The other files are completely customizable and can contain any HTML you desire.

Modifying the header

The default skin uses a file name header.html to display the TS logo and colored background. The only place header.html is referenced is in the main skin file, ts.html. You can change the height of the header frame by changing the frameset “rows=” value in ts.html.

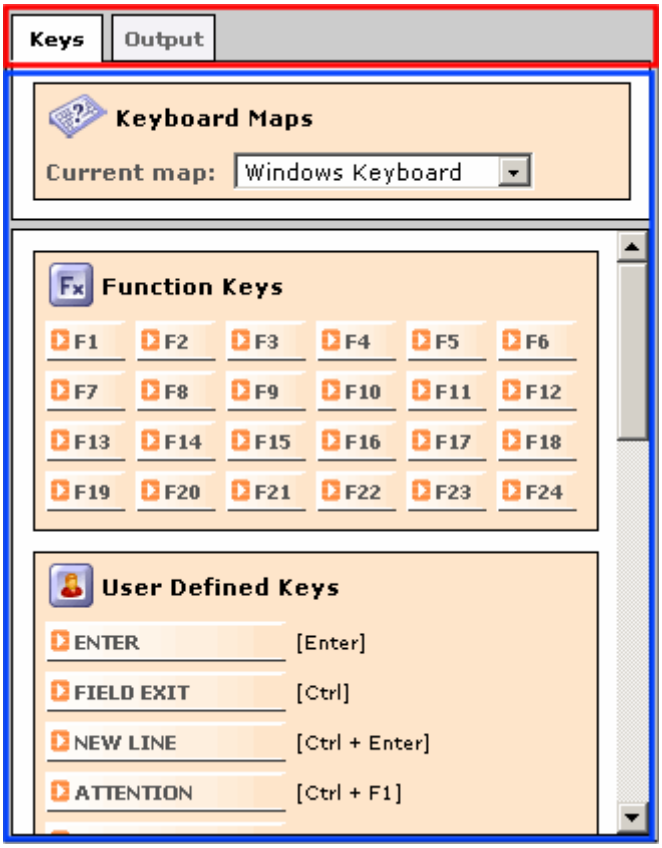
How do I change the throbber image?

The only functional object in header.html is the TS “throbber” logo that lets you know when the server is busy. If you are modifying this file, we recommend that you create an animated throbbing image of your logo to replace the TS logo.

To replace the existing throbber, change the “src” values for the two images with ID’s “throbber0” and “throbber1”. The first image should be replaced with a non-animated image that will be still when the server is not busy. The second, “throbber1”, should be replaced with the animated image.

Modifying the tabs

The default skin uses tabs.html to control what tabs are shown in the tab frame to the left of the terminal frame.



- <ul id="tabs"> -

The tabs.html file contains two <iframe> elements whose visibility is controlled by which tab is shown as active at the top of the tabs frame. By default, the tabs.html hierarchy is as follows:

tabs.html	**tabs.css**	Colors and fonts of tabs.
	tabs.js	Scripts for controlling tabs.
** **	**ws_frames.html**	Output tab frameset.
** **	**key_frames.html**	Keys tab frame set.

aXes Development Kit (ADK)

How do I add a tab?

To add a new tab you will need to modify tabs.html (or tabs_basic.html – the same as tabs.html but without the output tab.)

To add a new tab selection panel to the top of the tabs frame you need to add a new <li> block to the <ul id="tabs"> group block. Copy one of the existing <li> blocks, then change the following bold parameters:

```
<li>
<a href="#" id="YourName"
onclick="setCurrTab(this,'YourNametab'); parent.ts_frame.Terminal.focus();"
> YourName</a>
</li>
```

Save the changes then refresh the ADK to see the effect. Your tab heading will now show in the tabs group. However if you click on the heading you will receive a JavaScript error because there is no tab data to display yet!

To prevent this error we need to add a new to the block. Copy the existing block and paste it directly underneath. Now change the following bold parameters:

```
<div id="YourNametab" style="width:100%;height:100%; display:none; ">
      <iframe src="YourName.html" id="YourNameFrame"
      style="width:100%;height:100%" allowtranparency="true" frameborder="0"
      border="0" scrolling="no" /></iframe>
</div>
```

Make sure that the value for **YourNametab** is the same in both the <li> and the elements.

YourName.html will be the file that contains your tab data. It can contain any HTML you like. If you want to include the style sheet that is used in the existing tabs, make sure the file includes the following line:

```
<link TITLE="Stylesheet" rel="stylesheet" type="text/css"
href="ts_style.css">
```

ts_style.css is the file that controls how the current tabs are displayed.

There is still one more step before the tab addition process is complete. Notice that when you change between the tabs, your tab will always be shown as "active". To change this behavior you need to modify the setActiveTab() JavaScript function at the beginning of tabs.html.

Add the following line to the function:

```
document.getElementById('YourName').className = "";
```

Make sure that **YourName** is the same value you set for the id of the <a> element in the <li> block you added earlier.

Now create a file for YourName.html and the job is complete!

How do I set the default tab?

If you have created a new tab or want to make the "Output" tab the default tab loaded, first make the following changes to the elements in the <ul id="tabs"> group.

Remove the **class="active"** value from the previous default tab, and add that value to the <a> element of the tab you want to be loaded by default.

Then remove the "display: none" property from the tag and add "display: none" to the style attribute of the previous default tab (keytab).

The last step is to change the value of the variable "currTab" that is set at the top of tabs.html to "YourName", as set in the <li> tab elements.

aXes Development Kit (ADK)

How do I remove a tab?

Removing a tab requires the same steps as adding a tab, but in reverse. If you only want to remove the “Output” tab, note that there is already a version of tabs.html called tabs_basic.html that already has this configuration.

To remove a tab, open tabs.html and remove the line of script in setActiveTab() that refers to the tab you want to remove. Next, delete the and <li> blocks that contain the HTML for the unwanted tab.

Can I remove all of the tabs?

You can remove all of the tabs by removing the “tabset” frame from ts.html. When you remove the frame, you also need to update the bodyFrameset’s “cols=” attribute to “0,*”.

You will also need to remove the show/hide tab button from ts_frame.html. Simply delete the block.

Can I make the tabs show at startup?

By default, the tabs are hidden when the login screen is displayed. To make them visible, edit ts.html’s bodyFrameset’s “cols=” attribute to “3,319,*”.

How do I change the width of the tab frame?

The width of the tabs is set so that the “Output” tab values are shown without horizontal scroll bars. If you want to make the tab frame thinner or wider, edit the bold value shown in the following line from ts.html’s tabControl() function:

```
document.getElementById("bodyFrameset").cols = "3,319,*";
```

If you want the tab to show at startup, make sure the new value matches that shown in bodyFramesets “cols=” value.

Can I change the existing tabs styles?

The majority of the existing tabs contents are controlled with the ts_style.css style sheet. Examine the respective HTML file to see which classes you need to change.

The exception is the data list displayed in the output tab when you press the “Refresh” button. Note that this is a mockup version of the actual spooldata.html file. The list’s style is controlled by a file called tsspool.css, which is included in the ADK for customization purposes only.

See “How do I change the Output tabs data list style?” for more detail.

What’s different about the Keys tab?

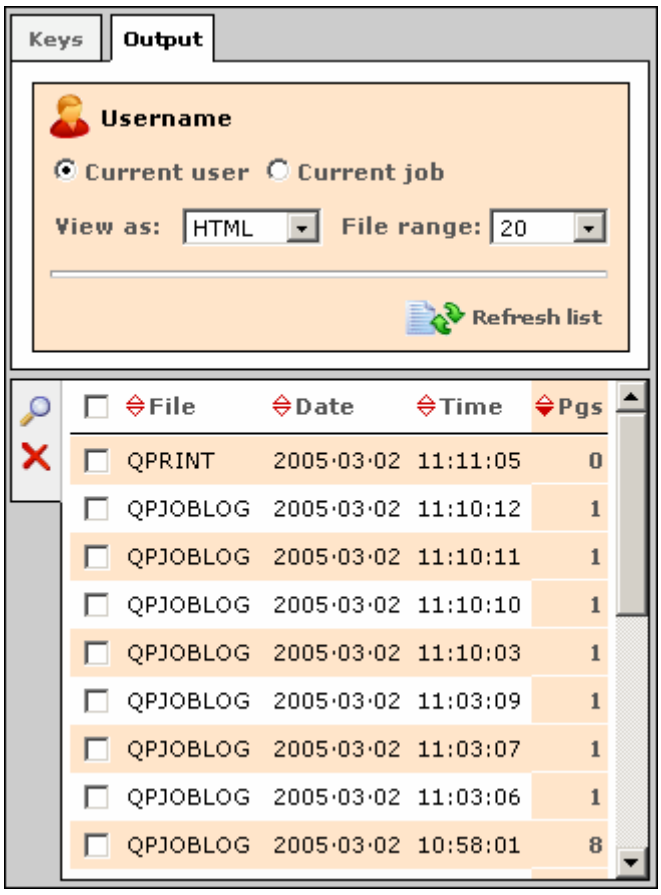
The keys tab contains a frame that is populated with an XSLT transformation. The keyboard maps are XML files that are transformed into HTML by KeyMap.xsl. The XSL file includes a link to ts_style.css so changing class values involves the same process as normal HTML tabs.

See “Customizing keyboard maps” for more information.

How do I change the Output tabs data list style?

You can display a list of test data in the Output tabs list by pressing the “Refresh List” button.

aXes Development Kit (ADK)



For the purpose of this offline kit only, two files from the aXes Web Spooler have been included. These are spooldata.html and tsspool.css. Spooldata.html is a mockup of the list that is normally retrieved. Tsspool.css is a copy of the file normally held in the /ws/skins/ directory.

The Output tab works by requesting a compact version of the spooldata.html file served by the aXes Web Spooler. The request URL is constructed in /ts/skins/ts_spoolpanel.html.

The file ts_spoolpanel.html has been modified for the offline kit so that it only retrieves the mockup version. ***Note that if you edit the ADK version of ts_spoolpanel.html you need to make the following code change before uploading to the host:***

In the refreshList() function change…

```
else {
            alert('You need to start an aXesTS session.');
            loggedIn = false;
            setUser("Username");
            parent.spoolFrame.location.href = "spooldata.html";
            return 0;
}
```

to…

```
else {
            alert('You need to start an aXesTS session.');
            loggedIn = false;
            setUser("Username");
            parent.spoolFrame.location.href = "blank_page.html";
            return 0;
}
```

ts_spoolpanel.html contains two variables that effect the displayed data. “MaxFiles=” which will set the number of records to retrieve (*does not work offline) and “spoolstyle=” which specifies which style sheet to use. This style sheet will be retrieved from the /ws/skins/ directory. By default spoolstyle is set to “tsspool.css”, which is the file included in the ADK for testing purposes.

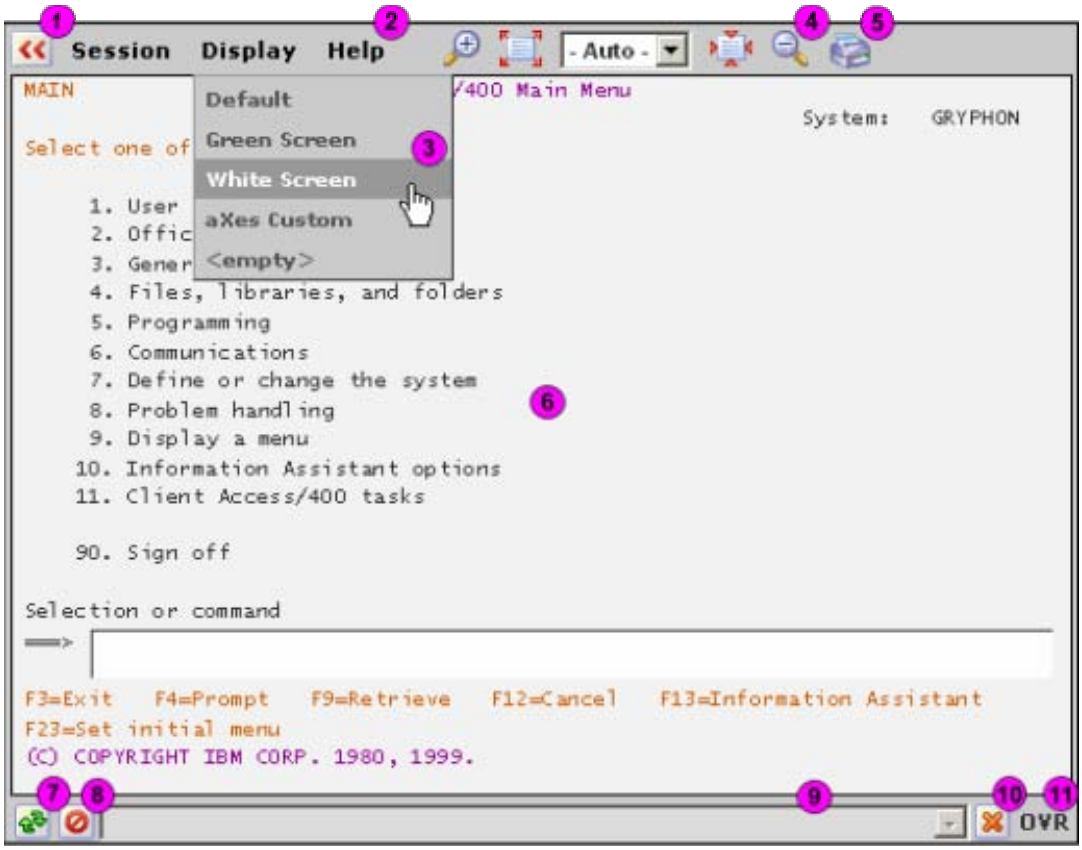
Tsspool.css contains all of the style properties for the spooled file list. If you need to change the name of tsspool.css, you must also edit the following line in the mockup file spooldata.html:

```
<link TITLE="Arterial Stylesheet" rel="stylesheet" type="text/css"
href="tsspool.css">
```

When you are finished modifying tsspool.css, it must be saved in the /ws/skins/ directory.

Modifying the main frame and menus

ts_frame.html is the most important file in the skins directory. It is the only file necessary to being a TS session, and in fact, can be called directly by the browser if you want to totally simplify the skin. Loading just this file removes the need for the header, tabs and other files described in previous sections.



The functional components of ts_frame.html are:

1. Show/Hide tabs button
2. Main menu headings
3. Popup menu items
4. Zoom control buttons
5. Print screen button
6. The terminal frame
7. Refresh button
8. Cancel transaction button
9. Message bar
10. Clear messages button
11. INS/OVR indicator and button

aXes Development Kit (ADK)

The hierarchy of files in ts_frame.html is as follows:

ts_frame.html	**skin_style.css**	Menu colors and fonts etc.
	skin_script.js	Scripts for loading login screen etc.
<iframe id="TermFrame">	Contains the terminal frame, login screen etc.	

What default values can I set in ts_frame.html?

The default values that can be set from ts_frame.html are as follows:

Variable name	Value	Description
VTLogBool	(True/False)	Creates a log of all transactions in a host file. For debugging purposes only.
auto_mode	(True/False)	Sets the AUTO zoom mode on or off.
OVR_bool	(True/False)	Forces TS to start in overwrite rather than insert typing mode.
xpos	No. of pixels	The relative horizontal position of drop down menus to their respective menu headings.
ypos	No. of pixels	The relative vertical position of drop down menus to their respective menu headings.
dflt_css	Filename	The default terminal style sheet filename.
key_map	Filename	The default keyboard map filename.
dflt_login	Filename	The default login screen filename. (Only replace the original file name, not the sslbase variable.)

How do I change the menu colors?

The text fonts and background colors can be modified using the file skin_style.css. For example, if you want to change the background color of the menu frame to white, change the “background-color:” value of the “BODY” rule from “#ccc” to “white”.

To change the color of the menu item text, change the “color:” value of the “.menu_item” style sheet rule.

How do I add a main menu heading?

The first step in creating a new main menu item is to copy one of the existing <li> blocks from within the <ul id=”minitabs”> group and then paste it below the last element.

Next, replace the following bold values with your new menu heading name:

```
<li><a class="menu_item" id="itemname" href="#" hideFocus="true"
       onclick="if (!disabled) showPopup('itemname','itemname',130)"
       onfocus="this.blur()">itemname</a>
</li>
```

When you refresh the ADK you will now see your “itemname” in the menu bar. Clicking on the item at this stage will cause a JavaScript error because there is no data to display in the popup window.

To create a list of items to be displayed in the new menu, you need to create a data block similar to the ones at the bottom of ts_frame.html.

To get started, copy the data block and everything contained within it, then paste it below the linksHTML element. You now need to replace the word “links” in the top two elements in your new block with the **itemname** you used in the menu heading.

You should now have the following data block beneath the links block:

```
<DIV ID="itemnameHTML" class="popup_menu">
<div ID="itemnameHTMLinner" class="popup_menu_inner" >

<a class="popup_item" href="#" hidefocus="true" onfocus="this.blur()"
```

```
  onclick="parent.window.open('http://www.arterialsoftware.com/', 'review')"
 >Arterial Software</a>

<a class="popup_item" href="#" hidefocus="true" onfocus="this.blur()"
   onclick="alert('Contact Arterial Software regarding \naXesTS interface
personalisation.')"
      >&lt;empty&gt;</a>

</div>
</DIV>
```

Because you have created the divs “itemnameHTML” and “itemnameHTMLinner” the menu heading you created earlier now has items to display in the drop down menu. Refresh the ADK and try pressing the new menu item.

You can now edit, delete or replace the popup menu items.

How do I add or remove a popup menu item?

Popup menu items are simply <a> elements that have the classname “popup_item”. You can add or delete as many of these elements as you like. Popup items can be used to open new browser windows, retrieve data from host scripts, or even execute host commands. These applications are beyond the scope of this document.

To remove a popup menu item, find the <a> element that contains the text of the menu item and delete it.

Popup menus will display any HTML you create inside the <DIV id=”itemnameHTML”> block.

Can I specify the length and width of the popup menu?

When you insert a new main menu heading, or modify an existing one, you can specify the width and maximum displayed height of the popup menu it creates.

The onclick event for each menu heading <a> element contains the following JavaScript:

```
onclick="if (!disabled) showPopup('itemname','itemname',130)"
```

The value “130” is the width that the menu will be created with. If your popup menu items are wider than this width then a horizontal scrollbar will be created.

You can also specify a maximum height for your popup menu as follows:

```
onclick="if (!disabled) showPopup('itemname','itemname',130, 300)"
```

By adding the last parameter you are setting a height at which vertical scrollbars will be enforced. So if the contents of your popup menu exceeds this height value, vertical scrollbars will be displayed so you can view all of the menus contents.

Can I remove the zoom control and print buttons?

You can safely remove the HTML that contains the print screen button, but removing the zoom items requires some modification of skin_script.js.

The zoom buttons on the screen are disabled with JavaScript that specifies each item by its ID. If an item is removed without modifying skin_script.js then a JavaScript error will be the result.

To safely remove a zoom button from the skin, search skin_script.js for the ID of the object you want to remove and any object contained within that HTML block. Use “//” to comment out any JavaScript references to those ID’s.

Can I add a header or footer to ts_frame.html?

You can add any HTML you wish to ts_frame.html provided it is contained within a element, and that that element is also contained within the main <div id=” ContentDiv”> block.

Can I add JavaScript to “skin_script.js”?

If you are confident enough with scripting languages to modify the existing JavaScript, then you are free to alter skin_script.js. However, if you want to add JavaScript for use just with your new skin, then it would be wise to include it in the <head> section of your custom skin or in another “.js” file.

aXes Development Kit (ADK)

Customizing keyboard maps

Keyboard maps have been made available for customization so that customers used to different keyboard configurations can maintain there existing layouts. By default there are two keymaps available, an AS/400 terminal based keymap, and a Windows keymap. These two keymaps cover the majority of aXes customers needs.

What is a keymap?

A keymap is an XML file that is read by aXes TS and used to translate key entries into iSeries commands. The default files are located in the /ts/ directory.

A keymap XML file is a collection of <MAP> tags such as:

```
<MAP>
    <KEY>Enter</KEY>
    <DESC>ENTER</DESC>
    <COMMAND>PostData</COMMAND>
</MAP>
```

This <MAP> tag shows the basic format of how all keys are mapped to their respective commands. The <KEY> tag designates the key entry that will run the command specified in the <COMMAND> tag. The <DESC> tag contains a description of the command, and is what is displayed as the text on the buttons in the “Keys” tab.

How do I create a new keyboard map?

As with any aXes customization, first make a copy of the keymap you want to modify and rename it. It is easier to modify an existing keymap rather than start from an empty file.

What are the available keys?

The list below shows the keys that are handled by aXes TS. If a key is not handled by aXes, it is passed through to Internet Explorer. If you want to replace a <KEY> value for a specific command, replace the existing <KEY> tag contents with one of the following values: (Text in brackets is for explanatory purposes.)

LeftMouseButton	**Cancel**
RightMouseButton	**Ins**
MiddleMouseButton	**Del**
BackSpace	**LeftWindows**
Tab	**RightWindows**
Clear	**Apps** (Next to the right windows key)
Enter	**NumPlus**
Shift	**NumMinus**
Ctrl	**Multiply**
Pause	**Space**
Esc	**Add**
PageUp	**Subtract**
PageDown	**Decimal**
End	**Divide**
Home	**F1-F12**
LeftArrow	**Numlock**
UpArrow	**ScrollLock**
RightArrow	**EBCDICZero**
DownArrow	**All Printable Chars** (A, a, B, b, 1, 2 etc)

For example, to replace the Enter key with the Ctrl key in WindowsKeyMap.xml, you need to change these entries:

```
<MAP>
     <KEY>Enter</KEY>
     <DESC>ENTER</DESC>
     <COMMAND>PostData</COMMAND>
</MAP>
```

to...

```
<MAP>
     <KEY>Ctrl</KEY>
     <DESC>ENTER</DESC>
     <COMMAND>PostData</COMMAND>
</MAP>
```

and...

```
<MAP>
     <KEY>Ctrl</KEY>
     <DESC>FIELD EXIT</DESC>
     <COMMAND>FieldExit</COMMAND>
</MAP>
```

to...

```
<MAP>
     <KEY>Enter</KEY>
     <DESC>FIELD EXIT</DESC>
     <COMMAND>FieldExit</COMMAND>
</MAP>
```

Note: You cannot have two entries for the same key, but you can have multiple keys executing the same command.

What about the Shift, Control and Alt keys?

To map these keys in combination with other keys use the "+" sign as follows:

```
<MAP>
     <KEY>Ctrl + F1</KEY>
     <DESC>ATTENTION</DESC>
     <COMMAND>Attention</COMMAND>
</MAP>
```

Note: The order of these combinations is important. The first key in the <KEY> block must be the first key depressed and so forth.

What are the available commands?

Commands are modified in the same manor as keys. The available iSeries commands are:

iSeries Command	Description
PostData	Send the form to the server
NextField	Move to the next field
PrevField	Move to the previous field
Function or F1, F2 etc	Function keys F1 - F12
ShiftFunction or F11, F12 etc	Function keys F11 – F24
FunctionZero	Function key 0
Home	
Noop	No operation. Use this to trap unwanted key presses
Move	

FieldExit	
ToggleSelect	Toggle a selection item (radio button, check box, etc) on and off
SystemRequest	
Refresh	Send a redraw command
DuplicateKey	
Attention	
NewLine	Start a new line in an entry field
ClearField	Clear the field with the focus
ClearScreen	Clear all fields on the screen
InsertYen	Insert the Yen symbol
PA1	
PA2	
PA3	
Help	
DoEnter	
DoRollDown	Scroll down the screen
DoRollUp	Scroll up the screen
DoRollLeft	Scroll to the left of screen
DoRollRight	Scroll to the right of screen
DoPrint	Print
DoRecordBackspace	
DoSLPAutoEnter	
DoForwardEdgeTriggerAutoEnter	

How do I add a keymap to the “Keys” tab?

Adding your new or modified keymap to the “Keys” tab drop down list is a simple process. Open the file /ts/skins/ts_keymaps.html and add the following <option> item to the <select id="keymapselect"> group:

<option value="**MyKeyMap.xml**" >**My Keyboard**</option>

Replace the value of MYKeyMap.xml with the exact filename of your keymap XML file. Then add a description to replace “My Keyboard”.

Refresh the ADK and your keymap will now be functional. When you save your keymap XML file to the /ts/ directory on the host iSeries, you will need to restart the aXes servers in order to access the file.

How do I set the default keymap?

See “What default values can I set in ts_frame.html?” in section 6.3 for information on setting the default keymap.

What is KeyMap.xsl?

KeyMap.xsl is a file used to translate keymap XML files into HTML. You can change the <style> element properties to change the colors of the buttons and backgrounds. Note that each of the default keymap XML files reference KeyMap.xsl, so if you modify the XSL file you should not rename it. Make a backup copy of the file before making any changes as XSL files are very sensitive and can be hard to debug.

aXes Development Kit (ADK)

“Companyone” customization example

The following is a worked example of how a company might customize aXes TS. All of the files are available in /ts/test/example/ in the ADK for your use, and you can see their results by loading the “companyone.html” skin.

Use this guide while studying the contents of the relevant file, and if necessary, compare the file in /ts/test/example/ with the respective file in /ts/skins/ to see what changes have been made.

The /example/ directory is for demonstration purposes only. Your customization should always take place in the /skins/ directory.

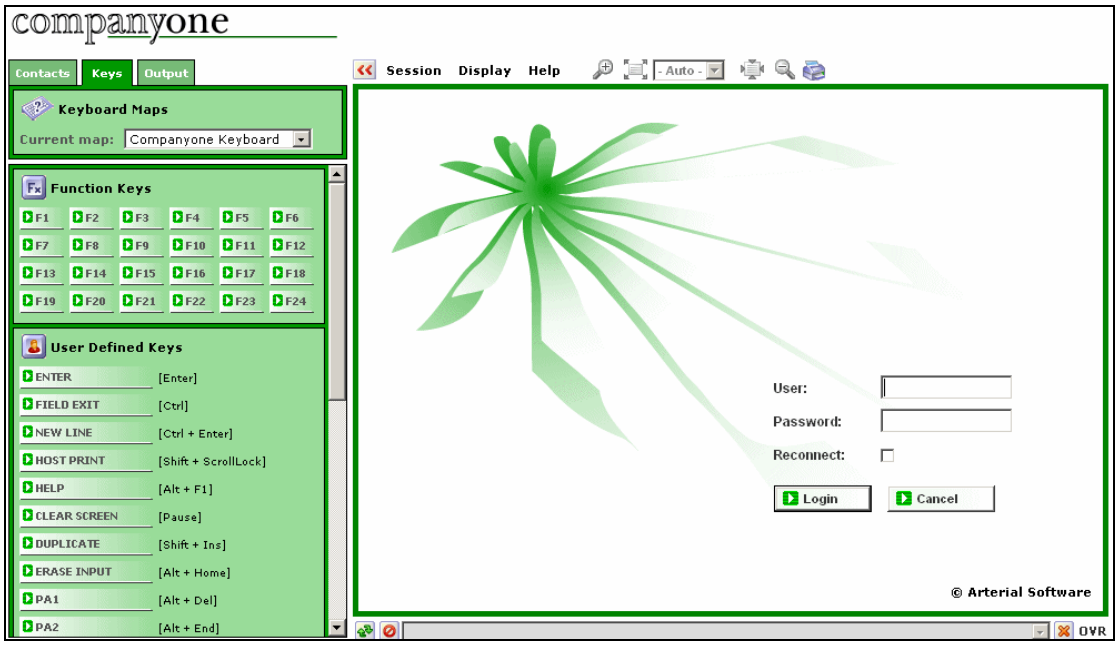
Background

“CompanyOne” had just installed aXes and decided to upgrade the interface of TS to meet their corporate image. They would like to restyle the skin, create a custom terminal style sheet and add a new live data tab.

Getting started

The first step in Companyone’s customization was to download the latest version of the aXes TS ADK from Arterial Software’s download page. Once it was installed on their local PC and running in the browser, they could begin their customization.

The Companyone marketing team provided their developers with a general framework for how they wanted the final result to look, as well as logos and company fonts and colours to use in the skins.



Creating the new skin

A duplicate of ts.html was made in the /ts/test/example/ directory and renamed companyone.html. This value was then added to the list of skins in /ts/test/command.html to make the testing process easier.

Modifying the header (header.html)

The size of the header frame was increased (in companyone.html) to make room for the companyone logo. The background color was changed to white and the borders were removed. The companyone animated logo was inserted in place of ts_anim_150x22.gif.

aXes Development Kit (ADK)

Changing the background color

To fit in with the new header, the grey background around the tabs and menus had to be changed to white. The “background-color:” property of body element in the following files was changed to white:

- vspacer.html
- tabs.css
- midspacer.html
- spacer.html
- skin_style.css

Changing the menu colors

In skin_style.css, the “.bottommenu” class’s background colour was set to white. The variations of the “.menu_item” and “.OVR_item” classes were changed so that the text colour was black and the hover color was “#009900” (dark green).

The “.popup_menu” class background colour was changed to dark green, and the colors of the hover background (“.popup_item:hover”) and popup menu items (“.popup_item”) were changed accordingly.

The border around the main frame was changed to 5px by modifying the elements style property in ts_frame.html.

Modifying the tabs

Before adding the new tab, the existing tabs styles (tabs.css, ts_style.css and tsspool.css) were updated to match the green and white combination of the menus. All of the border widths were increased to match the terminal frame border.

Tabs.html was then edited to include the new tab “Contacts”, which was set to be the default tab. The contents of the Contacts tab was based on the contents of the Keys tab. **Note:** ***The files employee.html and customer.html are mockups only. These could be replaced with links to live data in a real application.***

Customizing the login screen

axes_login.xml was copied and renamed comp_login.xml, and the background colors were set to white. In the new file, the “#loginContainer” rule was changed so that the entry fields were positioned in the bottom right corner of the screen.

The “#advButton” style was modified to include “display:none” so that users could not alter the advanced login fields.

The colors of the images on the login and cancel buttons were changed to green, and a filter was put in the buttons background.

A new image was created to fill the space in the top left corner of the page.

Customizing a new terminal style sheet

A copy of axes_white.css was renamed comp_default.css and added to the two style sheet lists in /ts/test/command.html for offline testing purposes. (Note that comp_default.css needs to exist in both /ts/skins/ and /ts/test/example/ for the ADK to recognize it correctly.)

The main menu transaction was used as a guide for editing comp_default.css. The color values for the items on the screen were changed to black and green variations. (Try each transaction screen to see what changes were made. Use the hover function to see the class names used.)

The new style sheet was then set to be the default style sheet used by changing the dflt_css value in ts_frame.html. It was then added to the list of available style sheets in the block by adding the following code:

```
<a class="popup_item" href="#" hidefocus="true" onfocus="this.blur()"
       onclick="parent.SetTermStyle('comp_default.css')" >Default</a>
```

aXes Development Kit (ADK)

Creating a new keyboard map

The /ts/WindowsKeyMap.xml file was duplicated and renamed /ts/comp_WindowsKeyMap.xml. In the new file the <MAP> blocks containing the values for “Attention” and “SystemRequest” were completely removed so that users would not have access to those buttons.

ts_frame.html’s “key_map” variable was set to “comp_WindowsKeyMap.xml” so that that keymap would be loaded as the default. ts_keymaps.html was then modified to replace the “WindowsKeyMap” option in the selection list.

KeyMap.xsl (comp_KeyMap.xsl) was edited to change the background colors, filter colors and button images. (Note that in this example it is renamed comp_KeyMap.xsl, this is only for example purposes and it should normally be left as is.)

Further testing

Before sending your changed files to the host, you should save as many live transactions from your iSeries application as possible. Make sure you have your terminal style sheet working exactly as you want on every screen.