CA 2E

Web Option User Guide

Release 8.6.00

Contact CA Technologies

Contact CA Support

For your convenience, CA Technologies provides one site where you can access the information you need for your Home Office, Small Business, and Enterprise CA Technologies products. At http://ca.com/support, you can access the following:

- Online and telephone contact information for technical assistance and customer services
- Information about user communities and forums
- Product and documentation downloads
- CA Support policies and guidelines
- Other helpful resources appropriate for your product

Providing Feedback About Product Documentation

If you have comments or questions about CA Technologies product documentation, you can send a message to techpubs@ca.com.

If you would like to provide feedback about CA Technologies product documentation, complete our short customer survey, which is available on the CA Support website at http://ca.com/docs.

Documentation Changes

We made the following documentation updates since the last release of this guide:

- Web Option Considerations (see page 21)—Updated existing and added new considerations.
- Install Web Option (see page 25)—Updated and improved the installation process.
- Configure Web Option (see page 32)—Updated and added new configuration tasks.
- Added new commands:
  - YGENMLS – Generate Markup Language Skeleton (see page 63)
  - YUPGW2EENV – Upgrade Web Option Environment (see page 75)
- Removed the obsolete command *YCOLORTST - Web Option Color/Attribute Testing.*
- Added new control values:
  - YCOLRTO - Column Ratio Multiplier Value (see page 85)
  - YDBGDTA - Debugging Data Flags (see page 86)
  - YBGSSN - Debug Session ID (see page 88)
  - YESCOPT - Screen String 'escape' Option (see page 96)
  - YFKYACC - Function Key Accelerator Key Format (see page 96)
  - YSKLCHK - Check Skeleton Date Frequency (see page 104)
  - YVRTTRM - Virtual Terminal Type (see page 113)
- Removed obsolete control values:
  - YDFTALL - Default Autoload Level for New Skeletons
  - YSRVALL - Auto-Load Level to Load at Server Start
- Web Option Limitations (see page 189)—Added this new Appendix

Contents

Chapter 1: Overview

Web Option is an add-on member of the CA 2E family of products that allows customers to web-enable their CA 2E applications by generating HTML skeleton pages directly from the CA 2E model. A router program runs on the IBM i that controls an interactive job and merges the generated HTML skeleton pages with 5250 runtime data, allowing the application to be executed from a browser. The router also allows non-CA 2E applications to be run in a browser, by dynamically creating an HTML page for each IBM i green-screen, which does not already have a generated HTML page.

This section contains the following topics:

Feature Details

The following list identifies the key features of Web Option:

- Configurable HTML generation from CA 2E model.
- CA 2E screens are automatically identified to Web Option.
- Generated skeleton HTML pages that are merged with runtime data.
- Fully editable using third party HTML editors or text editor.
- Shipped set of HTML default values that can be changed to provide your own corporate look-and-feel to every generated HTML page.
- Just-in-time HTML pages created if no HTML skeleton is available-no need to identify every screen or define which screens are connected.
- IBM i session runs in basic browser window.
- No browser plug-ins.
- No external PC-based software.
- No platform-specific requirements.
- Job and session management controlled on IBM i.

- Fully automatic screen identification if you regenerate and recompile functions using an amended screen header or footer.
- Ability to copy generated HTML source on the IBM i to a folder in the Integrated File System (IFS) for editing.
- Support for User Defined Macros (UDM)

Technical Details

This section contains Web Option technical details; it includes information about the 5250 data stream, HTML generator, Web Option router, Web Option server, and UDM Support.

5250 Data Stream

The IBM i system uses a data stream approach to interact with both dumb terminals and the various types of emulation software that exist for the IBM i. Essentially, data sent from the IBM i to the emulator consists of a stream of encoded characters that describe the screen that is displayed by the emulator. This stream is sent as discrete packets of data, which the emulator receives, interprets, analyzes, and formats into a recognizable screen. When you enter data on the screen and press either Enter or a command key, the emulator encodes this data into the data stream format prior to sending it back to the IBM i.

The data stream normally contains both static data and dynamic data. Static data is text that does not change, such as screen titles. Dynamic data includes all input fields and some output text that may change.

Web Option utilizes the 5250 data stream to run an interactive job with the IBM i. When you access your IBM i from a browser using Web Option, the IBM i is unaware that you are not directly connected to the IBM i and treats your Web Option session as a normal interactive job.

Web Option and the 5250 Data Stream

There are three main parts to the Web Option product:

- HTML generator
- Router
- Server

HTML Generator

The HTML generator generates a Markup Language Skeleton (MLS) page (known simply as a skeleton) for each CA 2E screen (panel). This skeleton contains both standard HTML tags and Web Option-specific tags. Skeletons are generated as members in a source file on the IBM i, where they remain during their use by Web Option. Skeletons can be edited post-generation, either using a third party tool such as Microsoft FrontPage or manually using e.g. SEU or any other IBM i source member editing tool.

The YGENMLS (Generate Markup Language Skeleton) command allows you to generate one or more skeletons for a specific function (one skeleton is generated per screen). The generator identifies the screen from the Data Description Specifications (DDS) source member and retrieves data about it from the CA 2E model. It uses information about the CA 2E function, the display screen, and other model- and system-level information to generate the skeleton source member in a file called YMLSSRC. At runtime, this skeleton is retrieved from YMLSSRC and is merged with live data to produce the HTML page that is sent to the browser.

Web Option Router

The Web Option Router is a program that resides on the IBM i and is called repeatedly from the browser through the IBM HTTP Server. It routes client (browser) requests to the server using data queues.

Web Option Server

The Web Option Server has two parts:

Auditor

The auditor monitors every client request handled by the Router and controls the auditing and reporting of Web Option. It also monitors the memory allocation that the server and router use.

Server

The server consists of several jobs, each of which performs screen identification, 5250-to-HTML merging, and HTML-to-5250 conversion for each client job.

The jobs that make up the Web Option Server normally run in the QSYSWRK batch subsystem and the Web Option Server is initially set up so that it starts automatically each day.

Web Option-related CA 2E Model Values

The following tables lists Web Option-specific model values for CA 2E.

Model Value	Function
YW2ELIB	Holds the name of the Web Option environment

For more information on how the YW2ELIB model value is used, see the section CA 2E model changes.

Web Option Control Values

Web Option has a number of control values, which are similar to IBM i system values or CA 2E model values. Many of these values are used in the generation of HTML source for panels and in the just-in-time creation of HTML pages by the router. Other values hold information necessary for the router, server and HTML generator processing.

For a full description of each value, see the appendix Control Values (see page 81).

Tag Languages

In a markup language (ML) such as HTML, there are a limited number of specific tags, each of which is easily identified by its standard format:

```
< + tag name + optional list of attributes + >
```

For example:

```
<HTML> or <TABLE WIDTH="100%">
```

Browsers recognize these tags and treat them in a particular way. Some HTML tags define the following:

- **Presentation**-how a page looks (either its layout or some of its attributes, such as the font color)
- **Hyperlinks**-links to other documents or to parts of the same document

In the same way, Web Option has a number of tags that are recognized by the router. A Markup Language Skeleton generated for a CA 2E function consists of the following:

- HTML tags - provide much of the presentation elements of the final HTML page.
- Web Option tags - define how live data should be merged into the MLS by the Web Option server at runtime.

At runtime, the server scans the skeleton for Web Option tags. For each Web Option tag it finds, the server performs certain processing, such as replacing the tag with specified live data, thus producing a full HTML page containing live data. It is this HTML page, and not the original skeleton, that is sent to the browser for display.

For a list of the Web Option Markup Language Tags currently in use, see the appendix Markup Language Tags (see page 119). Use this appendix to better understand the HTML skeletons CA 2E generates and to create your own Markup Language skeletons

Chapter 2: Setup and Configuration

This chapter contains the information needed to set up and configure Web Option.

This section contains the following topics:

Minimum System Requirements

Web Option has the following requirements:

- IBM i versions V5R3, V5R4, and V6R1 (IBM i 6.1) are supported
- IBM i must be accessible across a network or through the Web (Internet/Intranet/Extranet), either directly, through a firewall, or through a proxy server

Web Option Considerations

Web Option consists of the following shipped objects:

- An IBM i library known as the Web Option product library, or Y2WEB.

  Y2WEB contains the Web Option non-data objects such as programs, service programs, and job descriptions.

- An IBM i library known as the Web Option LDO library, or Y2WEBVENG.

  Y2WEBVENG contains the English Web Option LDO data objects such as files, message files, and commands.

The Y2WEB and Y2WEBVENG libraries are linked through the use of a data area in Y2WEB named YW2ELDORFA, which contains the name of the Web Option LDO library that is currently being used. They are also linked through the WEB2E_SVR job description in Y2WEB, which includes both Y2WEB and Y2WEBVENG in its initial library list. These objects are used when Web Option environments are created.

Notes:

- CA Provides additional Web Option LDO libraries for languages other than English. These libraries are named Y2WEBVxxx, where xxx is a three letter language code. For example, FRN for French and DTC for German.
- Throughout this document, we use the default names Y2WEB and Y2WEBVENG in code samples and command examples. However, because you can custom-name these libraries or might be using a different language LDO library, be sure that you modify the samples and examples accordingly.

Web Option Environments

When installing Web Option for the first-time, create a Web Option Environment data library, known as the environment library. The environment library is used to hold any user data that you create, such as HTML skeletons, screen cross-reference data, and element customization data. The environment library is linked to the Web Option product and LDO libraries through data areas in the environment library.

The combination of an environment library, LDO library, and product library is referred to as the *Web Option environment,* and it has the same name as the environment library it contains.

In addition to the data areas in the environment library that link to the product and LDO libraries, the environment library contains the WEB2E_SVR job description. WEB2E_SVR includes the Web Option libraries in the following order:

```
<environment-library>
<ldo-library>
<product-library>
```

The Web Option runtime retrieves the data it requires to function from all three libraries, with any data in the environment library overriding the data in the LDO library or the product library.

When you generate an HTML skeleton from a CA 2E model, you specify the name of the environment to use (by referencing the YW2ELIB model value held in the CA 2E model). When you start a Web Option server, you specify the name of the environment to start.

After you install Web Option and create the initial environment, you can create additional environments by using the Create Web Option Environment (YCRTW2EENV) command. You might do this to have separate Development, Test, and Production environments that divide out your development processes. Alternately, you can have different environments for different areas of your business; for example, Marketing, Sales, Finance environments.

Each of these environments can have the same LDO and product libraries, so they would all use the same programs. However, each environment uses its own environment library defaults to provide the look, feel, and functionality of the web pages that it delivers. Each environment can be defined to be accessed using the same HTTP server or different HTTP servers. This allows you to limit who can access each environment and from where – on your Intranet or outside your corporate firewall, for example.

Note: Throughout this document, we use the example environment library name of MYWEBENV for code samples and command examples. You can name your Web Option environment libraries any way you want, but remember to adjust the samples and examples accordingly to your library names.

Using Web Option with Other Languages

As shipped, the YW2ELDORFA data area in Y2WEB comes with the value *Y2WEBVENG*, indicating that the English LDO library is used. Also, the WEB2E_SVR job description in Y2WEB includes Y2WEBVENG in its initial library list. When Web Option environments are created, both objects are copied into the environment library and the WEB2E_SVR job description in the environment library is changed to also include the environment library.

If you need to run Web Option with a language other than English, you can obtain the necessary Web Option LDO library Y2WEBVDTC from CA and then follow these steps:

To change an existing Web Option environment to a language other than English

1. Change the YW2ELDORFA data area in the environment library to the new language:

   ```
   CHGDTAARA DTAARA(MYWEBENV/YW2ELDORFA) VALUE(Y2WEBVDTC)
   ```

2. Change the initial library list (INLLIBL) of the WEB2E_SVR job description in the environment library to include Y2WEBVDTC instead of Y2WEBVENG:

   ```
   CHGJOBD JOBD(MYWEBENV/WEB2E_SVR)
   ```

You then need to change the global Web Option defaults (which apply to every environment that is subsequently created) to use your language.

To change the global Web Option defaults

1. Change the YW2ELDORFA data area in the product library to the new language:

   ```
   CHGDTAARA DTAARA(Y2WEB/YW2ELDORFA) VALUE(Y2WEBVDTC)
   ```

2. Change the initial library list (INLLIBL) of the WEB2E_SVR job description in the product library to include Y2WEBVDTC instead of Y2WEBVENG:

   ```
   CHGJOBD JOBD(Y2WEB/WEB2E_SVR)
   ```

Notes:

- If you change the global Web Option defaults, any existing environments must be changed separately using these previous steps.
- You must also change any other job descriptions you have changed (such as the QBATCH job descriptions in CA 2E models) to use the correct Web Option LDO library.

Merging the Y2WEB and Y2WEBVENG Libraries

Warning! From Release 8.5 onwards, merging the Web Option product and LDO libraries must be performed after Web Option is initialized or upgraded.

If necessary, you can merge the Y2WEBVENG library into Y2WEB. Merging can help reduce the number of libraries and simplify the setup on your IBM i.

To merge the Y2WEBVENG library into Y2WEB

1. Copy all the objects from the Web Option LDO library into the Web Option product library:

   ```
   CPYLIB FROMLIB(Y2WEBVENG) TOLIB(Y2WEB) CRTLIB(*NO)
   ```

2. Delete the YW2ELDORFA data area in Y2WEB:

   ```
   DLTDTAARA DTAARA(Y2WEB/YW2ELDORFA)
   ```

3. Change any job descriptions that you previously changed so that they no longer include the Y2WEBVENG library.
4. Locate the initial library list of the WEB2E_SVR job description in Y2WEB and verify that it no longer includes the Y2WEBVENG library.

Notes:

- The authority of the Y2WEB library must be set to *PUBLIC *CHANGE.
- The authority on YMLSSRC in Y2WEB must be set to have all Data Authorities and Object Authorities of *OBJOPR, *OBJMGT, and *OBJEXIST

External Object Requirements

Web Option uses a number of objects that are not shipped in the Web Option product or LDO libraries, but are shipped in the CA 2E Toolkit ("Toolkit") product library. The Toolkit and CA 2E also require these objects. Y2WEB contains the data area *YRQDOBJLIB* (Required Objects Library), which holds the name of the library which contains these additional objects.

This data area is shipped with a value of *Y1SY*. If your copy of the Toolkit product library is not named Y1SY, before attempting to initialize or run Web Option, you need to change the YRQDOBJLIB data area with this command:

```
CHGDTAARA DTAARA(Y2WEB/YRQDOBJLIB) VALUE(toolkit-library-name)
```

If Web Option is running on an IBM i that does not have the Toolkit installed, create a new library on an IBM i where the Toolkit is installed (for example, *Y2WEBRQOBJ*) and copy the following objects from the Toolkit product library into it:

```
YCMDPRCS1R  *SRVPGM          Command processing
YCMPENCS1R  *SRVPGM          Compression/Encryption procedures
YDBCPRCS1R  *SRVPGM   RPGLE  DBCS processing
YERRPRCS1R  *SRVPGM          Error processing
YGNRPRCS1R  *SRVPGM          Generic procedures (I)
YGNRPRCS2R  *SRVPGM          Generic procedures (II)
YIFSPRCS1R  *SRVPGM          IFS procedures
YMSGPRCS1R  *SRVPGM          Messaging procedures
YPRCDTA1S   *SRVPGM          Data processing
YSORTS1R    *SRVPGM          Sort/search procedures
YSTRPRC1S   *SRVPGM          String processing procedures
```

You need to copy this library to the IBM i where Web Option is to be used. Change the YRQDOBJLIB data area in Y2WEB to have a value of *Y2WEBRQOBJ*. Also, change the WEB2E_SVR job description in Y2WEB to use *Y2WEBRQOBJ* rather than *Y1SY*.

If you do not have a license for the Toolkit, contact CA Support.

Install Web Option

For Web Option itself, there are two libraries that need to be installed on the IBM i. These libraries are shipped as Y2WEB and Y2WEBVENG. In this document, where "Web Option library" or Web Option product library" is used this should be taken to mean Y2WEB.

To install the Web Option libraries

1. Sign on to the IBM i with a user profile that has both *ALLOBJ and *IOSYSCFG special authorities.

   Your library list must include the following libraries (or their renamed or merged equivalents):

   - Y1SY
   - Y1SYVENG
   - QTEMP

   The order of the libraries is not important.

2. To restore the new version of the licensing library YLUSLIB0 and create and update the library YLUSLIB on your system, follow the instructions in your licensing document containing the Web Option Authorization codes.

   Note: If you have not received this authorization document, contact CA to obtain one. You cannot perform any further installation and initialization steps until you are correctly authorized.

3. If a previous release of Web Option is already installed on your IBM i, ensure no one is using it and rename the libraries (for example, to Y2WEBSAV and Y2WEBVSAV).

4. Change the YW2ELDORFA data area in the renamed product library to point to the renamed LDO library. Ensure these renamed libraries are in your library list:

   ```
   RNMOBJ OBJ(Y2WEB) OBJTYPE(*LIB) NEWOBJ(Y2WEBSAV)
   RNMOBJ OBJ(Y2WEBVENG) OBJTYPE(*LIB) NEWOBJ(Y2WEBVSAV)
   CHGDTAARA DTAARA(Y2WEBSAV/YW2ELDORFA) VALUE(Y2WEBVSAV)
   ADDLIBLE LIB(Y2WEBSAV) POSITION(*LAST)
   ADDLIBLE LIB(Y2WEBVSAV) POSITION(*BEFORE Y2WEBSAV)
   ```

5. Restore the shipped Web Option libraries using RSTLIB (Restore Library command):

   ```
   RSTLIB SAVLIB(Y2WEB) DEV(device-name) RSTLIB(*SAVLIB)
   RSTLIB SAVLIB(Y2WEBVENG) DEV(device-name) RSTLIB(*SAVLIB)
   ```

6. Add the restored Web Option libraries to your library list.

   ```
   ADDLIBLE LIB(Y2WEB) POSITION(*FIRST)
   ADDLIBLE LIB(Y2WEBVENG) POSITION(*BEFORE Y2WEB
   ```

The installation process that you perform depends on which of the following situations applies to you; choose only one of the following sets of steps:

Install Web Option for the First Time

If you are installing Web Option for the first time, follow these steps.

To install Web Option for the first time

1. Restore the /WEBOPT directory from the WEBOPT save file in Y2WEB with this command:

   ```
   RST DEV('/qsys.lib/y2web.lib/webopt.file') OBJ(('/webopt'))
   ```

2. Use the Initialize Web Option (YINZW2E) command to run the Web Option initialization.

   This creates all the required objects for Web Option to run correctly (such as user profiles and job descriptions). You are prompted to create an initial Web Option environment data library.

3. Specify *NONE for the Current Web Option library (CURW2ELIB) parameter when you run the YINZW2E command:

   ```
   YINZW2E CURW2ELIB(*NONE) W2ELIB(Y2WEB)
   ```

   As the command processes, you are given the opportunity to change some of the Web Option environment control values and add a job schedule entry. When you are prompted for the YDBFCCS control value:

4. Change the value of Code page from DSPSYSVAL QCHRID.

   Also, you are able to create all the necessary HTTP server objects to enable Web Option to run.

   Note: This command must be run interactively.

5. Change any 2E models so that the YW2ELIB model value refers to the environment library rather than the product library:

   ```
   CHGMDLVAL MDLVAL(YW2ELIB) VALUE(MYWEBENV)
   ```

6. Change any CA 2E model job descriptions and library lists so that they include the environment library in the library list above the new LDO and product libraries.

Upgrade Web Option from r8.5

If you are currently running Web Option r8.5, follow these steps.

To upgrade Web Option r8.5

1. Save your existing copy of the /WEBOPT directory in the IFS; WEBOPT85, for example.
2. Restore the new copy of /WEBOPT from the WEBOPT save file in Y2WEB and then copy any user-defined objects from WEBOPT85 into WEBOPT:

   ```
   RNM OBJ('/webopt') NEWOBJ('webopt85')
   RST DEV('/qsys.lib/y2web.lib/webopt.file') OBJ(('/webopt'))
   MOV OBJ('/webopt85/<object-name>') TODIR('/webopt') (repeat as required)
   ```

   Use the Upgrade Web Option environment (YUPGW2EENV).
3. Specify the name of your existing Web Option environment data library and the name of the new, restored Web Option product library and LDO library:

   ```
   YUPGW2EENV W2EENV(MYWEBENV) W2ELIB(Y2WEB) LDOLIB(Y2WEBVENG)
   ```

   This command upgrades your Web Option environment to the level of the new shipped Web Option product libraries.

   Note: This command must be run interactively.
4. Change any 2E models so that the YW2ELIB model value refers to the environment library rather than the product library:

   ```
   CHGMDLVAL MDLVAL(YW2ELIB) VALUE(MYWEBENV)
   ```
5. Change any CA 2E model job descriptions and library lists so that they include the environment library in the library list above the new LDO and product libraries.

Upgrade Web Option older than r8.5

If you are running a version of Web Option that is older than r8.5, follow these steps.

To upgrade a Web Option installation older than r8.5

1. Save your existing copy of the /WEBOPT directory in the IFS; WEBOPT81, for example.
2. Restore the new copy of /WEBOPT from the WEBOPT save file in Y2WEB and then copy any user-defined objects from WEBOPT81 into WEBOPT:

   ```
   RNM OBJ('/webopt') NEWOBJ('webopt81')
   RST DEV('/qsys.lib/y2web.lib/webopt.file') OBJ(('/webopt'))
   MOV OBJ('/webopt81/<object-name>') TODIR('/webopt') (repeat as required)
   ```

   Ensure that you do not accidentally overwrite the YSCRIPT.JS file, or any other files required by the Web Option runtime.

3. Use the Initialize Web Option (YINZW2E) command to upgrade your version of Web Option and create an initial Web Option environment library.

4. Specify the name of your existing Web Option product library (which you just renamed) for the current Web Option library (CURW2ELIB) parameter:

   ```
   YINZW2E CURW2ELIB(Y2WEBSAV) W2ELIB(Y2WEB)
   ```

5. Specify *COPY for the Data option (DTAOPT) parameter when you are prompted to create an environment library (the Create Web Option environment (YCRTW2EENV) command is automatically prompted and run as part of the upgrade process):

   ```
   YCRTW2EENV W2EENV(MYWEBENV) W2ELIB(Y2WEB) LDOLIB(Y2WEBVENG) DTAOPT(*COPY)
   ```

6. Change the YDBFCCS control value Code page from DSPSYSVAL QCHRID when you are prompted.

7. Delete the Y2WEB and Y2WEBVENG libraries and re-restore them after the YINZW2E processing has completed and you have created an environment library:

   ```
   RMVLIBLE LIB(Y2WEB)
   RMVLIBLE(Y2WEBVENG)
   DLTLIB LIB(Y2WEB)
   DLTLIB(Y2WEBVENG)
   RSTLIB SAVLIB(Y2WEB) DEV(device-name) RSTLIB(*SAVLIB)
   RSTLIB SAVLIB(Y2WEBVENG) DEV(device-name) RSTLIB(*SAVLIB)
   ```

   As part of the upgrade process, the old versions of the shipped skeletons such as YSIGNON, YERROR, and so on, are copied into the new environment library.

8. Perform the following steps if you previously made any changes to the skeletons (such as changing the background or links):

   a. Rename the old Y* skeletons in MYWEBENV to e.g. Z* (for instance, change YSIGNON to ZSIGNON)

   b. Copy the r8.6 YSIGNON skeleton from Y2WEBVENG into MYWEBENV

   c. Make any required changes to this new copy of YSIGNON in MYWEBENV

   Note: If you did not make any changes to the Y* skeletons in r8.1, delete these skeletons from YMLSSRC in MYWEBENV. By default, the shipped Y* skeletons in Y2WEB are used.

Following the upgrade process, the copy of the Web Option Control Values file YW2EVALRFP in the environment library is empty. If you changed any of the control values at the previous release, you need to run the YWRKW2EVAL command and change the displayed default values to match the previous values you used. Your previous values are still available in the copy of YW2EVALRFP in Y2WEBSAV.

9. Change the HTTP server configuration to point to the YROUTER program in the environment library rather than in Y2WEB:
   a. End the Web Option HTTP server instance using the ENDTCPSVR command.
   b. Run the Edit File (EDTF command) to edit the HTTP configuration file:
      ```
      EDTF STMF('/www/y2websvr/conf/httpd.conf')
      ```
   c. Page down until you find the following lines:
      ```
      <Directory /QSYS.LIB/Y2WEB.LIB/>
      ```
   d. Change it to reference the environment library:
      ```
      <Directory /QSYS.LIB/MYWEBENV.LIB/>
      ```
   e. Page down until you locate this line:
      ```
      ScriptAliasMatch ^/WEB2E$ /QSYS.LIB/Y2WEB.LIB/YROUTER.PGM
      ```
   f. Change this line to reference the environment library:
      ```
      ScriptAliasMatch ^/WEB2E$ /QSYS.LIB/MYWEBENV.LIB/YROUTER.PGM
      ```
   g. Press F2 to save your changes and press F3 to exit and restart the Web Option HTTP server instance using the STRTCPSVR command.

      Note: If you are upgrading an r8.1 SP1 PTF1 Web Option library, you need to change any HTTP server configurations that are used with Web Option to remove Y1SY8111 from the following line:
      ```
      SetEnv QIBM_CGI_LIBRARY_LIST "Y1SY8111;Y1SY"
      ```
10. Change any 2E models so that the YW2ELIB model value refers to the environment library rather than the product library:
    ```
    CHGMDLVAL MDLVAL(YW2ELIB) VALUE(MYWEBENV)
    ```
11. Change any CA 2E model job descriptions and library lists so that they include the environment library in the library list above the new LDO and product libraries.

Web Option Menu

To access the Web Option menu, enter the following:

```
YGO MENU(MAIN) FILE(Y2WEB/YDSNMNU)
```

You can also enter YGO MAIN if Y2WEB is above any other library that contains a copy of the YDSNMNU file.

From this menu, you can run various web option commands and web option-related commands. You can also access a Web Option Runtime submenu, which lets you perform various Web Option HTTP server and Web Option Server tasks.

Install the Web Option PC File

If you installed Web Option on your system for the first time, you should have restored the WEBOPT directory from the WEBOPT save file shipped in Y2WEB prior to performing the Web Option initialization processing. If you are upgrading an existing release of Web Option, you should have renamed your existing WEBOPT directory (to WEBOPT85, for example) and then, when you restored it, copied in any objects that you require into the new WEBOPT folder.

This directory is used for two purposes by Web Option:

- To hold a number of basic files required by Web Option at runtime: images for screen headers, the default JavaScript, and Cascading Style Sheet files, for example.
- To hold temporary copies of HTML skeletons while they are being edited by a third party tool.

After installing, check the Permissions on each object in /WEBOPT.

Perform these steps in Client Access

1. Open File Systems for the AS400.
2. Locate the directory WEBOPT in the root directory in the IFS.
3. Right-click on each object in the directory and take the Permissions option.
4. Ensure that Public has at least Read and Write authority. If not, check the Read and Write boxes.
5. Click Apply.

You can edit this HTML file using a basic text editor, such as Notepad, or with a graphical HTML editor, such as FrontPage. When you finish editing the HTML file, use the YPRCSKL command again to convert the edited HTML file back into a skeleton member in YMLSSRC in Y2WEBVENG.

Renaming WEBOPT

If you choose to rename the WEBOPT folder, to *webopt86,* for example*,* you must complete the following steps:

1. Change the value of the Web Option value YMLSFLR to be '/webopt86/', using the YWRKW2EVAL command.
2. Change the following lines in your HTTP configuration to refer to WEBOPT86 instead of WEBOPT:

```
<Directory /WEBOPT/>
AliasMatch ^/WEB2EDOC/(.*) /WEBOPT/$1
```

Files created using PRCSKL

After you generate a skeleton member in the YMLSSRC source file in your environment library, use the Process HTML Skeleton (YPRCSKL) command to convert it to an HTML file in the WEBOPT folder for editing. The HTML file has the same name as the skeleton member in YMLSSRC, but is suffixed with .HTM. For example, a skeleton member in YMLSSRC named *H0001025* is converted in WEBOPT to the file *H0001025.HTM.*

Configure Web Option

After installing Web Option, depending on your settings, you might have to perform certain configuration tasks that verify Web Option is working correctly. Certain defaults are assumed which might vary from machine to machine, such as the main sign-on screen, the language used on the machine, and other factors.

Virtual Terminal Setup

Web Option uses virtual terminals to access the IBM i - when a user initially accesses the IBM i using Web Option, the system assigns a virtual display device (normally called QPADEVnnnn, where nnnn is a number from 1 to 9999) to that Web Option session.

If a virtual display device description already exists and is not active (that is, it is not currently being used by an interactive job), Web Option uses that device description; otherwise, the system automatically configures a new virtual device (up to the maximum allowed by the QAUTOVRT system value). Thus, the QAUTOVRT system value effectively controls the maximum number of concurrent interactive jobs that are allowed. Consequently, if you are going to allow external users to access your IBM i through Web Option using the internet, you may need to update QAUTOVRT to a higher number.

Be aware that the higher the number of virtual terminals that can be auto-configured, the greater the potential security exposure of your system. For more details of the QAUTOVRT system value, see your IBM system manuals.

Although the shipped value of QAUTOVRT is 0, most systems have had this value set to a higher value (e.g. 999). To change the QAUTOVRT system value, use the CHGSYSVAL command:

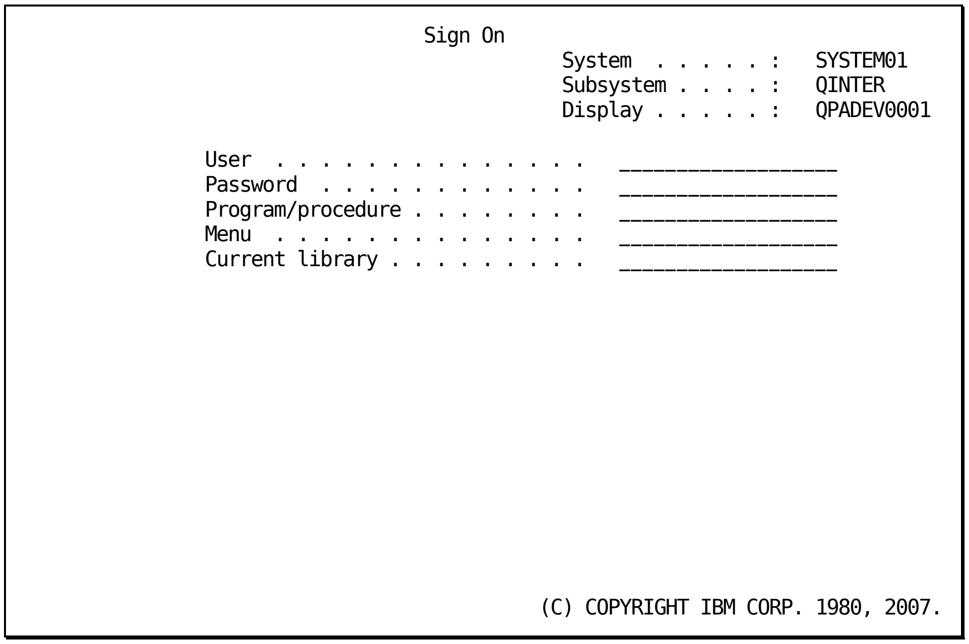
```
CHGSYSVAL SYSVAL(QAUTOVRT) VALUE(*NOMAX)
```

Note: You must have *ALLOBJ authority to change the QAUTOVRT system value.

Sign On Page Setup

Certain special processing occurs within the Web Option runtime when the sign on screen is displayed. Therefore, it is important that Web Option is able to correctly recognize the sign on screen when it appears.

As shipped, Web Option includes an HTML skeleton that matches the default IBM Sign On screen:



```
                                   Sign On
                                               System  . . . . . :   SYSTEM01
                                               Subsystem . . . . :   QINTER
                                               Display . . . . . :   QPADEV0001

                User  . . . . . . . . . . . . . .   ___________________
                Password  . . . . . . . . . . . .   ___________________
                Program/procedure . . . . . . . .   ___________________
                Menu  . . . . . . . . . . . . . .   ___________________
                Current library . . . . . . . . .   ___________________











                                                 (C) COPYRIGHT IBM CORP. 1980, 2007.
```

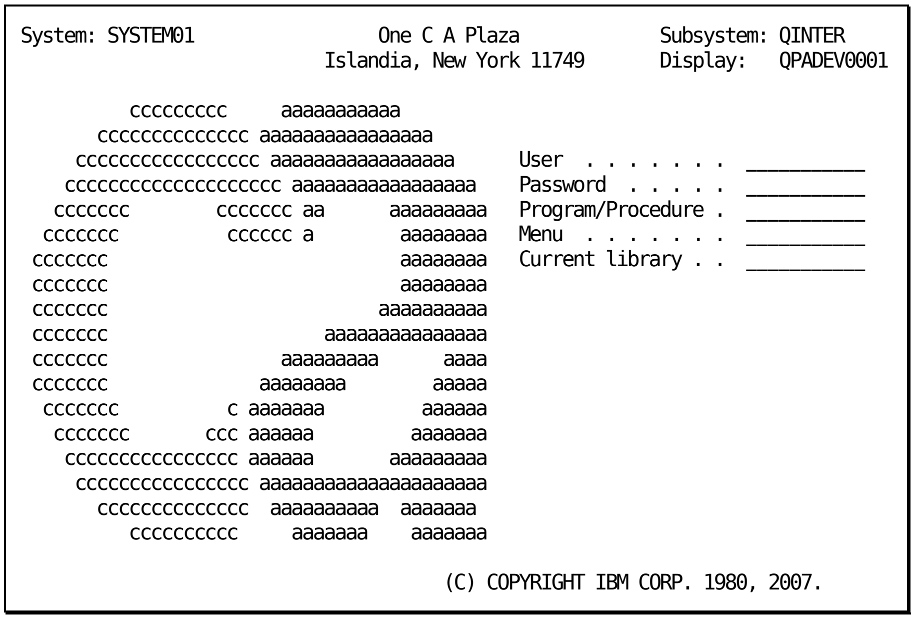
Web Option recognizes this screen because of the 7-character string on row 1, column 36 with a value of *Sign On*. If you modified your IBM Sign On screen, you need to modify the screen cross-reference file or the shipped YSIGNON skeleton to ensure that Web Option correctly recognizes it.

If your IBM Sign On screen does not include the text Sign On text at row 1, column 36, perform the following steps:

To update your Sign On text

1. Determine an alternate piece of text on your screen that uniquely identifies it.

   For example, in the following sign on screen, the 13-byte string *One C A Plaza* at row 2, column 34 could be used:



```
System: SYSTEM01                 One C A Plaza             Subsystem: QINTER
                            Islandia, New York 11749       Display:   QPADEV0001

          ccccccccc     aaaaaaaaaaa
       cccccccccccccc aaaaaaaaaaaaaaaa
     ccccccccccccccccc aaaaaaaaaaaaaaaaa      User  . . . . . . .  __________
    cccccccccccccccccccc aaaaaaaaaaaaaaaaa    Password  . . . . .  __________
   ccccccc        ccccccc aa      aaaaaaaaa   Program/Procedure .  __________
  ccccccc          cccccc a        aaaaaaaa   Menu  . . . . . . .  __________
 ccccccc                           aaaaaaaa   Current library . .  __________
 ccccccc                           aaaaaaaa
 ccccccc                         aaaaaaaaaa
 ccccccc                    aaaaaaaaaaaaaaa
 ccccccc                aaaaaaaaa      aaaa
 ccccccc              aaaaaaaa        aaaaa
  ccccccc          c aaaaaaa         aaaaaa
   ccccccc       ccc aaaaaa         aaaaaaa
    cccccccccccccccc aaaaaa       aaaaaaaaa
     cccccccccccccccc aaaaaaaaaaaaaaaaaaaa
       cccccccccccccc  aaaaaaaaaa  aaaaaaa
          cccccccccc     aaaaaaa    aaaaaaa

                                       (C) COPYRIGHT IBM CORP. 1980, 2007.
```

2. Use the YCVTRCO (Convert between row/col/offset) command in Y2WEBVENG to convert the row/column of the identifying text into an offset value:

   ```
   YCVTRCO ROWCOL(02 034)
   ```

   In this case, YCVTRCO returns a value of 0114.

3. Use the YWRKF (Work with Database File) command to edit the YSCRXRFP Screen cross-reference file in Y2WEBVENG:

   ```
   YWRKF FILE(Y2WEBVENG/YSCRXRFP)
   ```

4. Change the record which has a SID value of SYS0000001 so that the Screen ID offset (SIDOFF), Screen ID data (SIDDATA) and Screen ID length (SIDLEN) fields match the new screen identification data:

   ```
   Screen ID offset: 0114
   Screen ID data: One C A Plaza
   Screen ID length: 13
   ```

Note: On the default signon screen, the USERID and PASSWRD fields are at row 6, column 53 (offset 0453) and row 7, column 53 (offset 0533) respectively. If your changed signon screen has these fields at different positions, you must modify the YSIGNON skeleton to match these new positions.

To modify the skeleton

1. Determine the correct row/column to use for the two fields. To do this, from a green screen session, place your cursor in the first position of each field and note the row/column numbers displayed in the bottom right corner of the display.
2. Use the YCVTRCO command to convert the row/column values for the USERID and PASSWRD fields on your changed signon screen to offset values.
3. Use WRKMBRPDM on YMLSSRC in Y2WEB to modify the member YSIGNON (an HTML skeleton for the Sign On screen). Find the source lines for the following html:

   ```
   <input type=”text” name=”_F0453U” size=”10” value=”(_v0453)”>
   <input type=”password” name=”_F0533U” size=”10”>
   ```
4. Modify the 2E tags, indicated in bold, to match the offsets of the USERID and PASSWRD fields.

Web Option Running Apache HTTP Server DBCS

In order for Web Option to function correctly with the Apache HTTP Server and DBCS, the following directive must be added at the top of the HTTP server instance configuration file:

```
CGIDbcsOptimization Off
```

Note: This server directive is not documented in the IBM documentation.

Non-English Web Option LDO libraries

If you are using a non-English Web Option LDO library, use the YWRKF command to display the contents of the record in YSCRXRFP where Screen ID (SID) is SYS0000013 and Sequence number (SEQ) is 2:

```
YWRKF FILE(Y2WEBVENG/YSCRXRFP)
```

The Screen ID data (SIDDATA) field must match the value of the CAE0103 message in the QCPFMSG message file. This record links the YAUTSBM skeleton to the Display Program Messages screen. The record also ensures this skeleton is displayed only when the message being displayed is a warning message about multiple users signing on using the same user profile. If it does not match, change the Screen ID data (SIDDATA) and Screen ID length (SIDLEN) fields to match the CAE0103 value.

CA 2E Model Changes

For any CA 2E models where you need to generate skeletons, change the model library list to include the Web Option environment library, Web Option LDO library, and Web Option product library using the YADDLLE (Add Library List Entry) command. Also, ensure all three libraries exist in the library list for the Job Description specified for the YCRTJOBD model value (normally this is the QBATCH job description in the model library):

```
YADDLLE LIB(Y2WEB) POSITION(*LAST) LIBLST(MYMDL) UPDJOBD(*YES)
YADDLLE LIB(Y2WEBVENG) POSITION(*BEFORE Y2WEB) LIBLST(MYMDL) UPDJOBD(*YES)
YADDLLE LIB(MYWEBENV) POSITION(*BEFORE Y2WEBVENG) LIBLST(MYMDL) UPDJOBD(*YES)
CHGJOBD JOBD(MYMDL/QBATCH)
```

Chapter 3: Identifying Application Screens

Every generated skeleton corresponds to an individual screen within a CA 2E function. In a Web Option client job, each time a screen is displayed by the CA 2E application program that is running, Web Option retrieves the skeleton that was generated for that screen, merges variable data from the screen with the skeleton, and sends the resulting HTML page to the client browser.

For this process to work, there must be a link that associates the CA 2E screen and the skeleton that was generated for the screen. This association is done using a *Screen Identifier* (SID):

Real SID

For display file DDS generated from CA 2E, the SID is a 10-byte string of characters inserted into the DDS and output as a hidden (non-displayed) field .

Virtual SID

For all other types of display file DDS, the SID is a reference on a cross-reference file that refers to one or more uniquely identifying pieces of text on the screen. This virtual SID must be created by using the YWRKSCRXRF command.

Screen identification is controlled by two files held in Y2WEBVENG:

Screen cross-reference (YSCRXRFP)

This file holds the data that links a particular screen to its SID. A record is added to this file when a CA 2E screen that contains a SID is generated and also when a screen is manually identified. If Web Option encounters a screen that does not contain a real SID at runtime, it accesses the file to see if the screen has been manually identified.

Markup Language Skeleton cross-reference (YMLSXRFP)

This file holds data that links a particular SID to a generated skeleton. A record is added to this file when a skeleton is generated for a display file using the YGENMLS command. Web Option accesses this file at runtime, when it has retrieved a SID (either directly from the screen or by checking for a manually identified screen), to retrieve the name of the skeleton with which the screen data is merged.

This section contains the following topics:

Screen Identifiers in CA 2E Generated DDS

This section discusses using a *SCREEN ID in the header/footer of a CA 2E display file to identify screens. If you do not want (or are unable) to use a header/footer containing the *SCREEN ID, follow the instructions in the section Manually Identifying a Screen in this chapter.

A Define screen format (DFNSCRFMT) function named *STD SCREEN HEADINGS(MLS) is included in the *Standard Header/Footer file. This function is identical to the *STD SCREEN HEADINGS(CUA) function, except that it contains the *SCREEN ID field at row 2, column 2. The inclusion of the *SCREEN ID field in this header causes a SID to be generated into the DDS source of any function that uses this header/footer function.

The *STD SCREEN HEADINGS(MLS) header part of the header/footer appears as follows:

```
*PROGRAM   *PGMMOD                                 DD/MM/YY HH:MM:SS
*SCREEN ID            *STD SCREEN HEADINGS (MLS)
```

You can change and regenerate a function that currently uses the default *STD SCREEN HEADINGS(CUA) header/footer to use the *STD SCREEN HEADINGS(MLS) header/footer; the green-screen looks exactly the same and the positions of the fields on the screen do not change.

If you want to use the *STD SCREEN HEADINGS(MLS) header/footer function as the default for all your 2E functions, you need to set the new shipped MLS Header/Footer, *STD SCREEN HEADINGS(MLS), to be the default:

1. F into the *Standard header/footer file from the Edit Database Relations panel.
2. Z into *STD SCREEN HEADINGS(MLS) and press F7=Options.
3. Set the *Use as default for functions* flag to 'Y'.

If you only want certain functions to contain the *SCREEN ID in the device design, then do not make the above change and simply select the *STD SCREEN HEADINGS(MLS) for those functions before generation.

When CA 2E generates the display file and the header/footer contains the *SCREEN ID field, the SID is generated into the DDS as a non-display field, so that it is not visible to users. However, the HTML generator can see the SID in the DDS source and uses it to determine the name of the HTML skeleton to generate. It is also visible to the Web Option server, which uses it at runtime to determine the HTML skeleton with which to merge data.

The CA 2E display file DDS generator inserts the SID into the DDS source in the format C2E*nnnnnnn* (where nnnnnnn is a unique number assigned to that screen). Each screen in a multi-screen function has a different SID. For example, an EDTRCD3 function has four screens, Key Screen, Detail Screen 1, Detail Screen 2 and Detail Screen 3, each of which has a different SID. This results in 4 different skeletons being created when the YGENMLS command is run over this function.

For example, the DDS for a display file can include the following:

```
     * *SCREEN ID
    A                                          2  2'C2E0001025'
    A                                               DSPATR(ND)
```

If you do not use the *STD SCREEN HEADINGS(CUA) header/footer, but have defined your own header/footer, you can add the *SCREEN ID field to that header/footer. The *SCREEN ID field can be included anywhere in the header section.

Once your functions have been regenerated to include the *SCREEN ID field, you are ready to generate HTML skeletons for your screens.

Manually Identifying a Screen

If you cannot use the new *STD SCREEN HEADINGS(MLS) header/footer and cannot add the *SCREEN ID field to another header/footer that you use with your CA 2E display file functions, you must manually identify your CA 2E screens to Web Option before you can generate skeletons. You can also manually identify non-CA 2E screens and generate skeletons for them (if you have the source for the display file) or write your own skeleton by hand.

Manual identification requires selecting a piece of text that always appears on the screen each time the program is called, such as the screen title. Use the Work with Screen Cross-references (YWRKSCRXRF) command to manually identify this piece of text as the virtual SID and create an entry in the screen cross-reference file. Even though a SID is not actually generated into the source member, the generator can access the screen cross-reference file to determine whether this screen has been identified.

For example, if you have a CA 2E display file that does not contain the *SCREEN ID field, perhaps because it was originally generated prior to you installing Web Option, and you do not wish to regenerate it, you must use the YWRKSCRXRF command to create the screen cross-reference record before you generate the HTML skeleton using the YGENMLS command. The HTML generator automatically uses the cross-reference record you have just created to determine the name of the skeleton to generate.

Alternatively, you may have an IBM menu screen that you want to display in the browser. Even though the menu does not have a DDS source member, you can still create a SID to identify the menu, then manually create your own HTML skeleton for it (using an HTML editing tool).

The YWRKSCRXRF command allows you to create both a record in the screen cross-reference file YSCRXRFP and a record in the Markup Language skeleton cross-reference file YMLSXRFP. The screen cross-reference file holds data that links an identified screen with its SID. The markup language cross-reference holds data that links a SID with a generated skeleton name.

Chapter 4: Generating HTML Skeletons

Once you have created a Screen ID for your function screens, use the YGENMLS (Generate Markup Language Skeleton) command to create the MLS for that function. This chapter presents details for using the YGENMLS command are given here, all other commands are detailed in the appendix Commands (see page 55).

This section contains the following topics:

Overview

The YGENMLS command executes a data-driven generator program that analyzes the DDS for a function and uses the following Web Option data sources to generate the HTML skeleton for the panel:

- Markup Language Skeleton Syntax file (YMLSSYNRFP)

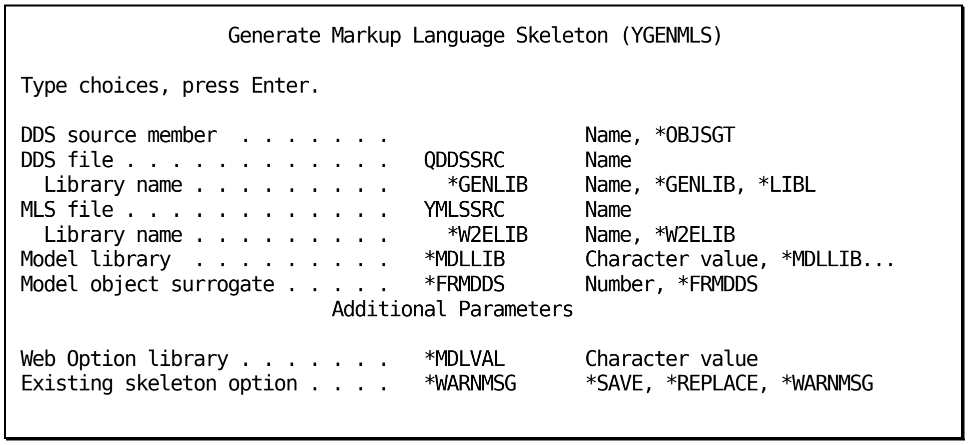
  This file contains the core of the Web Option Markup Language Syntax generator. For every element (input field, output field, text, and record format definition) that appears in a single screen, there is a corresponding entry (or entries) in this file, which define how that element is generated.

- Web Option control values (held in YW2EVALRFP)

  This file contains a number of control values that define how certain elements of the MLS are generated.

You can modify the data in all the above files to customize the look and feel of your generated HTML pages.

The YGENMLS command allows you to identify a function by object surrogate or by the name of the generated DDS source member.



```
                  Generate Markup Language Skeleton (YGENMLS)

 Type choices, press Enter.

 DDS source member  . . . . . . .                  Name, *OBJSGT
 DDS file . . . . . . . . . . . .   QDDSSRC        Name
   Library name . . . . . . . . .     *GENLIB      Name, *GENLIB, *LIBL
 MLS file . . . . . . . . . . . .   YMLSSRC        Name
   Library name . . . . . . . . .     *W2ELIB      Name, *W2ELIB
 Model library  . . . . . . . . .   *MDLLIB        Character value, *MDLLIB...
 Model object surrogate . . . . .   *FRMDDS        Number, *FRMDDS
                           Additional Parameters

 Web Option library . . . . . . .   *MDLVAL        Character value
 Existing skeleton option . . . .   *WARNMSG       *SAVE, *REPLACE, *WARNMSG
```

The parameters are as follows:

DDS Source Member (MBR)

Specifies the DDS source member that contains the generated DDS of the function you want to generate a markup language skeleton for.

Note: This is a required parameter.

***OBJSGT**

The DDS source member name is retrieved by analyzing the function specified by the OBJSGT parameter

member-name

Specify the DDS source member name. The source member should exist in the file and library specified by the DDSFIL parameter.

Note: You cannot specify MBR(*OBJSGT) if OBJSGT(*FRMDDS) is also specified.

DDS Source file (DDSFIL)

Specifies the name and library of the DDS source file that contains the DDS source member for the generated function. This file defaults to QDDSSRC.

The possible library values are:

***GENLIB**

The library is assumed to be the generation library for the model specified in the MDLLIB parameter.

***LIBL**

All libraries are checked until a match is found.

library-name

Specify the name of the library to be searched.

MLS Source file (MLSFIL)

Specifies the name and library of the MLS source file that is the target for the generated MLS source. This file defaults to YMLSSRC.

The possible library values are:

***W2ELIB**

This command uses the Web Option library specified in the YW2ELIB model value for the CA 2E model specified in the MDLLIB parameter.

library-name

Specify the name of the library to be searched.

MDLLIB (Model library name)

Specifies the model name used to generate the function for the required generated MLS source.

The possible library values are:

***MDLLIB**

The model library is the first one in the current library list.

model-library-name

The name of the CA 2E model can be entered explicitly.

OBJSGT (Function Surrogate)

Specifies the object surrogate for the function that requires MLS generation.

***FRMDDS**

The object surrogate is retrieved by analyzing the DDS source in the member specified by the MBR parameter

object-surrogate

Specify the object surrogate for the generated MLS function. The function must exist in the model library, specified by the MDLLIB parameter.

Note: You cannot specify OBJSGT(*FRMDDS) if MBR(*OBJSGT) is also specified.

W2ELIB (Web Option library)

Specifies the name of the Web Option product library.

The possible values are:

***MDLVAL**

The command uses the Web Option library specified in the YW2ELIB model value for the CA 2E model specified in the MDLLIB parameter.

library-name

The name of the Web Option product library can be entered explicitly.

MBROPT (Existing skeleton option)

Specifies the action to take if the skeleton being generated already exists in the specified MLS file.

The possible values are:

***WARNMSG**

If the YGENMLS command is being run within an interactive job, an inquiry warning message (Y2I1019) is sent. Depending on the response to this message, the existing skeleton is either saved or replaced. If the YGENMLS command is being run within a batch job, the existing skeleton is saved before the new skeleton is copied into the specified MLS file.

***SAVE**

The existing skeleton is saved before the new skeleton is copied into the specified MLS file. The existing skeleton has the same name as the newly generated skeleton, but with the first character changed to Y. So, if a skeleton called H0001745 is being generated and a previously generated version of H0001745 already exists in the specified MLS file, the existing version is renamed Y0001745 before the new version is copied into the specified MLS file. If a saved skeleton called Y0001745 already exists in the specified MLS file, it is deleted.

***REPLACE**

The existing skeleton is replaced with the new skeleton.

HTML Generation Processing

When you execute the YGENMLS command, it first looks for a real SID in the DDS for each panel (that is, a string in the format C2Ennnnnnn in the controlling record format for the panel). If the generator cannot find a real SID for a panel, it checks for a manual-identification record on the Screen cross-reference file.

If no SID is found for a panel, an error message is sent indicating that a SID (either a real SID generated into the source or a virtual SID record in YSCRXRFP) must be created in order to create a skeleton.

Each time a SID is found, the Markup Language Skeleton cross-reference file YMLSXRFP is accessed to retrieve/create the skeleton name and a separate source member (the skeleton itself) is generated into YMLSSRC.

If no record is found in YMLSXRFP for the SID, a new skeleton name is assigned for the screen in the form H*nnnnnnn* (where *nnnnnnn* is the numeric portion of the SID).

Finally, the name skeleton is generated as a member in YMLSSRC. Depending on the value of the MBROPT parameter, you may be prompted to overwrite an existing version of the skeleton.

Edit Generated Skeletons

Although generated skeletons work in their raw format, you may want to edit them (to add images or to change the layout of a page). However, you do not need to edit a generated skeleton; once the skeleton has been generated, it is immediately available for use. If a Web Option session starts and accesses the screen for which the skeleton was generated, the skeleton displays.

Generated skeletons can be edited in the following ways:

- Using Screen Entry Utility (SEU)

  You can edit the skeleton source member in YMLSSRC directly. This method of editing is very simple, but does not provide any means of displaying how the HTML looks, and consequently is normally only used where textual changes need to be made to the skeleton (such as correcting spelling mistakes).

- Using a PC-based editor

  This method requires you to copy the skeleton into the IFS (you can use the YPRCSKL command). Once you copy a skeleton to the IFS, you can access and edit it using a basic text editing tool such as Notepad or using a specialized HTML editing tool (Microsoft FrontPage or Macromedia DreamWeaver). See the appendix “Commands” for information about the YPRCSKL command.

  Note: If you use YPRCSKL to copy the skeleton to the IFS for editing using a third-party tool, you must remember to use YPRCSKL to copy the skeleton back into YMLSSRC prior to using it. Web Option uses the member in YMLSSRC at runtime, not a copy in the IFS.

When a Web Option session accesses a skeleton for the first time following a restart of the Web Option Server, the skeleton loads from the file into dynamic memory. Any subsequent access of the skeleton retrieves the version in memory. Consequently, any changes made to a skeleton once it has been accessed by a Web Option session are not visible to Web Option users until the Web Option server is restarted.

Change the HTML Skeleton Name for a Function

In some cases, you may want to change the HTML skeleton name to something meaningful. For example, if you copy the HTML member over to a PC to edit it using an HTML editor, then you may want to rename the file from H0001076 to EDTCUST1, so that you can easily see what function the file relates to without interpreting the skeleton name.

To change the HTML skeleton name, use the YWRKSCRXRF command to edit the cross-reference details for that function panel. You can edit the name and textual description for the skeleton.

Externalization of Text for Multi-language Applications

The extent of externalization that can be done for a Web Option application has includes the following Web Option control values:

YFKLST

The YFKYLST control value determines the surrogate number in the xxVLLSP file of the '*ALL values' LST condition for the '*Web Option *CMD key' 2E model field. This list contains values for all the command keys which cannot be explicitly specified in a 2E action diagram and is used by Web Option if YLNGOPT(*MULTIPLE) is specified, to provide localized text for these command keys. If '*MDLDFT' is specified, a default value of 1003204 will be used.

YSFSLST

The YSFSLST control value determines the surrogate number in the xxVLLSP file of the '*ALL values' LST condition for the '*Web Option *SFLSEL' 2E model field. This list contains values for all the default subfile select options cannot be explicitly specified in a 2E action diagram and is used by Web Option if YLNGOPT(*MULTIPLE) is specified, to provide localized text for these options. If '*MDLDFT' is specified, a default value of 1003217 will be used.

Chapter 5: Running Your Web-Enabled Application

To run a web-enabled application, you must start a Web Option session from a browser (using the IBM HTTP Server), which starts an interactive job on a specified IBM i.

This section contains the following topics:

HTTP Server - General Information

The IBM HTTP Server controls interaction between the Web (Internet/Intranet/ Extranet) and the IBM i. The IBM HTTP Server is a process that listens for requests coming in from external clients, such as web browsers.

Make a request to the IBM i IBM HTTP Server by typing a URL into the address bar of a browser (which is connected to an IBM i across a network or across the Web) in the following format:

```
http://as400-name[:port-number][/path]
```

```
That is:
```

```
http://MYAS400
http://myas400:80/images/image1.gif
http://myas400:1025/Welcome.html
```

Several different instances of the HTTP Server can run at the same time. An instance describes each separate server when a single IBM i starts multiple Web servers.

When an HTTP server instance receives a request from a client, it returns a file to the client (such as an image file or a static HTML page) or it calls a program on the IBM i to do specific processing based on the request. The type of action taken by the HTTP server instance depends on its configuration.

HTTP server instance configuration is done using a configuration file that is read by the HTTP server instance when it is started (using the STRTCP (Start TCP/IP) or STRTCPSVR (Start TCP/IP Server) command). The configuration file consists of a number of server directives-specialized instructions that determine what type of requests the HTTP server instance should listen for, on which port it should listen, and how it should process them. You can define several different HTTP server instances, each using different configuration files to process different types of requests.

Note: You can also do these tasks using the IBM Web Administration for i5/OS interface. For information on how to do this, please refer to the appendix *Web Administration for i5/OS Interface*.

HTTP Server - Web Option Information

When you initialize Web Option for the first time, the YCRTW2EHTTP (Create Web Option HTTP server) is automatically prompted, to enable you to create an HTTP server instance. By default, this HTTP server instance has the same name as the Web Option environment for which it is being created. When the HTTP server instance is started, a job of the same name starts running in the QHTTPSVR subsystem.

When the HTTP server job receives Web Option requests from the browser (that is, requests that match the specified URL format that follows), it calls the Web Option router program YROUTER that communicates with the Web Option server program YSERVER to either initialize a new Web Option session (start a new interactive job) or reconnect to an existing interactive job. When YSERVER finishes processing the request, a new HTML page is created and sent back to YROUTER (which then sends the HTML page to the client and ends).

The HTTP server configuration specifies that the HTTP server instance listen at the port specified in the YCRTW2EHTP command when the HTTP server instance was created (the default port value is 4100), and treat any requests that come in the following format as Web Option requests (that is, requests to call the Web Option router program YROUTER):

```
http://myibmi:4100/web2e
```

The HTTP server configuration file for e.g. HTTP server instance MYWEBENV can be found at the following location in the IFS:

```
/www/mywebenv/conf/httpd.conf
```

Further information about the meanings of the directives in the HTTP configuration file can be found in the HTTP Server for IBM i Webmasters Guide. Although the previous configuration directives are the defaults used when Web Option is initialized, you can edit the HTTP configuration, for instance, to allow access to more than one Web Option environment (a production and a testing environment). See the appendix “Commands” for information about creating and editing HTTP server instances with the YCRTW2EHTP (Create Web Option HTTP server) command.

Web Option URL

A Web Option session can be started by typing the following URL into the address bar of a browser window:

`http://`*`myibmi`*`:4100/web2e`

myibmi

The name of the IBM i where you have installed Web Option.

This makes the initial call to the Web Option router program that initiates an interactive job and starts a conversation between the browser and the IBM i.

You initially see the Web Option sign-on screen (unless you have configured Web Option to allow specific user sign-on or unless you have specified a default Web Option sign-on user profile). If you enter a valid IBM i user ID and password, an interactive job runs on the IBM i, and you can continue to press buttons and enter data on the Web Option screens (the conversation continues between your browser and the IBM i through the HTTP server instance until you close the session).

Close the session either by closing the browser window or by executing the W2E_CLOSE command using a button in the browser window (you can execute this command by clicking Close on the Web Option sign-on page).

Running a Web Option Session

When you are ready to run your application and a browser is active, you can call any of your functions and all panels display in the browser. You do not need to have generated HTML for all function panels-those that have not been identified have a basic HTML page created just-in-time by the router. This page is built using the default Page level Web Option values for such things as Background Color, Text Color, and Page Title. Field level attributes are retrieved from the HTML Attributes file, YHTMATRP. For more information about how this file is used, see the chapter “Setup and Configuration”.

The JIT GUI screen actually bears the closest resemblance to the original green screen, since the translation is done straight from the 5250 data stream. However, if you want to customize your screens, and do not want only the basic default HTML attributes, you should generate skeletons for the panels using the YGENMLS command and then modify the skeletons.

Important! Due to lack of monospaced fonts such as Courier, number fields may not be aligned exactly for a columnwise comparison by position.

How Web Option Processes a Request

When you press Enter on a Web Option HTML page, the following processing is invoked to process the request:

1. The HTTP server receives the request from the browser and calls the Web Option router (as specified in the HTTP Configuration).
2. The Web Option router retrieves any data entered into input fields on the HTML page, encrypts it and sends it to the Web Option server. It then waits for a response from the Web Option server, containing an HTML page to send to the HTTP server. If it does not receive a response from the Web Option server within the time interval specified in the **YRTRMWT** Web Option control value, it immediately returns an error page to the HTTP server and ends.
3. The Web Option server receives the request from the Web Option router and checks if the length of time since this client session last made a request exceeds the **YDFTSTO** Web Option control value. If so, it immediately returns the YTIMEOUT page to the Web Option router (go to step 9).
4. The Web Option server decrypts the data entered on the HTML page and converts it into 5250 data-stream format and sends the data to i OS. If it does not receive a response from i OS within the time specified in the **YSVRMWT** Web Option control value, it immediately returns the YERROR page to the Web Option router (go to step 9).
5. i OS processes the data and passes it to the application program as if the user had entered the data into the green-screen.
6. The application program processes the data and displays a new screen (or possibly the same screen with different data or with an error message displayed, and so on).
7. i OS retrieves the screen data and sends it to the Web Option server.

8. The Web Option server retrieves the screen data, identifies the screen, merges the screen data with the associated skeleton page (or builds a JIT page if the screen has not been identified), compresses the HTML page and sends the page to the Web Option router.
9. The Web Option server returns to DEQW status, waiting for further requests from the Web Option router.
10. The Web Option router receives the HTML page from the Web Option server, decompresses it, passes it to the HTTP server and ends.

The data is passed between the Web Option router, the Web Option server, and i OS through a number of data queues.

Appendix A: Commands

This appendix contains information on Web Option commands. All Web Option commands are shipped in the Y2WEBVENG library with *PUBLIC *USE authority.

This section contains the following topics:

YCRTW2EENV - Create Web Option Environment

When you run the YCRTW2EENV command, the environment library is created and environment-specific data from files in the specified product and LDO libraries is copied into versions of those files in the environment library (and optionally deleted from the product and LDO files if DTAOPT was *MOVE).

Parameters

The parameters for the command are as follows:

Web Option environment (W2EENV)

Specifies the name of the Web Option environment to be created. This is a required parameter.

Web Option library (W2ELIB)

Specifies the Web Option product library that should be linked to the Web Option environment specified in the Web Option environment prompt (W2EENV parameter).

***W2ELIB**

The first Web Option product library in the library is used.

library-name

Specify a Web Option product library.

Web Option LDO library (LDOLIB)

Specifies the Web Option LDO library which should be linked to the Web Option environment specified in the Web Option environment prompt (W2EENV parameter).

***W2ELIB**

The LDO library specified for the Web Option product library specified in the Web Option library prompt (W2ELIB parameter).

library-name

Specify a Web Option LDO library.

Data option (DTAOPT)

Specifies whether data from the Web Option product and LDO libraries should be copied or moved into the data library specified in the Web Option environment prompt (W2EENV parameter).

If DTAOPT(*INIT) is specified, the Web Option environment is created in an initialized form, with data from some control files copied into the data library, but data from user files not copied. The following table details of which files have data copied.

If DTAOPT(*COPY) is specified, data from all the Web Option files is copied to the data library.

If DTAOPT(*MOVE) is specified, data from all the Web Option files is copied to the data library and user data is then removed from the Web Option product and LDO libraries.

Note: DTAOPT(*MOVE) should be specified only if you intend to create a single Web Option environment. If you are planning to create multiple Web Option environments that use the same Web Option product and LDO libraries, you should specify either *INIT or *COPY for this parameter.

The following table shows the list of Web Option files for which data is copied using the different values for the DTAOPT parameter:

		*INIT	*COPY	*MOVE
Y2WEB				
	YCLSDFNP	C	C	NSD
	YDBGDTAP	E	E	EAD
	YELMCSTRFP	C	C	NSD
	YMLSDRFRFP	-	-	-
	YMLSFMTRFP	-	-	-
	YMLSHDRRFP	C	C	C
	YMLSSRC	-	-	-
	YMLSSYNRFP	C	C	C
	YSCRELMRFP	E	C	EAD
	YSCRIPT	E(1)	C	EAD(1)
	YW2EAUDP	E	E	EAD
	YW2EUSRRFP	E	C	EAD
	YW2EVALRFP	C	C	C
Y2WEBVENG				
	YMLSSRC	E	C	EAD
	YMLSTXTRFP	-	-	-
	YMLSUDTP	C	C	NSD
	YMLSXRFP	E	C	NSD
	YSCRXRFP	E	C	NSD

E = Empty file copied

C = File copied (including data)

- = File not copied

NSD = Non-system (user) records deleted after copy

AD = All records/members deleted after copy

(1) Except the DEFAULT member

***INIT**

Only data from Web Option control files is copied into the data library, to create an initialized Web Option environment.

***COPY**

Data is copied from the Web Option product and LDO libraries into the data library. No data is removed from those libraries.

***MOVE**

Data is copied from the Web Option product and LDO libraries into the data library and user data is then removed from those libraries.

YCRTW2EHTP - Create Web Option HTTP Server Instance

This command allows the creation of the HTTP server objects needed to access IBM i from a browser and run Web Option sessions. To run this command, the user must be signed on with a profile that has both *ALLOBJ and *IOSYSCFG special authorities. The following objects are created:

- An HTTP server instance that processes Web Option requests from the browser. This HTTP server uses the global HTTP server defaults (which can be viewed/edited by using the CHGHTTPA command) as well as having its own configuration file, which is located at /www/mywebenv/conf/httpd.conf.
- A Service Table Entry for the specified port number, designated as a Web Option port. This entry can subsequently be viewed by using the WRKSRVTBLE command.

Once the HTTP server instance and configuration are created, they can be started by running:

STRTCPSVR SERVER(*HTTP) HTTPSVR(http-server-instance)

Note: You can also start the HTTP server by using the IBM Web Administration for i5/OS interface. For details on how to do this refer to the appendix Web Administration for i5/OS Interface.

Once both the HTTP server instance and the Web Option server are running, the Web Option can be accessed from a browser by entering the correct URL in the browser address bar.

For example, if the Web Option environment is called MYWEBENV and the IBM i is called MYIBMI and you have taken the default values for the HTTPSVR and PORT parameters on this command, you should execute the following on your IBM i:

```
STRTCPSVR SERVER(*HTTP) HTTPSVR(MYWEBENV)
YSTRW2ESVR W2EENV(MYWEBENV)
```

Then the following URL can be used to access the IBM i from a browser using the Web Option with the following URL:

http://myibmi:4100/web2e

Parameters

The following are the Create Web Option HTTP Server Instance (YCRTW2EHTP) parameters:

W2ELIB (Web Option library)

This required parameter is used to create an HTTP server instance.

HTTPSVR (HTTP server instance)

The name of the created HTTP server instance and the instance must not already exist.

***DFT**

An HTTP server instance Y2WEBSVR is created.

HTTP-server-instance-name

Specify a name for the HTTP server instance

PORT (Port number)

The port number is used by the created HTTP server instance.

Note: The port is defined in the HTTP configuration specified in the HTTPCFG parameter. When the HTTP server instance is started, either through the STRTCP or the STRTCPSVR command, the HTTP server instance listens at the specified port number for requests.

***DFT**

The HTTP server instance uses the default Web Option port number 4100.

port-number

Specify a port to use. The valid range of ports is 1 - 65535, but you should specify a port number above 1024, since port numbers below 1024 are reserved for other TCP/IP applications. Ports 8080 and 8008 are commonly used by proxy servers and should also not be used.

Usage Details

The YCRTW2EHTP command is prompted for use when the Web Option is initialized. If the HTTP server instance is not created, it can be used at any future time to create additional HTTP server instances.

YCVTRCO - Convert between Row/Column/Offset

The YCVTRCO command is a utility to calculate the offset for a row/column position. Conversely, it can be used to convert an entered screen offset to its equivalent row/column.

Within generated HTML skeletons, Web Option tags and data are defined by their screen offset, where the top-left corner of a 24 * 80 screen is offset 1 and the bottom-right corner of a 24 * 80 screen is offset 1920. Because the concept of screen offsets is not one commonly used by IBM i developers or users, this command is supplied to easily provide a user-friendly conversion.

The conversion algorithm from row/column to offset and vice versa is:

```
OFFSET = ( ( ROW - 1 ) * 80 ) + COLUMN )
```

Parameters

The following are Convert Between row/col/offset (YCVTRCO) parameters:

OFFSET (Field Offset)

Offset to convert to row/column

***RC**

Convert to offset from row/column. If this value is entered then values must be entered for the row and column in the ROWCOL parameter

offset-value

Enter a value for the screen offset. This must be between 1 and 1920. The command returns a value for the row and column.

ROWCOL (Field Row/Column)

Row and column to convert to offset value

*0 *0

Convert to row/column from offset. If this value is entered then a value must be entered for the OFFSET parameter.

row-value column-value

Enter a value for row and for column. Row must be between 1 and 24 and column must be between 1 and 80. The command returns a value for the screen offset.

Usage Details

This command is a user utility, and returns a message to the caller. The message ID is W2I0001 and is shipped in the YW2EMSG message file. As shipped, the first-level text for W2I0001 is:

```
Screen offset &1 = Row &2, Column &3
```

where &1 is defined as (*CHAR 4) and &2 and &3 are both defined as *CHAR 2.

YEDTMLSSYN - Edit MLS Syntax

Allows the user to edit various sections of Markup Language Skeleton (MLS) syntax. For more information on using this command, see the appendix “Markup Language Skeleton Customization”.

Parameters

The following are Edit MLS Syntax (YEDTMLSSYN) parameters:

OBJGRP (Object group)

Specifies the name of the MLS object group to be Edited

***ALL**

Display all the MLS object groups for editing

object-group-name

Display only the selected MLS object group for editing

MLCDE (Markup language)

Specifies the name of the Markup Language

***HTML**

Edit only the HTML data within the MLS Syntax file

***JAVA**

Edit only the Java data within the MLS Syntax file

***PCML**

Edit only the PCML data within the MLS Syntax file

Note: The *JAVA and *PCML MLS Syntax data is used by the Enterprise Java Bean (EJB) Generator for CA 2E and is not used by Web Option. If you are not using the EJB Generator, you should not edit *JAVA or *PCML MLS Syntax data.

Usage Details

To have Edit capability when using this command, you need to be signed on as a User profile with *ALLOBJ authority. If the profile does not have sufficient authority, only the Display option is visible.

YENDW2ESVR - End Web Option Server

The End Web Option server (YENDW2ESVR) command ends an instance of the Web Option server job. See the Start Web Option server (YSTRW2ESVR) command for more details.

Parameters

W2ELIB (Web Option library)

Specifies the Web Option product library that is being processed by the server job.

Y2WEB

The Web Option server currently processing the Y2WEB Web Option product library is ended.

library-name

Specify a Web Option product library.

FORCE (Force end of previous job)

Specifies whether an instance of the Web Option server job already running should be ended automatically. This parameter is ignored if an instance of the Web Option server job is not currently running.

***NO**

If an instance of the Web Option server job is already running, this command returns error message W2E9001 and does not end the current instance of the Web Option server job.

***YES**

If an instance of the Web Option server job is already running, it is ended.

Usage Details

If you specify FORCE(*NO) and one or more client jobs are running, an error message displays. To end the Web Option server, you would then need to execute the command a second time, specifying FORCE(*YES).

YGENMLS – Generate Markup Language Skeleton

The YGENMLS command allows the user to generate one or more Markup Language Skeletons (MLS) for use with CA 2E Web Option for a specified display file. This command allows the user to identify display file by CA 2E function surrogate or by the name of the display file DDS source member. It works with a CA 2E-generated DDS, with a manually-created DDS, or a third-party tool generated DDS.

Parameters

MBR (DDS source member)

Specifies the source member that contains the DDS for the display file where you want to create a markup language skeleton.

Note: You cannot specify MBR(*OBJSGT) if OBJSGT(*FRMDDS) is also specified.

***OBJSGT**

The DDS source member name is retrieved by analyzing the function specified by the OBJSGT parameter.

member-name

Specifies the DDS source member name. The source member must exist in the file and library specified by the DDSFIL parameter.

DDSFIL (DDS source file)

Specifies the name and library of the of the DDS source file that contains the DDS source member for the generated function. The file defaults to QDDSSRC.

The possible library values are:

***GENLIB**

The library is the generation library for the CA 2E model specified in the MDLLIB parameter.

***LIBL**

All libraries in the user and system portions of the job's library list are searched until the first match is found.

library-name

Specify the name of the library to be searched.

MLSFIL (MLS file)

Specifies the name and library of the of the MLS source file that is the target for the generated MLS source. The file defaults to YMLSSRC.

The possible library values are:

***W2EENV**

The Web Option environment specified in the W2EENV parameter is used.

library-name

Specify the name of the target library for the generated MLS source.

MDLLIB (CA 2E model library)

Specifies the name of the CA 2E model that contains the function to generate a markup language skeleton.

***MDLLIB**

Special value indicating the CA 2E model library is the first one in the current library list.

***NONE**

Special value indicating that the display file was not generated from a CA 2E model or that the CA 2E model is not available on this system. In this case, you must enter a CA 2E function type in the FUNTYP parameter.

library-name

Specify the name of the CA 2E model library.

OBJSGT (Model object surrogate)

Specifies the object surrogate for the CA 2E function to create a markup language skeleton.

Note: You can not specify OBJSGT(*FRMDDS) if MBR(*OBJSGT) is already specified.

***FRMDDS**

The object surrogate is retrieved by analyzing the DDS source in the member specified by the MBR parameter.

object-surrogate

Specifies the object surrogate of the function to generate a markup language skeleton. The function must exist in the model library, specified by the MDLLIB parameter.

FUNTYP (Function type)

Specifies the type of function for which the DDS source member contains the display file specifications. This is used by the HTML generator to determine which record formats within the display file should be processed by this command.

This value is valid only if MDLLIB(*NONE) is specified.

CA 2E-function-type (*DSPFIL, *DSPRCD...)

Special value indicating that the function type is one of the 12 CA 2E function types for which a display file is generated.

The function types are:

■ *DSPFIL	Display file
■ *DSPRCD	Display record
■ *DSPRCD2	Display record 2
■ *DSPRCD3	Display record 2
■ *DSPTRN	Display transaction
■ *EDTFIL	Edit file
■ *EDTRCD	Edit record
■ *EDTRCD2	Edit record 2
■ *EDTRCD3	Edit record 2
■ *EDTTRN	Edit transaction
■ *PMTRCD	Prompt record
■ *SELRCD	Select record

One of these values should only be selected when the function was generated from a CA 2E model but the model is not available (for instance, when the model is on a different IBM i).

***AUTO**

Special value indicating that the display file source contains special CA 2E Web Option Key Instructions which define the record formats to be processed.

function-type-name

Specifies a function type name which has previously been defined in the YMLSFMTRFP file.

W2EENV (Web Option environment)

Specifies the name of the Web Option environment.

***MDLVAL**

The library is specified as the YW2ELIB model value in the CA 2E model specified by the MDLLIB parameter.

library-name

Specifies the name of the Web Option environment.

MBROPT (Existing skeleton option)

Specifies the action to take if the skeleton being generated already exists in the specified MLS file.

***WARNMSG**

If the YGENMLS command is being run within an interactive job, an inquiry warning message (W2G1019) is sent. Depending on the response to this message, the existing skeleton will either be saved (see below) or replaced. If the YGENMLS command is being run within a batch job, the existing skeleton is saved (see below) before the new skeleton is copied into the specified MLS file.

***SAVE**

The existing skeleton is saved before the new skeleton is copied into the specified MLS file.The existing skeleton has the same name as the newly generated skeleton, but with the first character changed to *Y*. Therefore, if a skeleton called H0001745 is generated and a skeleton with that name already exists in the specified MLS file, the existing version is renamed to Y0001745 before the new version is copied to the specified MLS file. If a saved skeleton named *Y0001745* already exists in the specified MLS file, it is deleted.

***REPLACE**

The existing skeleton is replaced with the new skeleton.

***PRV**

The value that was used with the previous generation request is used. You can determine this value by checking the contents of the YMLSGENOPT data area that is created in QTEMP when the YGENMLS command is first run within a job. If you specify MBROPT(*PRV) on the first execution of YGENMLS within a job, *WARNMSG processing is used.

Usage Details

The YGENMLS command can be run from a command line or from the Work with Screen Cross-references (YWRKSCRXRF) screen. The command is also available as an option from within a CA 2E model, and is available in the following areas within the CA 2E model:

- As subfile option 'H' from the EDIT FUNCTIONS panel.
- As subfile option 'H' from the DISPLAY ALL FUNCTIONS panel.
- As subfile option 94 from the Edit Model Object List (YEDTMDLLST) screen.
- As subfile option 40 from the Display Model Usages (YDSPMDLUSG) and Display Model References (YDSPMDLREF) screens.

YINZW2E - Initialize Web Option

This command performs release-specific initialization, including a reset of the Web Option server (see YSTRW2ESVR - Start Web Option Server (see page 71) for details), and an initialization of all Web Option system and environment variables. Additionally, it creates a Web Option user profile, enabling the Web Option system to function correctly.

Note: This command can only be run by a user with both *ALLOBJ and *IOSYSCFG special authorities.

Parameters

CURW2ELIB

Specifies the name of the current Web Option library being used, which you have already renamed.

***NONE**

Indicates that you are loading Web Option onto your system for the first time.

New Web Option library (W2ELIB)

Specifies the name of the Web Option library being installed. This is the name of the library that was restored from media.

Y2WEB

The library restored from media is called Y2WEB.

library-name

Specifies the name of the library restored from media.

Usage Details

The YINZW2E command must be run after the Web Option product has been licensed for the given release, but before any processing has taken place or any users have attempted to use Web Option, either to generate new skeletons or to access an IBM i from a browser.

For full details on using the YINZW2E command, see the section Install Web Option (see page 25).

YPRCSKL - Process HTML Skeleton

This command enables a generated HTML skeleton to be converted from an IBM i source member to an HTML file in the WEBOPT directory in the root file system. While generated skeletons must exist as source members for the Web Option to use them, they are not easily editable in that format. Any editing of an HTML skeleton on the IBM i must be done with a text-based editor that does not allow easy cut and paste options and that does not allow the user to see how the HTML page will actually look in the browser.

Consequently, the YPRCSKL command can be used to convert the skeleton to a format that can be easily edited.

Parameters

The following are Process HTML Skeleton (YPRCSKL) parameters:

SKELETON (Skeleton name)

Specifies the name of the processed skeleton. This is the same as the member name within the YHTMSRC file.

skeleton-name

Specifies the name of the markup language skeleton being processed.

***SELECT or *S**

Allows the user to select a skeleton from a list of skeletons in the YMLSSRC file.

OPTION (Processing Option)

Specifies the type of processing you want for the specified skeleton.

***CVTTODOC (Convert To Document)**

This option performs a conversion from a source member in YMLSSRC to an HTML file in the WEBOPT directory. After selecting this option, the HTML document exists in the folder; the document can be accessed from a PC and edited using a graphical HTML editor or text-based editing tool.

***CVTTOSKL (Convert To Skeleton)**

Following editing of the HTML skeleton, this option can be used to convert the edited HTML file back into a source member in YMLSSRC.

REPLACE (Replace existing skeleton/file)

Specifies whether an existing target object is replaced. If OPTION(*CVTTODOC) is specified, the target object is the HTML file in the Web Option folder. If OPTION(*CVTTOSKL) is specified, the target object is the member in YMLSSRC in the Web Option library.

***NO**

If the target object already exists, it is not replaced, and an error message is sent.

***YES**

If the target object already exists, it is replaced.

DLTDOC (Delete document)

Specifies whether the HTML file in the IFS is deleted after it is converted back into a member in YMLSSRC. If the conversion process is unsuccessful, this parameter is ignored and the HTML file is not deleted from the IFS

This parameter is only valid if OPTION(*CVTTOSKL) is selected.

Note: Specifying DLTDOC(*YES) ensures that 'stray' copies of skeletons are not retained in the IFS after they are edited using a 3rd-party HTML editor. This eliminates confusion about whether the member in YMLSSRC (which is used by Web Option at runtime) is up-to-date.

***YES**

If the HTML file has successfully been converted back into a member in YMLSSRC, the HTML file is deleted from the IFS.

***NO**

The HTML file is not deleted from the IFS following a successful conversion into a member in YMLSSRC.

Usage Details

Once the Web Option Control Values are set, you may need to edit the HTML skeletons. However, if you do need to do so, map a network drive to enable you to access a skeleton from a PC.

Select the skeleton you want to edit and use the YPRCSKL command to convert it from a member in Y2WEB/YMLSSRC to a PC file:

```
YPRCSKL SKELETON(skeleton-name) OPTION(*CVTTODOC)
```

When you have finished editing the skeleton and have saved it, use the YPRCSKL command on the IBM i to convert the HTML file back to a source member in Y2WEB/YMLSSRC:

```
YPRCSKL SKELETON(skeleton-name) OPTION(*CVTTOSKL) DLTDOC(*YES)
```

Note: At runtime, Web Option does not read skeletons directly from the YMLSSRC file, they are loaded into memory when first needed and then accessed from there. Therefore, changes made to a skeleton that is already loaded into memory are not seen at runtime until you restart the Web Option server, which reloads the skeletons from YMLSSRC into memory.

YSCPRCV - Script Recovery Action

YSCPRCV defines how an error within a script process should be handled. If it is set to *RECOVER*, the Web Option server attempts to recover from the error automatically. If it is set to *ERRPAGE*, the error page defined in the YSCPERR Web Option control value displays. If it is set to *ENDJOB*, the user's job is ended and the YERROR page displays.

Usage Details

ERRPAGE	(*ERRPAGE|*RECOVER|*ENDJOB)

YSRVALL

Auto-load level to load at server start and YDFTALL - Default Autoload level for new skeletons in the “Control Values” appendix for more information.

YRTVW2EVAL - Retrieve Web Option Value

This command is meant mainly for internal use by Web Option application programs. It allows the retrieval of any of the Web Option control values stored in the YW2EVALRFP file. It can be used only within a CL program.

Parameters

W2EVAL (Web Option control value name)

Specifies the 7-character name of a Web Option control value.

VALUE (CL variable for W2EVAL (80))

Specifies an 80-character field that holds the returned actual value for the specified Web Option control value

YSTRW2ESVR - Start Web Option Server

The Start Web Option Server (YSTRW2ESVR) command is used to start an instance of the Web Option server.

Note: If you execute the YSTRW2ESVR command for a specified Web Option environment and a Web Option server instance is already processing that environment, the new instance attempts to end the current instance. However, if there are active Web Option client jobs running, the command returns error message W2E9001 and neither ends the current Web Option server instance nor starts a new Web Option server instance. See the details of the End Web Option Server (YENDW2ESVR) command for information about ending Web Option server jobs.

The server job controls the Web Option runtime environment in which the Web Option router program runs. It also controls the running of the specific jobs accessed through the Web Option router and performs clean-up operations when these jobs are ended. The Web Option server job also performs all audits processing for Web Option. See the details of the Audit type (AUDIT) parameter for more information on Web Option auditing.

Parameters

The following are Web Option server (YSTRW2ESVR) parameters:

W2ELIB (Web Option library)

Specifies the Web Option product library processed by the server job.

Y2WEB

The Web Option server is started and processes the default Y2WEB Web Option product library.

library-name

Specifies a Web Option product library

AUDIT (Audit type)

The Web Option router program writes audit data when the Web Option jobs starts and finishes or when errors occur during the running of those jobs. Audit data can be used to see who accessed the IBM i using the Web Option, when they accessed the IBM i and how they accessed the IBM i.

***FILE**

Audit data is written to the Web Option Audit file YW2EAUDP. Data is written to this file sequentially, but it can be viewed in a number of ways using the various logical files built over YW2EAUDP.

***MSGQ**

Audit data is written to the Web Option server message queue WEB2E_SVR as individual messages.

***NONE**

No audit data is written.

CLEAR (Clear audit record)

Specifies whether any previous audit data is cleared prior to new audit data being written by this Web Option server job.

***NO**

Audit data written by this job is appended to audit data from any previous Web Option server jobs.

***YES**

Audit data from previous Web Option server jobs is deleted.

RESTART (Restart Web Option server)

Specifies whether the Web Option environment should be restarted. Restarting the Web Option environment causes the Web Option server and auditor jobs to re-initialize themselves with the latest control values and skeletons: it does not affect any Web Option client jobs that are currently running. These jobs continue to run, although they use any changed control values.

When the Web Option environment restarts, the following data is re-initialized.

The object groups within the MLS Syntax file that are used by both the MLS generator and by the JIT processor are reloaded from file. These object groups are: *HTTP, *HTML, *HEAD, *BODY, *HEADER, *FOOTER, and *BODYEND, plus any linked object groups.

All Web Option control values are reloaded from file.

Note: A change to the YNBRSVR Web Option control value takes effect only when the Web Option server is started with RESTART(*NO) .

***NO**

The Web Option server is started as normal.

***YES**

If a Web Option server is currently running, it is re-initialized using any changed control values and skeletons. If a Web Option server is not currently running, it is started as normal.

INSTVALS (Instance Start up Value)

Specifies additional startup values to be used for this Web Option server instance. These values are used to override any control values defined for this Web Option environment.

Note: These Instance startup values remain in effect until the Web Option server instance is ended using the End Web Option Server (YENDW2ESVR) command or until it is restarted using the Start Web Option Server (YSTRW2ESVR) command specifying RESTART(*YES) and with either INSTVALS(*DEFAULT) or a non-blank INSTVALS parameter value.

Specify the instance startup values using the following format:

INSTVALS('-name value -name value')

-name

Defines the 7-character name of a Web Option control value in lower case and immediately preceded by a minus-sign.

value

Defines the override value to use for this Web Option server instance.

Specify multiple Instance startup values, with each name/value pair separated by blanks. For example:

```
INSTVALS('-ynbrsvr 1 -yurltyp *ABSOLUTE -yrtrmwt 30')
```

Any Web Option control value not specified in the INSTVALS parameter uses its default value stored in the YW2EVALRFP file. Invalid or nonexistent values are ignored.

Note: The YDFTTTL and YDFTUSR Web Option control values cannot be overridden using the INSTVALS parameter. The YNBRSVR Web Option control value is ignored if RESTART(*YES) is also specified.

blank-value

If RESTART(*YES) is specified, any instance startup override values that are currently in effect will remain in effect. If RESTART(*NO) is specified, the Web Option server starts using default values.

***DEFAULT**

If RESTART(*YES) is specified, any instance startup override values that are currently in effect are cleared and the Web Option server instance is restarted using default values. If RESTART(*NO) is specified, the Web Option server starts using default values.

instance-startup-values

Specifies a string of one or more instance startup name/value pairs. These values overwrite all values specified for the INSTVALS parameter in a previous invocation of the Start Web Option Server (YSTRW2ESVR) command and override the default values for the specified Web Option control values for the Web Option server instance.

Usage Details

Under normal circumstances, a Job Schedule Entry is added for the Web Option server job to ensure that it runs every day prior to anyone using the Web Option. However, if not, the Web Option server must be run on a daily basis at a time when no one is using the Web Option (for instance as part of a system startup job).

The server job can also be re-started at any point during the day, if changes have to be made to the Web Option product. See the details of the End Web Option Server (YENDW2ESVR) command for more information.

YUPGW2EENV – Upgrade Web Option Environment

The Upgrade Web Option environment (YUPGW2EENV) command is used to upgrade a Web Option environment data library to the latest release. Prior to running this command, you must first restore the latest versions of the Web Option product library (Y2WEB) and LDO library (Y2WEBVxxx) from media or from a saved file. If you have renamed these libraries during the restore process, you must ensure that the YW2ELDORFA data area in the renamed Web Option product library is changed to point to the renamed Web Option LDO library.

Note: This command can only be run by a user with both *ALLOBJ and *IOSYSCFG special authorities

Parameters

W2EENV (Web Option environment)

Specifies the name of the Web Option environment to upgrade.

W2ELIB (Web Option product library)

Specifies the latest version of the Web Option product library. This should be the name of the new Web Option product library restored from media.

***LIBL**

The first Web Option product library in the library list is used.

***W2EENV**

The Web Option product library currently defined for the Web Option environment specified in the Web Option environment prompt (W2EENV parameter) is used.

Library-name

Specifies a Web Option product library.

LDOLIB (Web Option LDO library)

Specifies the latest version of the Web Option LDO library. This should be the name of the new Web Option LDO library restored from media.

***W2ELIB**

The LDO library linked to the Web Option product library specified in the Web Option product library prompt (W2ELIB parameter) is used..

***W2EENV**

The Web Option LDO library currently defined for the Web Option environment specified in the Web Option environment prompt (W2EENV parameter) is used.

Library-name

Specifies a Web Option LDO library.

Usage Details

For full details on using the YUPGW2EENV command, see the section Install Web Option (see page 25).

YWRKELMCST - Work with Element Customization

Allows the editing of global element customization data accessed from the model to provide element-level customization for CA 2E screens.

Parameters

This command has no parameters

Usage details

See the appendix "Markup Language Skeleton Customization" for details on model-level customization

YWRKSCRXRF - Work with Screen Cross-References

Allows screen cross-reference records, which associate a screen with the corresponding HTML skeleton for that screen, to be created, updated, displayed or deleted. A screen is a single panel or view. Note that a single display can contain several screens (a 2E DSPRCD3 (Display record 3) function is an example of a display file that contains 4 separate screens - a key screen and 3 detail screens). Each separate screen within a display file must be identified.

Any CA 2E display files can include the *SCREEN ID field in their Header/Footer. Inclusion of this field results in automatic identification for each screen within that display file, and consequently, this command does not have to be run for those screens.

However, this command should be used whenever one of the following situations exists:

- A CA 2E-generated display file was generated prior to Release 8.0+ of CA 2E and you do not wish to regenerate it to include the *SCREEN ID field.
- You have manually created a skeleton for a non-CA 2E-generated screen. This might be an IBM screen, (for instance, the Work with All Spooled Files screen) that is part of a display file you have written yourself, or that you have received from a third-party vendor.

Manual identification is the process by which you determine a piece of output data on the screen (for example, the screen title) that uniquely identifies the screen. Using the YWRKSCRXRF command to manually identify a screen results in a record being created in the Screen Cross-Reference file, with its own unique Screen Identifier. It is this screen identifier that further links the screen to a skeleton. For more details about manual identification, see the Usage Details section that follows.

Parameters

Following are the parameters for this command:

Screen ID (SID)

Specifies the Web Option screen identifier for which you wish to edit the cross-reference data.

Model Library (MDLLIB)

Specifies the CA 2E model library for which you wish to edit cross-reference data. Cross-reference records for all functions created from that model are displayed.

DDS source member (DSPF)

Specifies the display file for which you wish to edit cross-reference data. Cross-reference records for all screens for that display file are displayed.

Selection type (SELTYP)

Specifies the selection type to use when selecting cross-reference data.

***POS**

The parameters passed in this command are used as a positioner to the initial Web Option selection screen. Using F5 (Refresh) or changing the subfile control positioners results in further records not related to the parameters passed in this command being displayed.

***RST**

The parameters passed in this command are used as a restrictor to the initial Web Option selection screen. F5 (Refresh) is not available and the subfile control fields are not displayed. No other records not related to the parameters passed in this command can be displayed or edited. Additionally, F6 (Create) is not available.

SEQ (Sequence number)

Specifies the sequence number of a specific skeleton cross-reference record to process.

***ALL**

The multiple-record screen is displayed showing all the records that match the specified criteria. Entering a specific sequence number along with a valid SID (Screen ID) results in the unique record displaying in a single-record screen. If no record matches the specific SID and SEQ parameters, an error message is returned.

sequence-number

If the SID and SEQ parameters are both specified, the unique record that they define will immediately be displayed in the single-record screen. If no record matches the specified SID and SEQ parameters, an error message is returned.

Usage Details

When using the YWRKSCRXRF command to manually identify a screen, the following guidelines apply:

- Choose a piece of text near the top of the screen to use as the identifying data. This gives the best performance when the Web Option merge processor is trying to identify the screen.
- Choose a piece of text that is unlikely to be duplicated on another screen. For example, the*PROGRAM field that appears in CA 2E screens at the top left corner is not a good choice as an identifying field because it contains the same value for different screens within a multiple-screen function (for example, an DSPRCD3).
- Ensure that the field you choose contains data that does not change between different invocations of the program.

For example, if you wanted to manually identify the following screen (a CA 2E -generated screen that does not include the *SCREEN ID field):

```
*PROGRAM   *PGMMOD  000000000000000000000000000000
                    Edit Region
Corp    Div                                     Region


 ?  Corp    Div     Description                   Region Description
                    0000000000000000000000000
                    0000000000000000000000000
                    0000000000000000000000000
                    0000000000000000000000000
                    000000000000000000000000
                    0000000000000000000000000
                    0000000000000000000000000
                    0000000000000000000000000

F3=Exit      F4=Prompt      F9=Go to 'Change' mode
```

If you want Edit Region to be the identifying text, and the skeleton to be called EDTREG1, then use F6 to create a Screen-cross reference record as shown in the following panel:

```
YSCRXRFE1R  CHANGE   Edit Screen Cross-reference


Screen ID: USR0001432   'Edit Region Panel 1'_________________________

Screen identifying data:  'Edit Region'______________________

SID row: 02   SID column:  36

DDS source member: RGHPEFKD
DDS source file  : QDDSSRC
                    *LIBL_____
Record format    : ZSFLCTL__


           ***** Skeleton details *****

Skeleton:   EDTREG1  Skeleton for Edit Region Panel 1_______________

Autoload level: 6   Server autoload level: 7


 F3=Exit  F12=Cancel
```

Note: You must enter the Record Format Name (there may be more than one screen associated with a function, and therefore more than one eventual Screen Identifier) as well as the DDS Source File and Member details.

The Screen Id is automatically assigned with the prefix USR, indicating that this panel is manually identified.

YWRKW2EVAL - Work with Web Option Control Values

Allows you to change any of the Web Option control values that are held in the Web Option system. These control values are held in the YW2EVALRFP file and control much of the behavior of the Web Option product. Details on each of these values and possible values are described in the appendix “Control Values”; they are divided into a number of different categories:

- Environment control values
- These values hold information that the Web Option uses to perform its basic functions, such as the name of the Web Option library, the default Web Option session timeout value, and others. Some of these values cannot be changed except when the Web Option environment is initialized prior to use.
- Formatting values

  The HTML generator uses these values to determine the formatting of the generated HTML skeleton. These values include default HTML table and background colors, sizes, and attributes. Some of these values are also used by the Web Option runtime to determine formatting of JIT HTML pages.

Parameters

The following are YWRKW2EVAL - Work with Web Option Control Values parameters:

VALTYP (Web Option control value type)

Type of Web Option control value to display (see W2EVAL):

***ALL**

Displays all Web Option control values

***ENV**

Displays only environment Web Option control values

***FMT**

Displays only formatting Web Option control values

W2EVAL (Web Option control value)

Name of the Web Option control value to use to position in the list of Web Option control values

Usage Details

Before changing any user-supplied Web Option control values, you should study the appendix “Control Values” to determine the result of changing the value.

Appendix B: Control Values

This appendix contains Control Values used by the Web Option and also a list of substitution variables that are defined by Web Option. All control values are held in the YW2EVALRFP file and can be displayed and edited by using the Work with Web Option Control Values (YWRKW2EVAL) command. The system-defined substitution variables can be specified in certain script statements and can be manually included in HTML skeletons.

Additionally:

- Most values can have the special value *NONE.

  For Control values used for formatting HTML, this normally causes the string that uses this value not to be output.

- Some of the Control Values used for formatting HTML source are color-values (for instance, the YDFTBGC (Default background color) value).

These values can be one of the following:

- One of 16 standard color names: black, silver, gray, white, maroon, red, purple, fuchsia, green, lime, olive, yellow, navy, blue, teal, or aqua.
- Any valid RGB (red-green-blue) color in the #RRGGBB format (for example, #00FF00)

This section contains the following topics:

YAUTDSC - Auto Disconnect at Sign-Off

YAUTDSC determines whether sessions should automatically disconnect from the IBM i when the user signs off from the IBM i.

***YES**

Sessions should automatically disconnect when you sign off

***NO**

Session should not automatically disconnect when you sign off

Shipped Value (Allowed Values)

*YES (*YES|*NO)

Usage

This control value controls the display of the sign-on (YSIGNON) page and the availability of virtual display devices as follows:

- If set to *YES

  Web Option users do not see the YSIGNON page when they sign off from the IBM i. Instead, the virtual display device they are attached to is automatically disconnected from the IBM i and the YENDED page displays. The virtual display device is available to other Web Option users.

- If set to *NO

  Web Option users see the YSIGNON page when they sign off from the IBM i. The device they are attached to is not automatically disconnected from the IBM i and the YSIGNON page displays. The virtual display device must be explicitly disconnected or it is not available to other Web Option users.

Note: If set to *NO, it is the responsibility of the developer to provide functionality on the YSIGNON page through which the user can explicitly disconnect from the virtual display device on the IBM i. This is done through the use of a button or similar HTML feature called W2E_CLOSE.

If a button is included on any Web Option skeleton with the name W2E_CLOSE as follows:

```
<input type=”submit” name=”W2E_CLOSE” value=”Disconnect”>
```

Tthe Web Option server disconnects from the virtual display device to which it is connected when that button is pressed, and the YENDED page displays to the client. If the button is pressed from any page other than the YSIGNON page, the user's interactive job ends abnormally and a job log is produced. If the button is pressed from the YSIGNON page, the user's interactive job ends normally.

YCMPRSS - Markup Language Skeleton Compression

This value determines how the generated HTML skeletons are loaded into memory prior to being merged with live data. If the value is set to *NO, each skeleton is loaded into memory in exactly the same format as exists in the YMLSSRC file, including any blank lines, indenting, and trailing blanks . If the value is set to *YES, each skeleton is compressed to remove unnecessary white space before being loaded into memory, resulting in better performance.

Shipped Value (Allowed Values)

*YES (*YES|*NO)

Usage

In most cases, YCMPRSS should be set to *YES. However, if a user-edited skeleton does not give the expected HTML results when displayed in the browser at runtime, this value can be set to *NO, resulting in more readable HTML source from the browser.

YCOLRTO - Column Ratio Multiplier Value

The YCOLRTO control value determines the multiplier used to convert a green screen column into the COLSPAN attribute of a *[assign the value for TD in your book]* HTML tag. By default, it is shipped as *2*, meaning that each column on a green screen is equivalent to a COLSPAN value of 2. Use any numeric value of 1 or greater.

Shipped Value (Allowed Values)

2	(1 - n)

Usage

The shipped value for the YCOLRTO Web Option control value of 2 is correct for most environments where the web pages are displayed on a PC-based web browser. However, if you are generating skeletons that display on a different device with a smaller screen, a smart phone, for example, you would change this value to get a smaller screen.

YDBFCCS - Database Coded Character Set

The YDBFCCS Web Option control value determines the value that should be used for the FsCcsid configuration when creating an HTTP server instance. It is also used when copying skeletons between the YMLSSRC source file and the IFS and when determining the code page of special characters in Web Option.

Shipped Value (Allowed Values)

37	(37 - 1123)

Usage

The shipped value for the YDBFCCS Web Option control value (37) is correct for most US English environments. If you are not running a US English environment on your IBM i, it should be changed to be the same as the QCHRID code page control value on your IBM i. When you initialize Web Option you are prompted to change it to the correct value.

Warning Note

The YDBFCCS Web Option control value is used by both Web Option and EJB Option. If you have both of these products installed, be aware that a change to it affects processing in both products.

YDBGDTA - Debugging Data Flags

YDBGDTA holds the flags that define the type of debugging data you want to retain. This value consists of thirteen *Yes* or *No* flags as follows:

- 1: CGI data from client
- 2: HTML data to client
- 3: VT input data
- 4: VT output data
- 5: VT open data
- 6: VT close data
- 7: Script statements
- 8: Dump errors
- 9: Client requests
- 10: Generator
- 11: Print definition
- 12: JIT errors
- 13:Element customization

Shipped Value (Allowed Values)

All flags set to *N*   (each flag can be Y or N)

Usage

The separate flags in the YDBGDTA control value allow you to specify the following debugging options:

1. CGI data from client

   If set to *Y*, CGI data from the client (browser) to the Web Option server is written to the debug data file.

2. HTML data to client

   If set to *Y*, HTML data sent from the Web Option server to the client (browser) is written to the debug data file.

3. VT input data

   If set to *Y*, 5250 data stream sent to the virtual terminal APIs from the Web Option server is written to the debug data file.

4. VT output data

   If set to *Y*, 5250 data stream sent from the virtual terminal APIs to the Web Option server is written to the debug data file

5. VT open data

   If set to *Y*, the data used to open the virtual terminal is written to the debug data file.

6. VT close data

   If set to *Y*, the data used to close the virtual terminal is written to the debug data file.

7. Script statements

   If set to *Y*, all script statements executed at runtime are written to the debug data file.

8. Dump errors

   If set to *Y*, all errors in the Web Option runtime result in the errant procedure being dumped to a QPPGMDMP spooled file.

9. Client requests

   If set to *Y*, client request records denoting different actions performed by the Web Option router and server are written to the debug data file.

10. Generator debugging

    If set to *Y*, debugging lines are included in the generated skeleton.

11. Print definition

    If set to *Y*, a spooled file is created when a skeleton is generated, describing the data that was retrieved from the display file

12. JIT errors

    If set to *Y*, and Just-In-Time errors are dumped to a QPPGMDMP spooled file

13. Element customization

    If set to *Y*, element customization records are written to the job log of the job running the YGENMLS command.

Warning! Only use debugging data when requested by CA support staff. All values must normally be set to *N*. Pay attention specifically to flags 1– 4; these can cause performance issues if set to *Y*. Flag 8 can cause major performance issues if errors are being encountered on an ongoing basis. Additionally, if set to *Y*, flag 1 results in data that is entered on the web page being written to a file, to include secure items such as passwords.

YBGSSN - Debug Session ID

YDBGSSN holds the 16-character session ID that should be debugged. You can set it to the special values **NONE* (meaning that no user jobs will be debugged) or **ALL* (meaning that all user jobs will be debugged), or you can set it to a valid 16-character session ID. The specific types of information to trace and debug are held in the YDBGDTA (see page 86) Web Option control value.

Note: The YDBGSSN control value is ignored unless at least one YDBGDTA flag is set to *Y*.

Shipped Value (Allowed Values)

*ALL	(*ALL|*NONE|session-identifier)

YDDLCND - Drop-Down List Conditions to Generate

YDDLCND determines which conditions for a CA 2E status field are generated as drop-down list (DDL) options.

- *ALL - all conditions for the field are generated
- *ACT - only those conditions specified in the action diagram (AD) are generated

Shipped Value (Allowed Values)

*ACT (*ACT|*ALL)

YDDLFMT - Drop-Down List Format (1, 2, 3, 4 or 5)

This value controls the formatting of generated drop-down lists from the CA 2E model status field. YDDLFMT takes values of 1, 2, 3 or 4 only. *NONE is not allowed. It controls the formatting of generated drop-down lists from a CA 2E model status field. For example, if a STS field condition has an internal value of '2' and an external value of 'Edit', the drop-down list entry for that value is generated as follows for each YDDLFMT value:

YDDLFMT value	Generated value
1	2=Edit
2	Edit (2)
3	Edit
4	2

Shipped Value (Allowed Values)

3 (1|2|3|4|5)

YDFTALC - Default HTML Active Link Color

YDFTALC is a color-value that defines the screen color of active links (that is, the color of a hyperlink). It can take the special value *NONE or one of the 16 special colors or a valid RBG color value.

Generated into the <BODY> tag of any JIT HTML windows and any pre-generated skeletons:

```
<BODY ...ALINK=ydftalc-value...>
```

Shipped Value (Allowed Values)

#CC33FF(color-value|*NONE)

Usage

Set YDFTALC to *NONE if no YDFTALC value is used. This results in the entire ALINK=”ydftalc-value” section not being generated within the BODY tag.

YDFTBGC - Default HTML Background Color (BGCOLOR)

This color-value is generated into the <HEAD> section of any JIT HTML windows and any pre-generated skeletons:

```
<LINK REL="STYLESHEET" TYPE="TEXT/CSS" HREF="ydftbgc-value.CSS”>
```

Shipped Value (Allowed Values)

WHITE	(color-value)

Usage

Unlike the other color-values, YDFTBGC cannot be *NONE. When a skeleton displays in the browser, the YDFTBGC value controls the color of the screen (and with it, the color of all the fields on that screen). It specifies the name of a Cascading Style Sheet (CSS) file which contains definitions for the background color, all field attributes, screen title attribute, command key attributes, and other style-related information. Two default CSS files are shipped with Web Option: WHITE.CSS and BLACK.CSS.

YDFTBKG - Default HTML Background (BACKGROUND)

YDFTBKG defines a default background image that displays when an HTML page is loaded. It takes the special value *NONE. If specified, the image should be a .GIF or .JPG file and should be held in the WEBOPT folder. This value, if specified, is generated into the <BODY> tag of any JIT HTML windows and any pre-generated skeletons:

```
<BODY ...BACKGROUND=ydftbkg-value...>
```

By default, most browsers tile an image to cover the entire screen.

Shipped Value (Allowed Values)

*NONE	(.gif or .jpg file name|*NONE)

Usage

Set YDFTBKG to *NONE if no YDFTBKG value is to be used. This results in the entire BACKGROUND=”ydftbkg-value” section not being generated within the BODY tag.

YDFTCSS - Default Cascading Style Sheet (CSS)

YDFTCSS specifies the name of the default Cascading Style Sheet (CSS) file to use for both generated and JIT pages. It can take the special value *NONE. If specified, the CSS file is held in the WEBOPT folder.

This value, if specified, is generated inside the <HEAD> tag of any JIT HTML windows and any pre-generated windows where it was defined within the CA 2E model.

Shipped Value (Allowed Values)

Y800600.CSS	(.css file name|*NONE)

Usage

If a value other than *NONE is specified, the following lines are generated into the HEAD section of a skeleton:

```
<LINK REL="STYLESHEET" TYPE="TEXT/CSS" HREF="/web2edoc/&&ydftbgc.css”>
```

A style sheet can be used to provide standard formatting across every page, giving a common look and feel. Although the use of Web Option values provides this in some ways, you may wish to use an external style sheet instead of or in addition to the common formatting provided by the Web Option system.

YDFTJSC - Default JavaScript File

This value specifies the name of the default JavaScript (.JS) file used for both generated and JIT pages it can take the special value *NONE. If specified, the JS file should be held in the WEBOPT folder. A JS file called YSCRIPT.JS is shipped with Web Option and contains a number of JavaScript functions to perform client-side validation of entered data.

Shipped Value (Allowed Values)

YSCRIPT.JS	(.js file name|*NONE)

Usage

If a value other than *NONE is specified, the following lines are generated into the HEAD section of a skeleton:

```
<script language="JavaScript"

src="http://WEB2EDOC/ydftjsc-value"></script>
```

The YSCRIPT.JS JavaScript file contains a number of functions that are used by the Web Option at runtime, to perform client-functionality, such as validating entered data before it is sent to the IBM i. You should not change or remove any of these functions or change the YDFTJSC value unless you replace them with equivalent functionality. You can edit the YSCRIPT.JS file to add functionality.

YDFTLKC - Default HTML Unvisited Link Color (LINK)

YDFTLKC is a color-value that defines the screen color of unvisited links (that is, the color of a hyperlink which has not yet been visited). It can take the special value *NONE or one of the 16 special colors or a valid RBG color value.

It is generated into the <BODY> tag of any JIT HTML windows and any pre-generated skeletons:

```
<BODY ...LINK=ydftlkc-value...>
```

Shipped Value (Allowed Values)

#0000FF (color-value|*NONE)

Usage

Set YDFTLKC to *NONE if no YDFTLKC value is to be used. This results in the entire LINK=”ydftlkc-value” section not being generated within the BODY tag.

YDFTSKL - Default Shipped Skeleton Names

The YDFTSKL control value holds the names of seven shipped skeletons at the following positions:

Position	Skeleton
1-10	Sign-on page (YSIGNON)

Position	Skeleton
11-20	General error page (YERROR)
21-30	Timeout error page (YTIMEOUT)
31-40	Authorization error page (YNOAUTH)
41-50	Just-In-Time error page (YJITERR)
51-60	Session ended page (YENDED)
61-70	Auto-submit page (YAUTSBM)

Shipped Value (Allowed Values)

'YSIGNON   YERROR   YTIMEOUT   YNOAUTH   YJITERR   YENDED   YAUTSBM'

Usage

The Web Option server uses this control value to determine the name of the skeleton to use when a system-defined event occurs (such as the displaying of the sign-on screen or ending a Web Option session). Under normal circumstances, you should not change this value.

YDFTSTO - Default Session Time Out Value (Seconds)

YDFTSTO defines the maximum amount of time (in seconds) that the Web Option Server waits for a response from a Web Option client job before ending the client job. It can take any numeric value between 180 (3 minutes) and 43200 (12 hours).

Shipped Value (Allowed Values)

3600   (180 - 43200)

Usage

This value is used as a security control by Web Option to ensure that Web Option client sessions cannot accidentally be left open, with the client's browser displaying a Web Option page on the IBM i. When a Web Option session times out, the job on IBM i is immediately ended. The IBM i HTTP server can also be configured to have a timeout value, and, if so, this value should not exceed the Web Option value.

YDFTTTL - Default HTML Page Title (TITLE)

YDFTTTL holds a text string (up to 80 characters) that is used as the HTML page title for JIT pages. It is generated into the HTML between the <TITLE> and </TITLE> HTML tags within the HEAD element:

```
<TITLE>ydftttl-value</TITLE>
```

Shipped Value (Allowed Values)

Unknown JIT page	(text-string)

Usage

*NONE is not allowed for this Web Option control value

YDFTTXC - Default HTML Text Color (TEXT)

YDFTTXC is a color-value that defines the default color of text on the screen (where that text is not specifically given a color). This color-value, if specified, is generated inside the <BODY> tag of any JIT HTML windows and any pre-generated skeletons:

```
<BODY ...TEXT=ydfttxc-value...>
```

Shipped Value (Allowed Values)

#000000	(color-value)

Usage

Set YDFTTXC to *NONE if no YDFTTXC value is to be used. This results in the entire TEXT="ydfttxc-value" section not being generated within the BODY tag.

YDFTTXS - Default Text Size <FONT SIZE=...>

YDFTTXS defines the default text size. It can take a numeric value from 1-5, where 1 is normal size text and 5 is extremely large text.

Shipped Value (Allowed Values)

3	(1|2|3|4|5)

YDFTUSR - Web Option Default User/Password

YDFTUSR defines a single default user ID that can be accessed through Web Option. It can take the special value *NONE. If specified, every user who accesses the IBM i through Web Option automatically sign on to the IBM i using this user ID. It should be specified as 'user-id   password ' (characters 1-10 = user-id, 11-20 = password) where 'user-id' and 'password' correspond to a valid user id/password combination on the IBM i.

Shipped Value (Allowed Values)

*NONE	(user-id/password combination)

Usage

See the appendix “Markup Language Skeleton Customization” for details on specifying a value for YDFTUSR.

YDFTVLC -Default HTML Visited Link Color (VLINK)

YDFTVLC defines the screen color of previously visited links (that is, the color of a hyperlink which has already been visited). It can take the special value *NONE or one of the 16 special colors or a valid RBG color value. It is generated into the <BODY> tag of any JIT HTML windows and any pre-generated skeletons:

```
<BODY ...VLINK=ydftvlc-value...>
```

Shipped Value (Allowed Values)

#FF0000 (color-value|*NONE)

Usage

Set YDFTVLC to *NONE if no YDFTVLC value is to be used. This results in the entire VLINK=”ydftvlc-value” section not being generated within the BODY tag.

YESCOPT - Screen String 'escape' Option

The YESCOPT control value determines if text that appears on green screens (either as a text constant or as a field value) is *escaped* for display in a web page. Escaped strings use special HTML tags to represent certain characters, for example , single quote ('), double quote ("), and the -than (<) characters. Valid values for the YESCOPT control value are currently **ALL* or **NONE*.

Shipped Value (Allowed Values)

*ALL (*ALL|*NONE)

YFKYACC - Function Key Accelerator Key Format

The YFKYACC control value determines the format (if any) for the accelerator character, which is used for command key buttons. The possible special values are:

- *NONE
- *UNDERLINE
- *ITALIC
- *BOLD.

The default value is *UNDERLINE. Alternatively, you can specify a user-defined value by entering a before-value and an after-value, separated by a comma. For example:

```
<span class="accKey">,</span>
```

Shipped Value (Allowed Values)

*NONE (*NONE|*UNDERLINE|*ITALIC|*BOLD|user-defined value)

Usage

The Web Option server uses this value when building command keys during skeleton generation time. This control value defines the look of the text on a command key button. For instance, specifying *UNDERLINE generates the following standard command keys so that they use the following default accelerator for each key: Exit, Prompt, Refresh, Toggle, Cancel, Enter, Help, Page Up, Page Down. User-defined command keys are determined using the first available initial letter that does not clash with an already-used letter from the previous list.

YFKYCNN - Function Key Connector

YFKYCNN can take the value of '=', '-' or ':' only. It defines the connector used in function key (F-key) text on existing green-screens and is used by the generator and the JIT processor to build F-key buttons in an HTML skeleton or on a JIT HTML page. It is assumed that all your applications use the same connector.

Shipped Value (Allowed Values)

= (=|-|:)

Usage

The Web Option server uses this value when creating JIT screens to identify command-key description strings. Submit buttons are automatically built from these descriptions. See the section YFKYTYP in this appendix for related details.

YFKYFMT - Function Key Format (1, 2, 3 or 4)

YFKYFMT can take values of 1, 2, 3 or 4 only. *NONE is not allowed. It controls the formatting of the text in generated submit buttons. For example, if the on-screen text for function key F3 is 'F3=Exit', the text displayed on the relevant submit button is generated as follows for each YFKYFMT value:

YFKYFMT value	Generated value
1	F3=Exit
2	Exit (F3)
3	Exit
4	F3

Shipped Value (Allowed Values)

(1|2|3|4)

YFKYIGN - Function Keys to Ignore during Generation

The YFKYIGN Web Option control value specifies command keys that exist for a screen which should be ignored during skeleton generation.

Shipped Value (Allowed Values)

Help (HL) | Print (PR) |Page Down (DN) |Page Up (UP) | Enter (EN) | Clear (CL)

Usage

The YFKYIGN Web Option control value defines any command keys which may exist in a display file which should be ignored when generating skeletons for that display file. It should be specified as a comma-separated list of command keys (including leading zero where appropriate). For example, “01, 04, HL”.

YFKYLST - Function Key LST Condition

The YFKYLST control value determines the surrogate number in the xxVLLSP file.

Type	**Value**
Value type	E (E=Environment, F=Formatting)
Data type	N (C=Character, N=Numeric)
Data value	1003204

Shipped Value (Allowed Values)

1003204

Usage

The YFKYLST control value determines the surrogate number in the xxVLLSP file of the '*ALL values' LST condition for the '*Web Option *CMD key' 2E model field. This list contains values for all the command keys which cannot be explicitly specified in a 2E action diagram and is used by Web Option if YLNGOPT(*MULTIPLE) is specified, to provide localized text for these command keys. If '*MDLDFT' is specified, a default value of 1003204 will be used.

YFKYTYP - Function Key Type (F, CF, CMD)

YFKYTYP can take a value of 'F', 'CF' or 'CMD' only. These define the format of the function key (F-key) type used on existing green-screens, by the generator and the JIT merge processor to build the various F-key buttons in an HTML skeleton or on a JIT HTML page (the F-key type is the 1-, 2- or 3-character string before the F-key number, for example, 'F3', 'CF04' or 'CMD12'). It is assumed that all your applications use the same F-key type.

Shipped Value (Allowed Values)

F	(F|CF|CMD)

Usage

The router uses this value when creating JIT screens to identify command key description strings. Submit buttons are automatically built from these descriptions.

YHTMOLF - HTML OnLoad Function

YHTMOLF defines the name of the 'onLoad' JavaScript function that should be run as soon as an HTML page is loaded. It can take the special value *NONE.

It is generated into the <BODY> tag of any JIT HTML windows and any pre-generated skeletons:

```
<BODY ...onload=”yhtmolf-value”>
```

Shipped Value (Allowed Values)

ONLOADFUN()	(JavaScript-function|*NONE)

Usage

YHTMOLF can take the special value *NONE or any function name up to 20 characters, including a leading blank and a pair of empty parentheses (for example, ONLOADFUN). The specified function must exist within the JavaScript file specified in the YDFTJSC value. An onLoad function is run by the browser when the page has finished loading and can contain simple functionality such as setting the cursor position.

YHTMOSF - HTML OnSubmit Function

YHTMOSF defines the name of the 'onSubmit' JavaScript function that is run whenever an HTML page is submitted, for example when a submit button has been pressed. The command can take the special value *NONE or any function name up to 20 characters including a pair of empty parentheses and a leading blank. The YHTMOSF command is generated into the <BODY> tag of any JIT HTML windows and any pre-generated skeletons:

```
<FORM ...onSubmit=”return yhtmosf-value”>
```

Shipped Value (Allowed Values)

ONSUBMITFUN (JavaScript-function|*NONE)

Usage

YHTMOSF can take the special value *NONE or any function name up to 20 characters including a pair of empty parentheses (for example, ONSUBMITFUN). If specified, it is generated into the <FORM> tag of both HTML skeletons and JIT HTML pages as <FORM... onsubmit="return yhtmosf-value">. If the function returns false, the page is not submitted. A Submit function can contain basic validation procedures to check that numeric values are entered in numeric fields and so on.

YLNGOPT - Language Option

The YLNGOPT Web Option control value specifies the type of environment for which Web Option skeletons should be generated.

Shipped Value (Allowed Values)

*SINGLE (*SINGLE|*MULTIPLE)

Usage

Specifying *SINGLE means that data generated into skeletons for drop-down lists is retrieved directly from the CA 2E model files YCNDDTARFP and YCONDTARFP, and data generated for command key text is taken from the screen. Specifying *MULTIPLE means that special replacement tags are generated into drop-down lists and command key text definitions; these tags are replaced when the skeleton displays at runtime with the relevant data from the CA 2E Values List file Y2VLLSP or a specified copy of that file. For more details, see the section on changing the ValuesListFile and ValuesListLib values in the appendix “Scripting”.

YMLSFLR - Folder for Markup Language Skeletons

YMLSFLR defines the name of the folder within the '/' (root) folder where HTML skeletons are placed by the YPRCSKL command allowing them to be edited. YMLSFLR is also the folder where any static PC objects such as JavaScript files, CSS files, background images etc. are held. *NONE is not allowed for the YMLSFLR value.

Shipped Value (Allowed Values)

/WEBOPT/	(directory-name)

Usage

YMLSFLR is shipped as /WEBOPT/ and unless this folder is used by other processes, should not be changed.

YNBRSVR - Number of Server Jobs

YNBRSVR controls the number of Web Option server jobs that are started when the YSTRW2ESVR (Start Web Option Server) job is executed. When the YSTRW2ESVR command is executed, it automatically starts one *audit* job and one or more *server* jobs (controlled by the YNBRSVR value). The minimum number of server jobs that can be started is 1 and the maximum number is 500. The default is 4. Web option server and audit jobs run using the WEB2E user profile and the WEB2E_SVR job description and have the same name as the Web Option library.

Shipped Value (Allowed Values)

4	(1–500)

Usage

At runtime, as clients access the IBM i using the Web Option, the Web Option auditor job 'registers' their job with one of the server jobs it has started (based on the current load being experienced by each server job) and 'deregisters' the client job when it disconnects from the IBM i. In general, the higher the YNBRSVR value, the better the performance. However, a higher number here results in more server jobs, each of which can use large amounts of memory and processing power. If necessary, additional Web Option server jobs are started to cater for client demand.

YNETCCS - Net Coded Character Set

The YNETCCS Web Option control value

YPREFIX - HTML Skeleton Prefix Character

YPREFIX holds a single character that is concatenated with a unique 7-digit number to create the name of a Markup Language Skeleton. Each skeleton is automatically assigned a number when it is first generated. Thus if YPREFIX has a value of 'H', then skeletons are automatically named 'H0001025', 'H0001026' and so on. YPREFIX can take any character from A to X (Y and Z are reserved for internal use).

Shipped Value (Allowed Values)

H	(A|B|C|D|E|F|G|H|I|J|K|L|M|N|O|P|Q|R|S|T|U|V|W|X)

YRTRMWT - Router Maximum Wait Time

The YRTRMWT Web Option control value defines the maximum amount of time (in seconds) that the Web Option router should wait for a response from the Web Option server, following a client request. It can take values ranging from -1 (forever) to 300 (5 minutes) The default shipped value is 120 (2 minutes).

Shipped Value (Allowed Values)

120	(-1 - 300)

Usage

When a Web Option client session submits a page, the Web Option router sends a request to the Web Option server and then waits for a response. Several different factors can affect the length of time it takes for a request/response to be processed, including the number of active Web Option client sessions, the number of Web Option servers running and also the processing power of the IBM i. A value for YRTRMWT that is too low can be exceeded if many clients are accessing few servers, resulting in clients being disconnected. However, if the value is set at -1 (forever), then if a communications error occurs or if the Web Option server crashes, the HTTP server job in which the router is running remains at DEQW status waiting for a response from the server, which can affect performance across the IBM i. It is suggested that the YRTRMWT value not be changed unless you notice that client sessions are receiving communication error messages. Note that prior to Release 8.1 if Web Option, the router used an equivalent value of -1 (forever).

YSCPERR - Script Error Page

YSCPERR holds the name of the error page that displays if an error is encountered while processing a script and the YSCPRCV Web Option control value is set to '*ERRPAGE'. If the specified page cannot be found (or if this value is set to '*NONE', the default YERROR error page is used.

Shipped Value (Allowed Values)

YSCPERR(error-page-name|*NONE)

YSCPRCV - Script Recovery Action

YSCPRCV defines how an error within a script process is handled. If it is set to '*RECOVER', the Web Option server attempts to recover from the error automatically. If it is set to '*ERRPAGE', the error page defined in the YSCPERR Web Option control value displays. If it is set to '*ENDJOB', the users job is ended and the YERROR page displays.

Shipped Value (Allowed Values)

*ERRPAGE	(*ERRPAGE|*RECOVER|*ENDJOB)

YSFSLST - Subfile Select (*SFLSEL) LST Condition

The YSFSLST control value determines the surrogate number in the xxVLLSP file.

Type	Value
Value type	E (E=Environment, F=Formatting)
Data type	N (C=Character, N=Numeric)
Data value	1003217

Shipped Value (Allowed Values)

1003217

Usage

The YSFSLST control value determines the surrogate number in the xxVLLSP file of the '*ALL values' LST condition for the '*Web Option *SFLSEL' 2E model field. This list contains values for all the default subfile select options cannot be explicitly specified in a 2E action diagram and is used by Web Option if YLNGOPT(*MULTIPLE) is specified, to provide localized text for these options. If '*MDLDFT' is specified, a default value of 1003217 will be used.

YSKLCHK - Check Skeleton Date Frequency

The YSKLCHK control value determines the frequency that Web Option checks for skeleton updates after they are initially cached by the Web Option server. It can take any numeric value greater than or equal to 0 (zero). A value of 0 means that skeletons, once cached, are never checked for updates. A value of 1 means that skeletons are checked for updates every time they are accessed. Any other positive value N means that each skeleton is checked for updates every Nth time it is accessed.

Shipped Value

1	(0 or greater)

Usage

At runtime, the Web Option server checks if a skeleton has been changed since it was last displayed. This ensures that the latest version of a skeleton is always used, but adds overhead because every skeleton is checked each time it is displayed. In a production environment, where skeletons are not updated once installed, set the control value to 0, which provides the greatest performance benefit. In a development environment where you or others are making regular changes to skeletons, set the control value to any positive value.

Backwards-compatibility Note: The default value (1) gives the worst performance, but is used for backwards-compatibility purposes. As noted above, in a production environment, it should be changed to zero (0).

YSVRLMT - Server Load Limit

The YSVRLMT Web Option control value specifies the maximum number of client sessions that can be attached to a single Web Option server job. It can take any numeric value from 1 to 2147483647.

Shipped Value (Allowed Values)

(1 - 2147483647)

Usage

The YSVRLMT Web Option control value specifies how many client sessions each server job can handle. It is used in conjunction with the YNBRSVR Web Option control value to specify a limit to the number of client sessions that can be active at any time. For instance, if 4 Web Option server jobs were started initially and YSVRLMT is set to 20, then a maximum of 80 client sessions can be active at once. Any additional client sessions that try to connect to Web Option receive an error message stating that they cannot do so.

YSVRMWT - Server Maximum Wait Time

The YSVRMWT Web Option control value specifies the maximum amount of time (in seconds) that the Web Option server should wait for a response from i OS once it has sent a formatted client request to i OS.

Shipped Value (Allowed Values)

(10 - 180)

Usage

When the Web Option server processes a client request, it sends formatted data to i OS, which then passes that data to the application program currently running within the client job, as if the user had entered data on the green-screen. The application program than processes the data and displays a new screen, which is retrieved by i OS and passed back to the Web Option server. Typically, this process takes less than one second (since it is effectively the length of time taken for one green-screen to transition to the next, following a key-press by the user). The shipped value for YSVRMWT (30 seconds) should therefore obviously be sufficient for almost all transactions. However, if the application program performs some complex processing such as running a query interactively, or if the IBM i is very slow, then this value can be exceeded. If the YSVRMWT value is exceeded, the client job is automatically ended.

YSYSTEM - System IBM i Name

YSYSTEM holds the name of the IBM i on which CA 2E is being run. By default it takes the special value '*SYSNAME' which evaluates at runtime to the IBM i system name (SYSNAME). However, YSYSTEM can also take either the URL or TCP/IP address of the IBM i, (for example www.company-name.as400.com or 123.123.123.123).

Shipped Value (Allowed Values)

*SYSNAME	(system-name/TCP/IP number[:port-number])

Usage

If the Web Option is running on an IBM i called S1013RFT, then the following situations can exist:

- S1013RFT is only accessed from within a network using Web Option port 4100:
- Set YSYSTEM to S1013RFT:4100
- S1013RFT is accessed from the internet using the URL www.mycompany.as400.production.com using Web Option port 4100:

  Set YSYSTEM to www.mycompany.as400.production.com: 4100

YTBLBDR - Default Table Border Width <TABLE BORDER=>

YTBLBDR defines the default width (in pixels) of the border around tables in generated skeletons.

It is generated into <TABLE> tags within any pre-generated skeletons:

```
<TABLE ...BORDER=”ytblbdr-value”>
```

Shipped Value (Allowed Values)

*NONE	(0 - 100|*NONE)

Usage

Any value from 0-100. A value of *NONE is the same as a value of 0 and results in no border being displayed.

YTBLBGC - Default Table Background Color

YTBLBGC is a color-value that defines the background color of every table in a generated HTML skeleton.

It is generated into <TABLE> tags within any pre-generated skeletons:

```
<TABLE ...BGCOLOR=”ytblbgc-value”>
```

Shipped Value (Allowed Values)

#FFFFFF (color-value|*NONE)

YTBLCPD - Default Table Cell Padding

YTBLCPD defines the value (in pixels) used for TABLE cell padding in a generated HTML skeleton. Cell padding refers to the margin inside a data cell, between the edge of the cell and the text or image inside the cell. It is generated into <TABLE> tags within any pre-generated skeletons:

```
<TABLE ...CELLPADDING=”ytblcpd-value”>
```

Shipped Value (Allowed Values)

1	(0 - 100|*NONE)

Usage

YTBLCPD can take a value from 0-100 or the special value *NONE (which equates to a value of 0).

YTBLCSP - Default Table Cell Spacing

YTBLCSP defines the value (in pixels) used for TABLE cell spacing in a generated HTML skeleton. Cell spacing refers to the space around and between data cells (between the inside edge of the table border and the edge of the cell). It is generated into <TABLE> tags within any pre-generated skeletons:

```
<TABLE ...CELLSPACING=”ytblcsp-value”>
```

Shipped Value (Allowed Values)

2	(0 - 100|*NONE)

Usage

YTBLCSP can take a value from 0-100 or the special value *NONE (which equates to a value of 0).

YTBLHGT - Table Height (% of Screen)

YTBLHGT determines the height of a TABLE on an HTML page, as a percentage of the page size. It can take a value from 0-100 or the special value *NONE (which equates to a value of 0). It is generated into <TABLE> tags within any pre-generated skeletons:

```
<TABLE ...HEIGHT=”ytblhgt-value”>
```

Shipped Value (Allowed Values)

*NONE	(0 - 100|*NONE)

YTBLWTH - Table Width (% of Screen)

YTBLWTH determines the width of a TABLE on an HTML page, as a percentage of the page size. It can take a value from 0-100 or the special value *NONE (which equates to a value of 0). It is generated into <TABLE> tags within any pre-generated skeletons:

```
<TABLE ...WIDTH=”ytblwth-value”>
```

Shipped Value (Allowed Values)

100	(0 - 100|*NONE)

YTDCALG - Table Data Cell ALIGN (default = LEFT)

YTDCALG determines the alignment of text and images within a table data cell. It is generated into [assign the value for TD in your book] tags within any pre-generated skeletons:

```
<TD ...ALIGN=”ytdcalg-value”>
```

Shipped Value (Allowed Values)

*NONE	(LEFT|CENTER|RIGHT|JUSTIFY|*NONE)

Usage

If *NONE is specified for YTDCALG, the ALIGN attribute is not generated for the [assign the value for TD in your book] HTML tag and browsers assume the default value of LEFT.

YTDCBGC - Default Table Data Cell Color <TD BGCOLOR=>

YTDCBGC is a color-value that defines the background color of every table data cell in a generated HTML skeleton.

It is generated into [assign the value for TD in your book] tags within any pre-generated skeletons:

```
<TD ...BGCOLOR=”ytdcbgc-value”>
```

Shipped Value (Allowed Values)

#FFFFFF (color-value|*NONE)

YTDCHGT - Table Data Cell Height (Pixels)

YTDCHGT determines the height of a table data cell on an HTML page, as a percentage of the page size. It can take a value from 0 to 100 or the special value *NONE, which results in the browser automatically formatting the table data cell height based on the size of the data within the table data cells that make up the table. It is generated into [assign the value for TD in your book] tags within any pre-generated skeletons:

```
<TD ...HEIGHT=”ytdchgt-value”>
```

Shipped Value (Allowed Values)

20	(0 - 100|*NONE)

YTDCVAL - Table Data Cell VALIGN (default = MIDDLE)

YTDCVAL determines the vertical alignment of text and images within a table data cell. It is generated into [assign the value for TD in your book] tags within any pre-generated skeletons:

```
<TD ...VALIGN=”ytdcval-value”>
```

Shipped Value (Allowed Values)

*NONE	(TOP|MIDDLE|BOTTOM|BASELINE|*NONE)

Usage

If *NONE is specified, the VALIGN attribute is not generated for the [assign the value for TD in your book] HTML tag, and browsers assumes the default value of MIDDLE.

YTRWALG - Table Row ALIGN (default = LEFT)

YTRWALG determines the alignment of table data cells within a table row. It is generated into <TR> tags within any pre-generated skeletons:

```
<TR ...ALIGN=”ytrwalg-value”>
```

Shipped Value (Allowed Values)

*NONE	(LEFT|CENTER|RIGHT|JUSTIFY|*NONE)

Usage

If *NONE is specified, the ALIGN attribute is not generated for the [assign the value for TD in your book] HTML tag, and browsers assumes the default value of LEFT.

YTRWBGC - Default Table Row Color <TR BGCOLOR=...>

YTRWBGC is a color-value that defines the background color of every table row in a generated HTML skeleton. It is generated into <TR> tags within any pre-generated skeletons:

```
<TR ...BGCOLOR=”ytrwbgc-value”>
```

Shipped Value (Allowed Values)

#FFFFFF (color-value|*NONE)

YTRWVAL - Table Row VALIGN (default = MIDDLE)

YTRWVAL determines the vertical alignment of table data cells within a table row. It is generated into <TR> tags within any pre-generated skeletons:

```
<TR ...VALIGN=”ytrwval-value”>
```

Shipped Value (Allowed Values)

*NONE	(TOP|MIDDLE|BOTTOM|BASELINE|*NONE)

Usage

If *NONE is specified, the VALIGN attribute is not generated for the [assign the value for TD in your book] HTML tag, and browsers assumes the default value of MIDDLE.

YURLTYP - URL Type

YURLTYP defines the type of URL to be used in the <FORM...> tag within generated skeletons and JIT pages (replacing the '(_xCALLURL)' Web Option tag) and JIT pages. If set to *ABSOLUTE, the full URL used to initially access Web Option is used. If set to *RELATIVE, the protocol and IBM i name are not included (for example, '/WEB2E' is used).

Shipped Value (Allowed Values)

*ABSOLUTE	(*ABSOLUTE|*RELATIVE)

Usage

Under most circumstances, *ABSOLUTE and *RELATIVE works equally well. However, if your IBM i is behind a firewall or is being accessed through a proxy server, such that neither the IBM i name nor the IBM i IP address is publicly available, the relative URL format is used.

YVARTYP - Substitution Variable Type

The YVARTYP Web Option control value specifies the type of substitution variables that are allowed within Element Customization

Shipped Value (Allowed Values)

*ALL	(*ALL|*SHORT|*LONG)

Usage

Within Element Customization, there can by default be two different types of substitution variables. Short variables (which were originally available in Release 7.0+) are all 2-character variables, such as &v and &f. Long variables (available since Release 8.0) are 9-character variables such as &&ydftbgc and &&_fldval. The YVARTYP Web Option control value specifies which of these substitution variable types are processed within Element Customization when a skeleton is generated. If *SHORT is specified, long variables such as &&_fldval are not replaced with their relevant value and appear in the generated skeleton. If *LONG is specified, short variables such as &v are not replaced with their relevant value and appear in the generated skeleton. If *ALL is specified, both types of substitution variables are replaced with their relevant values in the generated skeleton.

Note: If the Web Option control value YVARTYP is set to *ALL or *SHORT, then all short substitution variables within element customization must be specified without an immediately following alpha character (a-z orA-Z). Any other character (including a blank, single quote or double quote) is acceptable.

If it is required to have, for example, "...value=&vxyz..." and the '&v' should be converted to the relevant value, then the YVARTYP control value should be set to *ALL and the &v should be replaced with its equivalent long-form substitution variable as "...value=&&_fldvalxyz...".

YVRTTRM - Virtual Terminal Type

The YVRTTRM control value determines the type of virtual terminal that should be created for Web Option users. Values are:

- **DS3* – 24 x 80 SBCS display)
- **DS4* – 27 x 132 SBCS display
- **DBC* – 24 x 80 DBCS display

Shipped Value (Allowed Values)

*DS3 (*DS3|*DS4)

Web Option Substitution Variables

A number of substitution variables are used within Web Option to provide generic processing. Two main types of substitution variables are available:

- System-defined substitution variables
- User-defined substitution variables

System-defined substitution variables come in two different subtypes: generation-time substitution variables and run-time substitution variables. Both subtypes consist of a specific set of variables which are defined below.

User-defined substitution variables have a specific name-format, but can have any value which does not clash with any system-defined substitution variable.

System-Defined (Generation-time) Substitution Variables

System-defined (generation-time) substitution variables are specified within records in the MLS Syntax File YMLSSYNRFP and when the skeleton is being generated, they are replaced with skeleton-specific values. Generation-time substitution variables are always 7-characters in length and begin with an underscore (_). Generation-time substitution variables can also be included in element customization, but should not be directly included by manually adding them to a generated skeleton.

_w2elib	Web Option product library
_w2evsn	Web Option version
_ckynam	Command key name
_ckyval	Command key value
_intval	Internal value (Y2VLLSP external value)
_extval	External value (Y2VLLSP condition text)
_filnam	Display file name
_fillib	Display file library
_filtxt	Display file text description
_mdllib	CA 2E model library
_gentim	Time skeleton was generated
_gendat	Date skeleton was generated
_genusr	User who generated skeleton
_fmtnam	Record format name

_fmttyp	Record format type
_fmttxt	Record format text description
_scrttl	Screen title
_sfltx1	Subfile text line 1
_sfltx2	Subfile text line 2
_sflpmt	Subfile prompt text
_ckystr	Command key string
_cfmtxt	Confirm prompt text
_cmdfld	Confirm prompt field name
_cfmvly	Confirm prompt Yes value
_cfmvln	Confirm prompt No value
_cfmoff	Confirm prompt field offset
_fldatr	Field attribute
_fldrow	Field row
_fldcol	Field column
_fldoff	Field offset
_fldtxtx	Field text description
_fldalg	Field alignment (left|right)
_colspn	Element COLSPAN value
_fldval	Field value
_scrval	Screen value
_fldnam	Field DDS name for example, Z1ABTX
_mlsnam	Field (MLS) name for example, _F1234U
_fldlen	Field length
_objgrp	Object group being processed
_fldtag	Field tag will be replaced at runtime with the correct data for the field

System-Defined (Run-time) Substitution Variables

System-defined (run-time) substitution variables are directly generated into skeletons within System Variable tags and are replaced at runtime with session-specific values. There are currently 14 run-time substitution variables. These substitution variables must not be modified within a generated skeleton and are included here for information purposes only.

CALLURL

The CALLURL substitution variable is replaced at runtime with the URL (Uniform Resource Locator) used to access Web Option, for example, http://myas400:4100/web2e. It is automatically included in the <form> HTML tag in all generated skeletons:

```
<form...action=”(_xCALLURL)”...>
```

SKEL

The SKEL substitution variable is replaced at runtime with the name of the Web Option skeleton being displayed. The skeleton name is automatically included in the <form> HTML tag in all generated skeletons and also within the confirm prompt:

```
<form...name=”(_xSKEL)”...>
```

SSNID

The SSNID substitution variable is replaced at runtime with the unique 128-character Web Option client session identifier. The session identifier is automatically included in all generated skeletons as a hidden field:

```
<input type="hidden" name="W2E_SSNID" value="(_xSSNID)">
```

SELFIELD

The SELFIELD substitution variable is replaced at runtime with the name of the field which is currently 'selected' - that is, the field in the HTML page that should receive focus when the page displays in the browser. The selected field is automatically included in all generated skeletons as a hidden field:

```
<input type="hidden" name="W2E_SELECT" value="(_xSELFIELD)">
```

SCRMSG

The SCRMSG substitution variable is replaced at runtime with the message being displayed in the message line on the underlying green-screen. The SCRMSG substitution variable is automatically included in all generated skeletons:

```
(_xSCRMSG)
```

ERRPGM/ERRPRC/ERRMOD/ERRSTS/ERRSQN/ERRRVN/ERRJOB/ERRMSG/ERRMSD

The ERR* substitution variables are only valid in the YERROR shipped skeleton and are replaced at runtime with the error details which caused the YERROR page to be displayed.

User-Defined Substitution Variables

User-defined substitution variables have the same format as system-defined (generation-time) substitution variables - they must be 7 characters long and must begin with an underscore (_).

User-defined substitution variables can be defined within scripts and then retrieved and included in a run-time HTML page. To include a user-defined substitution variable on an HTML page, the substitution variable should be incorporated into a System Variable tag. For instance, a user-defined substitution variable called _myvar1 would be included in a page as follows:

```
...(_x_myvar1)...
```

The example script below can be used on a page which has multiple command keys and uses the F24 key to display further command keys. The script (which could be specified as a screen-load script) retrieves the text on lines 22 and 23, presses F24, retrieves the new text on lines 22 and 23 and the finally presses F24 again to redisplay the first screen.

```
get r22c01l80 $row221
get r23c01l80 $row231
put &&_row221 $row221
put &&_row231 $row231
send 24
get r22c01l80 $row222
get r23c01l80 $row232
put &&_row222 $row222
put &&_row232 $row232
send 24
```

The skeleton to which this script is attached as a screen-load script could include the following JavaScript to pass all the command key text to a JavaScript function which could format the text into a number of command key buttons:

```
<script>
cmdkeytxt = '';
if ("(_x_row221)" != "") {
  cmdkeytxt += "(_x_row221)"
}
if ("(_x_row231)" != "") {
  cmdkeytxt += " " + "(_x_row231)"
}
if ("(_x_row222)" != "") {
  cmdkeytxt += " " + "(_x_row222)"
}
if ("(_x_row232)" != "") {
  cmdkeytxt += " " + "(_x_row232)"
}
setCommandKeys(cmdkeytxt);
</script>
```

This processing would allow screens which have dynamic command key text to display correctly, since a generated skeleton for these screens would generate no command keys.

Appendix C: Markup Language Tags

When a Markup Language Skeleton (MLS) is generated, the resulting source member contains a combination of HTML tags and Web Option tags. The HTML tags provide much of the presentation elements of the final HTML page, the general layout and 'look-and-feel' of the HTML page, whereas the Web Option tags define how live data is merged into the MLS at runtime. The skeleton may also contain special Web Option control comments that provide extra information to the Web Option server about how to process the skeleton.

At runtime, the server takes a copy of the skeleton and scans the copy looking for tags. For each tag, the server performs certain processes, such as replacing tags with specified live data, thus producing a full HTML page containing live data. It is this HTML page, and not the skeleton, that is sent to the browser for display.

There are several different types of HTML tags, each of which follows the same general format but with its own specific attributes and own specific sub-format. This appendix covers the different formats of the various tags and discusses how they are used in Markup Language Skeletons.

Additionally, input HTML elements (<INPUT...> and <SELECT...> elements) in the generated skeleton use a standard naming system allowing the Web Option to process them correctly.

This section contains the following topics:

Web Option Control Comments

In each Web Option skeleton, one or more special HTML comments known as Web Option control comments may automatically be generated or manually added. Web Option control comments can be recognized as having the following format:

```
<!--%W2E% comment-string %W2E%-->
```

where comment-string contains control information used by the Web Option server job to perform skeleton-specific processing prior to the data-merge processing.

In all generated skeletons, the following Web Option comment line is generated automatically into the top of each skeleton as part of the *HTML MLS Syntax object group:

```
<!--%W2E% -vallstfil Y2VLLSP -pmtmsgf UUPMTMSG %W2E%-->
```

At runtime, this string is processed by the Web Option server job and the specified Values List File and Prompt Message File are loaded into memory. If the skeleton uses any replacement tags (following), these values are used by the Web Option server.

There is also a client-level value called PMTMSGFLIB which will hold the current prompt message file library name that the client should use. Therefore, the values list file variables VALLSTFIL and VALLSTLIB are totally separate from the prompt message file variables PMTMSGF and PMTMSGFLIB. The PMTMSGFLIB value can be set as follows:

1. Within a Web Option skeleton by including the following code within the special Web Option comment line (beginning "<!--%W2E%" and ending "%W2E%-->): <!--%W2E%...-PMTMSGFLIB library-name ...%W2E%-->

   For example:

   ```
   <!--%W2E% -pmtmsgf UUPMTMSG -pmtmsgflib RUNLIB %W2E%-->
   ```

   This forces the Web Option server to retrieve messages from message file UUPMTMSG in library RUNLIB when it encounters a prompt-message reference tag such as (_rMAA1B2F). If not specified (or *LIBL is specified), the Web Option server will calculate the actual library name to use (i.e. the highest library in the clients library list which contains the specified prompt message file) and will use that library until the PmtMsgfLib value is again set.

   Note that, by default, Web Option comment lines will continue to include only references to the VALLSTFIL and PMTMSGF variables - the VALLSTLIB and PMTMSGFLIB variables must either be added manually post-generation or the relevant MLS Syntax object group should be changed to include these variables. When not specified, *LIBL is assumed for both these variables.

2. Within a script PUT statement which uses the following syntax:

   ```
   PUT library-name YPMTLIB
   ```

   If a referenced tag (one beginning "(_r") is found within a skeleton and a Web Option comment line is NOT found and a load-script has not been specified, the tag processor will set the PMTMSGFLIB variable to *LIBL and pass that to the prompt message processing procedure, which will update PMTMSGFLIB to the correct library name.

Allowable switch/value pairs in a Web Option comment are as follows:

-waittime nnn

Specifying this switch ensures that the wait time used when the page is submitted will be the maximum of the specified value (nnn) and the current YSVRMWT value.

-timeout nnn

Specifying this switch ensures that the timeout value used when the page is submitted is the maximum of the specified value (nnn) and the currentYDFTSTO value.

-debugsession

Specifying this switch ensures that the Web Option server job processing this client session will have its internal YDBGSSN value set to the session identifier for the client session.

-nodebugsession

Specifying this switch ensures that the Web Option server job processing this client session will have its internal YDBGSSN value set to the current YDBGSSN Web Option control value.

-autodisconnect

Specifying this switch will ensure that the Web Option client session will have its auto-disconnect value set on (irrespective of the current YAUTDSC Web Option control value).

-noautodisconnect

Specifying this switch ensures that the Web Option client session will have its auto-disconnect value set off (irrespective of the current YAUTDSC Web Option control value).

-jitonly

Specifying this switch ensures that the Web Option client session will only use Just-In-Time screens for the remainder of its connection.

-vallstlib lib-name

Specifying this switch will ensure that the Web Option client session will have its internal Values List Library set to the specified value (lib-name).

-vallstfil file-name

Specifying this switch will ensure that the Web Option client session will have its internal Values List File set to the specified value (file-name). If the file is not already loaded by the Web Option server job processing the client session, the file will be loaded (either from the library specified for the -vallstlib switch or from the client jobs library list if the -vallstlib switch has not been specified).

-pmtmsgf msgf-name

Specifying this switch will ensure that the Web Option client session will have its internal Prompt Message File set to the specified value (msgf-name). If the message file is not already loaded by the Web Option server job processing the client session, it will be loaded (either from the library specified for the -vallstlib switch or from the client job's library list if the -vallstlib switch has not been specified).

pmtmsgflib lib-name

Specifying the current prompt message file library name that the client should use for finding the prompt message file.

The MLS Generator generates the following Web Option comment line into each skeleton:

```
<!--%W2E% -vallstfil &&_lstfil -pmtmsgf &&_pmtmsf %W2E%-->
```

where the 'Y2VLLSP' and 'UUPMTMSG' values are retrieved from the model definition. This comment line is defined in the *HTML object group in the MLS Syntax file YMLSSYNRFP.

Web Option Tag Format

The format of the Web Option tags is significantly different from HTML tags. There are a number of different Web Option tag types - the following example shows two possible Web Option tags.

(_v0835) or (_tSALESFIG)

Each tag follows the same basic format:

- An opening bracket and an underline ((_), which identifies the tag to the Web Option server
- An identifying letter (such as v or t in the example) defines the type of processing that the Web Option server performs for the tag (see the Web Option Tag Identifying Letters section in this appendix for more details)

  Tag specific data used by the Web Option server (such as 0835 or SALESFIG in the example) when processing the tag

  A closing bracket ()) used to identify the end of the tag

Web Option Tag Identifying Letters

The following table lists the identifying letters currently used with the Web Option.

Identifying Letter	Used For
d	Dependency tags
f	Formatted field tags
r	Replacement language tags
q	Exit program tags
t	User-defined text tags
v	Variable tags
x	System variable tags

Details about these tag types, how they are used, and the format of the data that is included in them is detailed in the following sections.

Note: Fields passed in the data stream is by the offset of that field on the screen being processed. Valid values for data offsets are between 2 and 1920 (for 24 by 80 screens) and between 2 and 3564 (for 27 by 132 screens) and are always generated as 4-character strings with leading zeros where applicable. Thus 1047 and 0043 are valid offsets, whereas 2430 and 84 are invalid (value outside range (for a 24 by 80 screen) and no leading zeros respectively). The standard algorithm for converting between screen offset and screen row/column is:

```
OFFSET = ((ROW - 1) *  maximum-columns) + COLUMN
```

Where maximum columns is either 80 or 132.

The YCVTRCO command can be used to convert between row or column and offset, and vice-versa.

Note: Within this document, where an offset value is defined within a tag description, it is represented as *NNNN*.

Dependency Tags

Usage

Dependency tags are used to provide data-specific HTML generation. When the server encounters a start-dependency tag, it processes the data up to the associated end-dependency tag. Within the tags, the server determines whether or not to include the data within the HTML page, which is sent to the browser. For example, drop-down lists for sub-file selections and the standard CA 2E confirm prompts are generated in a MLS to include dependency tags, since they are only displayed under certain circumstances.

Format

Dependency tags can take one of three start formats, an otherwise format, and also a single end format:

- Output-when-match:
- (_dNNNN+xxxxxxxxxx)

  If comparison string **xxxxxxxxxx** is the same as the screen data at screen offset NNNN, all the data between the start and end dependency tags is written to the browser (including any other Web Option tags, which is processed as normal). The lines containing the start and end tags themselves are not written.

- Skip-when-match

  `(_dNNNN-xxxxxxxxxx)`

  If comparison string **xxxxxxxxxx** is the same as the screen data at screen offset NNNN, none of the data between the start and end dependency tags are written to the browser. The lines containing the start and end tags themselves are not written.

- Output-when-field:

  `(_dNNNNI)`

  If an input field exists on the screen at offset NNNN (whether or not it contains data), all the data between the start and end dependency tags are written to the browser. The lines containing the start and end tags themselves are not written.

- Otherwise tag:

  `(_dNNNNX)`

  If a start tag of any type has been encountered with screen offset NNNN specified and that tag was not fulfilled, then the lines between this tag and the end-tag are output.

- End tag:

  `(_dNNNN)`

The end-tag ends a section of dependent skeleton.

Example

Below is an example of the CONFIRM prompt HTML that is generated into the HTML skeleton:

```
<!--(_d1904+CONFIRM)  Confirm prompt (dependent)-->

 <INPUT TYPE="HIDDEN" NAME="_F1913U" VALUE=" " SIZE="1">

 <SCRIPT>

 if (confirm("Confirm changes?"))

  window.document.forms[0]._F1913U.value = 'Y'

 else

  window.document.forms[0]._F1913U.value = 'N';

 if( ONSUBMITFUN() == true) window.document.forms[0].submit();</SCRIPT>

<!--(_d1904) End of dependent confirm prompt-->
```

In this example, if 'CONFIRM' appears at screen offset 1904 (that is, if the confirm prompt is showing on the green-screen) the selected information is included in the HTML page sent to the browser only .

Following is an example of the generated HTML for a CA 2E sub-file select prompt:

```
<!-- (_d0722I) Dependent display when input field -->

  <SELECT NAME="_F0722">

    <OPTION SELECTED VALUE="(_v0722)">(_v0722)

    <OPTION VALUE="4">Delete</OPTION>

  </SELECT>

<!-- (_d0722X) Otherwise display output field -->

  (_f0722)

<!-- (_d0722) End of dependent output -->
```

In this example, the drop-down list is included in the HTML page sent to the browser only if an input field appears at screen offset 722 (that is, if the sub-file select field itself is on the green-screen and is input-capable). If the sub-file select field is not input-capable (when the confirm prompt displays), then the output data at the same offset (in green) is outputted instead.

Formatted Field Tags

Usage

Field tags are used where the value from a field displays on the screen (as with a variable tag), but where the attributes of that field are not known prior to runtime. For instance, if a field is defined in the display file DDS as having a number of possible attributes, each dependent on a different indicator, then the field tag is used instead of the variable tag. The resulting tag is the same as the results of a variable tag, but include various field attributes, such as font color and underlining.

Format

Field tags have the following format:

`(_fNNNN)`

Example

The following display file DDS is being analyzed:

```
A            Z1ABNC        5  0B  4 19TEXT('KeyFld 1 NBR')
A                                      CHECK(RZ)
A                                      EDTCDE(4)
A N25                                  OVRDTA
A  31                                  DSPATR(RI PC)
A N31                                  DSPATR(UL)
A  85                                  DSPATR(BL)
A N25                                  OVRATR
```

Because the Z1ABNC field could appear in several different ways (reverse-image, underlined, blinking) the MLS generator generates a Field tag as follows:

`[assign the value for TD in your book](_f0259)</TD>`

The server determines the style and formatting for this field by looking at the attribute byte for the field (which is passed in the data-stream with the field and changes as the field displays in different ways).

Thus, where the value of the field at screen is 00012, the server might render the above as:

`[assign the value for TD in your book]<B><FONT COLOR=#00FFCC>00012</FONT></B>[assign the value for TD in your book]`

or as

`[assign the value for TD in your book]<I><FONT COLOR=GREEN>00012</FONT></I>[assign the value for TD in your book]`

Exit Program Tags

Usage

In addition to the normal flow of processing created by the underlying application, exit program tags allow the Web Option server to call external IBM i programs, written in any High-Level Language (HLL). The exit program can return a single 254-character value, which displays on the HTML in place of the tag itself. A sample exit program YEXITPGM (written in CL) is shipped with the Web Option product. If a value is to be returned from the exit program, it is the job of the programmer to ensure that this value is sent from the exit program as a program message in message id W2U0001 in message file YW2EMSG in Y2WEB.

Format

Exit program tags take the following format:

```
(_qprogram-name:parameter:parameter:parameter:…)
```

Program-name identifies the program (qualified or unqualified) on the IBM i and *parameter* is the parameter to that program. Any number of parameters can be passed, as long as they are all character values and are separated by semi-colons.

Example

Following is an example of an exit program tag that has been added to a generated HTML skeleton:

```
(_qPRODLIB/LOOKUP:WRK:(_v0345):RTN)
```

This tag causes the server to call program LOOKUP in library PRODLIB passing three parameters: WRK, the value of the screen at offset 345, and RTN.

Program LOOKUP can use these values to perform whatever processing it chooses and, if it returns a value, the following line (or HLL-specific equivalent) is coded in it:

```
SNDPGMMSG  MSGID(W2U0001) MSGF(YW2EMSG) MSGDTA(&MSGDTA)
TOPGMQ(*PRV) TOMSGQ(*TOPGMQ) MSGTYPE(*INFO)
```

Once the server has called the specified exit program, it attempts to receive message W2U0001. If it receives this message, it displays the contents of the message on the screen in place of the exit program tag; otherwise, the tag does not display.

The value returned from the exit program to the Web Option server can be any text string up to 254 characters. Depending on the processing performed by the exit program, it might be a value from a database, or a simple completion message or even an HTML string.

User-Defined Text Tags

Usage

User-defined text tags are not included in generated skeletons by default, but you can add them where you want to include data (that is not passed in the data-stream and is not a system variable) in the final HTML page.

Format

User-defined text tags have the following format:

```
(_tAAAAAAAAAA)
```

Where *AAAAAAAAAA* is a string of up to 10 characters that identify a particular element of user-defined text, held on the User-defined Text field YMLSUDTP. Each record on this file has a 10-character key and a 254-character data field (it is this data field that replaces the tag).

Example

If you wanted to display the latest sales figures for your company on the sign-on screen and you also want to be able to update this figure on a daily basis without changing the skeleton, you could code the following in the generated skeleton:

```
<H1>Our latest sales have climbed to $(_tSALESFIG)!!!</H1>
```

You also need to add a record to the YMLSUDTP file as follows:

UDTNAME	UDTTEXT
SALESFIG	1,234,567.89

The line would be included in the HTML page that is sent to the browser as:

```
<H1>Our latest sales have climbed to $1,234,567.89!!!</H1>
```

Any change to the UDTTEXT value in the YMLSUDTP file would be reflected every time the sign-on screen is accessed.

Variable Tags

Usage

Variable tags are generated in a skeleton wherever a value from a field on the green-screen needs to be inserted into the final HTML page and where the attributes for that field are known prior to runtime (or are unimportant). Compare Variable tags with Formatted Field Tags in this appendix.

Format

Variable tags have the following format:

```
(_vNNNN)
```

Example

A line might be generated into a skeleton as follows:

```
[assign the value for TD in your book]<FONT COLOR=#FF00CC>(_v0242)</FONT></TD>
```

In this case, the server replaces the (_v0242) tag with the value of the input field at screen offset 242. For instance, if an output field exists on the screen at offset 242 with the value “Select one of the following:” then the previous line would display in the browser as:

```
[assign the value for TD in your book]<FONT COLOR=#FF00CC>Select one of the
following:</FONT></TD>
```

System Variable Tags

Usage

System variable tags are used throughout generated skeletons to identify where system variable values are inserted. System variables are values that are specified by the Web Option system and are used to hold default values, screen and text colors, folder names, and so on.

Format

System variable tags have the following format:

```
(_xAAAAAAAAAA)
```

Where *AAAAAAAAAA* is a string of up to 10 characters that identifies a particular system variable.

Example

Every HTML skeleton generated by CA 2E contains the following lines:

```
<SCRIPT LANGUAGE="JAVASCRIPT" SRC="HTTP:///WEB2EDOC/...
<FORM METHOD=”POST” ACTION=”(_xCALLURL)”...>
```

At runtime, the CALLURL value is replaced by the 'calling-url' value stored for the client session when the Web Option is first accessed and the (_xSYSTEM) value is replaced by the Web Option YSYSTEM control value. For instance, if the YSYSTEM Web Option CONTROL value is myas400:4100 and the URL used to access Web Option the first time was http://myas400:4100/WEB2E, the HTML page sent to the browser would display the previous lines as:

```
<SCRIPT LANGUAGE="JAVASCRIPT" SRC="HTTP://WEB2EDOC/...
<FORM METHOD=”POST” ACTION=”http://WEB2E...>
```

Users can include user-defined substitution variables in HTML pages by specifying them within System Variable tags. For instance, the following tag would be replaced at runtime by the _w2evsn substitution variable:

```
(_x_w2evsn)
```

Some substitution variables are defined by Web Option and others can be created by users in scripts. See The GET and PUT commands in the appendix “Scripting” for more details on how to create your own substitution variables in a script.

Replacement Language Data Tags

Usage

Replacement language data tags are used within drop-down list option tags and command key buttons, where the data to be displayed in the drop-down list options must be retrieved at runtime from the Y2VLLSP CA 2E application object Values List File or from a specified Prompt Message File. The names of the Values List File and Prompt Message File from which to retrieve data are typically set by the Web Option server when it processes a Web Option control comment line embedded in the skeleton source. The libraries that are used for the Values List File and Prompt Message file are taken from the lib-names that are specified in the Web Option control comments -vallstlib and -pmtmsgflib.

Format

Replacement language data tags have the following format:

```
(_rNNNNNNNAAA...)
```

or

(_rXXXXXXX)Where *NNNNNNN* is a 7-digit list surrogate number from the Values List File and *AAA…* is a string of up to 25 characters that identifies the external value from the file or where XXXXXXX is a message identifier in the prompt message file specified for the skeleton in which the replacement tag is found. At runtime, the list surrogate and the external value are used to lookup the Values List File and are replaced with the Condition text for that record or the prompt message identifier is used to lookup the specified Prompt Message File and the message text for the message is returned.

Example

The following drop-down list options are generated into a skeleton:

Drop-down lists:

```
<select name="_F0345U">
        <option value="O">(_r1234567O)</opyion>
        <option value="C">(_r1234567C)</opyion>
        <option value="D">(_r1234567D)</opyion>
</select>
```

At runtime, the three replacement data tags are replaced with the condition text in the Values List file for the three records for list surrogate 1234567 which have external values of O, C and D. In an English language environment, the Values List file might contain condition text values of Open, Closed and Deleted. However, if the user were running Web Option in a French language environment, the Values list file would contain Ouvert, Fermée and Rayée. Thus a single skeleton can be easily used with different national languages.

Command key text:

For a command key button F14, the following HTML would be generated:

```
<td bgcolor="#FFFFFF" height="20">
<button type="submit" id="_K14" class="cmdkey" title="CF14"
onclick="name=id">(_r100151414)</button>
</td>
```

At runtime the replacement tag is replaced with the condition text in the Values List file for the record with List Surrogate 1001514, which has external value 14. In an English environment, the condition text for F3 might be Exit. However, if the user were running web option in a French environment, the corresponding record in the Values List file in the French library would be 'Sortir.'

Web Option HTML Input Element Format

When a button is pressed on a browser screen, all HTML input elements are sent to the Web Option server for processing and routing to the correct interactive job. Data that the Web Option router processes must be one of the following element types:

Input Element	**Description**
Input field	HTML names beginning with _F, (for example, _F0453U)
Command key	HTML names beginning with _K (for example, _K04)
Macro	HTML names beginning with _M (for example, _M1607:90)
Script	HTML names beginning with _S, (for example, _SREPEAT)

Input Fields

Input fields are defined in the HTML as having a name in the format _F*nnnnxxxx* where *nnnn* is the offset to that field's position on the underlying green-screen and *xxxx* is an optional string of characters that the browser uses to identify the field type so it can be initially validated at the client (browser) level using JavaScript.

Examples

The following field is a 10-character input field that allows only uppercase input (specified by the U in the field name) and that initially appears on the screen with a blank value:

```
<INPUT TYPE=”TEXT” NAME=”_F0432U” VALUE=”” SIZE=”10” MAXLENGTH=”10”>
```

The following field is a 7-character input field, which allows only numeric values to be input and initially appears on the HTML page with a value of 22.35:

```
<INPUT TYPE=”TEXT” NAME=”_F0586N” VALUE=”22.35” SIZE=”7” MAXLENGTH=”7”>
```

The following field is a drop-down list having two acceptable values and that initially appears on the HTML page with a value of 'Yes' (the underlying green-screen appears with an initial value of 'Y'):

```
<SELECT NAME=”_F1234”>
   <OPTION SELECTED VALUE=”Y”>Yes</OPTION>
   <OPTION          VALUE=”N”>No</OPTION>
</SELECT>
```

When a submit button is pressed, all changed input fields (including drop-down select fields) are validated to ensure their contents conform to their field data type prior to submitting the page. If any field contains data that is inconsistent with the field data type (for instance, if you have entered character data in a field defined as being a numeric field), a pop-up error displays and the page is not submitted to the IBM i.

Once all changed fields contain acceptable data, the page is submitted to the IBM i, and all input fields are sent with their respective values.

The *value* attribute of input fields refers only to the value of the field when it initially displays on the HTML page.

Command key buttons

Each command key is created in the HTML skeleton or in a JIT page as a button, for example:

```
<button type="submit" id="_KEN" class="cmdkey" onclick="name=id">Enter</button>
```

If a button is created to simulate.

Where button-text is the text that appears on the button face itself (for instance, Cancel or Select for processing), nn is a 2-letter code for the relevant command key, and xxxx is the (optional) cursor position on the screen when the command key is pressed.

For generated skeletons, the button text is taken from the DDS source; for JIT screens, it is taken directly from the 5250 screen. The available 2-letter command-key codes are as follows:

01-24-**Relevant F-key**

EN-**Enter**

HL-**Help**

CL-**Clear**

UP-**Page Up**

DN-**Page Down**

PR-**Print**

If required, the button can include a cursor position on the screen when the command key is pressed, by including it in the button id, as follows:

```
<button type="submit" id="_K04xxxx" class="cmdkey"
onclick="name=id">Prompt</button>
```

xxxx

Specifies the cursor offset to be sent to Web Option when this button is pressed. This allows you to create buttons that simulate pressing, for example, F4, with the cursor positioned at a specific input field.

If the cursor position is not specified, Web Option assumes that the cursor is positioned at the first input field (this is consistent with green-screen processing). If cursor position is specified, the Web Option router sends both the relevant command key and the cursor position to the IBM i.

Although both the HTML generator and the JIT processor create buttons to provide the same basic functionality as would be available on the green-screen, you can add additional button definitions to generated HTML skeletons to gain further functionality than the HTML generator can provide (such as cursor-specific buttons).

Examples

The following are examples of button definitions you can add to generated HTML skeletons.

- Generated button to simulate a user pressing F3 (as generated into JIT screens and HTML skeletons):

  ```
  <INPUT TYPE="SUBMIT" NAME="_K03" VALUE="Exit">
  ```

- Generated button to simulate a user pressing Enter (as generated into JIT screens and skeletons):

  ```
  <INPUT TYPE="SUBMIT" NAME="_KEN" VALUE="Enter">
  ```

- User-created button to simulate a user pressing the help key while positioned at offset 1202 (button display = ?):

  ```
  <INPUT TYPE="SUBMIT" NAME="_KHL1202" VALUE="?">
  ```

- User-created button to simulate a user pressing F4 (Prompt) while positioned at offset 845 (button display = …):

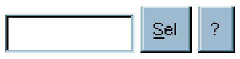
  ```
  <INPUT TYPE="SUBMIT" NAME="_K040845" VALUE="...">
  ```

As in the examples above, you can add command keys to generated skeletons to provide functionality that is not automatically generated (such as field-sensitive help). For instance, the following HTML code displays an input field (which only allows uppercase input), with both a Select button (to mimic existing F4-prompt functionality) and a ? button (to mimic field-sensitive help) alongside it:

```
[assign the value for TD in your book]
<INPUT TYPE="TEXT" NAME="_F0452U" VALUE="" SIZE="10">&nbsp;<INPUT TYPE="SUBMIT"
NAME="_K040452" VALUE="Sel">&nbsp;<INPUT TYPE="SUBMIT" NAME="_KHL0452" VALUE="?">

</TD>
```

The &nbsp; is an HTML non-breaking space, allowing a small gap between the input field and each button. The previous code would display as:



Before adding additional buttons to HTML skeletons, ensure that the underlying processing exists-in the previous example for instance, ensure that the F4-prompt functionality is active for the field in question.

When a page is submitted to the IBM i, only the command key pressed is sent to the Web Option router, along with all the input fields. Therefore, an HTML page can contain definitions for buttons representing many different command keys, but only one is included in the input field data sent to the IBM i.

Macro buttons

A macro button allows the user to add a value to an existing field. Macro buttons in the Web Option are designed so they specify both the location of the (existing) command-line or input capable field, and the string to pass to the IBM i in that field. Macro buttons are not generated into any Web Option skeletons by default, but can be added to a skeleton post-generation.

Macro buttons are coded as follows:

```
<INPUT TYPE=”SUBMIT” VALUE=”button-text” NAME=”_Mnnnn:data-string” >
```

where *button-text* is the text that displays on the button itself and *nnnn* is the offset to the input field in which *data-string* is entered. The single colon separating the input field offsets the data-string.

Examples

You can change the default Y1MENU_O.HTM screen so that the command line is removed (so external users cannot access the command line), but you can add a macro button to enable users to execute a specific command. Even if you remove the command-line input field from the HTML page, you can add the following button to the Y1MENU_O skeleton:

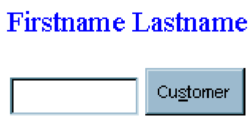
```
<INPUT TYPE=”SUBMIT” VALUE=”Change profile” NAME=”_M1607:?QSYS/CHGPRF”>
```

A macro button can only be used to insert a single pre-specified string into a single input-capable field on a screen. However, the string can be a Web Option tag, so the following HTML code can copy the contents of one output field to a different input field and submit the page to the IBM i for processing:

```
<FONT COLOR=”BLUE” SIZE=”+2”>(_v1513)</FONT>
<INPUT TYPE=”TEXT” VALUE=”(_v1593)” NAME=”_F1593” SIZE=”20”>&nbsp;
<INPUT TYPE=”SUBMIT” VALUE=”Customer” NAME=”_M1593:(_v1513)”>
```

When the page displays in the previous case, if the contents of the output field on the green-screen at offset 1513 are “Firstname Lastname”, then the HTML output is:

```
<FONT COLOR=”BLUE” SIZE=”+2”>Firstname Lastname</FONT>
<INPUT TYPE=”TEXT” VALUE=”(_v1593)” NAME=”_F1593” SIZE=”20”>&nbsp;
<INPUT TYPE=”SUBMIT” VALUE=”Customer” NAME=”_X1593:Firstname Lastname”>
```



The user now has the option of either entering a new name in the _F1593 field or pressing the Customer button, which submits the page to the Web Option router with the _F1593 field having the value Firstname Lastname.

Script buttons

Like Macro buttons, Script buttons perform 'hidden' processing. However, a Script button actually processes a Web Option script, which is running on the IBM i. A script consists of a number of commands, which are held in the YSCRIPT source file in Y2WEB. Scripts can be written to do the following:

- Execute a series of commands on the IBM i.
- Retrieve data from one screen, change to a different screen and enter the saved data on the new screen.
- Drop the user back out through several screens to a predefined point.

A Script button is coded as follows:

```
<INPUT TYPE=”SUBMIT” VALUE=”button-text” NAME=”_Sscript-name” >
```

where *button-text* is the text that displays on the button itself and *script-name* is the name of the script member in Y2WEB/YSCRIPT.

Examples

A button could be coded as follows:

```
<INPUT TYPE=”SUBMIT” NAME=”Execute my script” VALUE=”_SMYSCRIPT”>
```

This button would submit the HTML page and execute the MYSCRIPT script (held as a member called MYSCRIPT in Y2WEB/YSCRIPT).

When an HTML page is submitted by a Script button any of the data on that page is available to the script processor.

Appendix D: Markup Language Skeleton Customization

This chapter identifies the modifications that can be made to Web Option generated Markup Language Skeletons. Use these instructions to modify generated skeletons to include some features that are not generated by the MLS generator.

Note: Certain sections in this document discuss the manual editing of Web Option file data to achieve changes in generated skeletons. Until Web Option includes processing to enable this automatically, this is the only way that you can edit these files. We strongly suggest that you make copies of the following files prior to changing any data:

- **YMLSHDRRFP**-MLS syntax header file
- **YMLSSYNRFP**-Markup language skeleton syntax file

Additionally, it is assumed that any person making changes that affect the generation of Web Option HTML skeletons is proficient in HTML and understands the effects of the data changes they are making.

This section contains the following topics:

Overview

This chapter is divided into three sections:

- Model Customization (see page 140)

  Model customization covers the way changes are made within a CA 2E model that affect how the skeletons (or a specific skeleton) are generated. Customization changes made within a CA 2E model are re-used when a skeleton is regenerated.

- Global Customization (see page 148)

  Global customization covers areas where making changes affects the way all skeletons are generated. These include changing Web Option control values and changing the contents of the Markup Language Skeleton Syntax file. Changes made at a global level affect the way the HTML generator works and the data it uses to generate a skeleton. Changes made at a global level normally affect only skeletons generated after the customization changes have been made.

- Specific Customization (see page 151)

  Specific customization covers changes made at the model, field or screen element (screen field) level. Specific customization can include the addition of page-specific images, the conversion of input status fields to drop-down boxes, the inclusion of hyperlinks, and so on.

Model Customization

Integration with the CA 2E Model

From r8.0 onwards, users apply customization directly to screen elements in the model itself, meaning that when a skeleton is generated for a screen that contains those elements, the element-specific customizations are automatically applied to the skeleton.

Element-level customization can be applied at the following levels:

- Field-level

  When a skeleton screen is generated having screen fields, the customization is selected. Screen element customization overrides field level customizations.

- Header/footer-level

  Element customization can be applied to a Define screen format (DFNSCRFMT) function based over the *Standard header/footer file. This customization will apply to every function and allows common customizations to be applied across multiple screens.

- Display-file-level

  When a skeleton screen is generated having display fields, the customizations are selected. Display-file-customization is implemented in place of the normal *BODY and *HEADER elements that are generated for all skeletons.

- Record-format-level

  When a skeleton screen is generated having record format levels, the customizations are selected. Record-format-customization is implemented in place of the first *TR element that is generated for all skeletons.

- Screen-element-level

  When a skeleton screen is generated having a screen-element level, the customizations are selected. Screen-element-customizations are implemented in place of the *INPUTFIELD, *OUTPUTFIELD, *DYNAMICFIELD or *OUTPUTTEXT elements that are generated for the screen element.

Note: Because it is not possible to associate element customization with a specific header/footer, element customization applied to fields on any header/footer will apply to every screen in the model, whether they use that header/footer or a different one.

Model-level customization data is held in two files within Web Option:

YELMCSTRFP

This file contains the actual customization data that is created by users. The data in this file can be used by different elements in different models. The data in this file can include variables, which are replaced at MLS generation time with skeleton-specific data.

YSCRELMRFP

This file contains links between the elements in each model and the customization data held in YELMCSTRFP.

The customization data held in YELMCSTRFP can be edited using the YWRKELMCST (Work with Element Customization Data) command.

Adding Element-level customization

To add element-level customization within a model:

Field-level customization

Note: This assumes a field exists in your model called State code, which is a two-letter status field that holds the US state codes (CA, CO, etc.). Values exist for this field for each of the 50 US states, to create a drop-down list:

1. Display the state names instead of the two-letter state codes
2. Default the option to blank and signal the user that they must select a state
3. Only allow the user to pick California, New York and New Jersey

Example

1. Use the YWRKELMCST (Work with Element Customization) command to display the various types of element customization.
2. Use F6 to create a new record with the following values:

```
Customization ID: DDL_STATE           Sequence no.:   1.00


Customization information:  <select name="&&_fldnam">

Customization type: I  (Input/Output)

Object type/attribute: FLD STS

Customization description: Drop-down list for state codes


 F3=Exit  F12=Cancel  F13=Fast Exit
```

3. Use F6 to create further DDL_STATE records so the following data records are created in YELMCSTRFP:

```
Customization ID      Customization information

DDL_STATE  001.00     <select name="&&_fldnam">
DDL_STATE  002.00     <option selected value="  ">*Select*</option>
DDL_STATE  003.00     <option value="CA">California</option>
DDL_STATE  004.00     <option value="NY">New York</option>
DDL_STATE  005.00     <option value="ND">North Dakota</option>
DDL_STATE  999.00     </select>
```

4. Enter the CA 2E model and edit the details of the State code field
5. Press F17 from the EDIT FIELD DETAILS screen to display the Edit Screen Element Customization screen. This screen shows the element details (the fact that it is a field of type STS). Type DDL_STATE in the Customization Identifier input field, press Enter and confirm.

```
Element Name: AFCD       State code

   Object Type: FLD  Object Subtype: CDE


   Customization Identifier:  DDL_STATE           (Use '?' to select)

   Customization Description: Drop-down list for state codes


   F3=Exit  F12=Cancel  F13=Fast Exit
```

6. Generate a skeleton for a function where a screen for that function includes the State code field, the customization data you entered in steps 2 and 3 is generated into the skeleton, with the &&_fldnam in the <SELECT> tag being replaced with _Fnnnn where nnnn is the screen offset of the state field on the screen.

   For instance:

```
<!--ROW=010 COL=070-->
  <TD COLSPAN="004" HEIGHT="020">
  <!-- Field-level element customization -->
  <select name="_F0790">
     <option selected value="  ">*Select*</option>
     <option value="CA">California</option>
     <option value="NY">New York</option>
     <option value="ND">North Dakota</option>
  </select>
  </TD>
```

7. When the skeleton is merged with live data into an HTML page and sent to the browser, the drop down list displays as a drop-down list with *Select* being displayed.

File-level customization

File-level customization gives a specific look-and feel to an entire HTML page. Under normal circumstances, where there is no file-level customization, the following two lines are generated in each skeleton:

```
<BODY BGCOLOR="WHITE" TEXT="#000000" LINK="#0000FF" ALINK="#CC33FF"
VLINK="#FF0000" onload=" ONLOADFUN()">
<IMG SRC="HTTP://WEB2EDOC/CA_WH.GIF" WIDTH="255" HEIGHT="60"><DIV
ALIGN="CENTER"><IMG SRC="HTTP://WEB2EDOC/ADV2E.GIF" WIDTH="200" HEIGHT="60"></DIV>
```

The above code is generated from the *BODY and *HEADER object groups in the MLS Syntax file and the YDFTBGC, YDFTTXC, YDFTLKC, YDFTALC, YDFTVLC and YHTMOSF Web Option values.

If file-level customization is applied to a display file for a CA 2E function, then skeletons generated for screens for that function do not include the above lines. Consequently, file-level customization must at least consist of a <BODY> tag, which is required by HTML.

Example

1. Use the Work with Element Customization (YWRKELMCST) command to display the various types of element customization.

2. Use F6 to create a new record with the following values:

```
Customization ID: BODY_BG_PINK        Sequence no.:   1.00

Customization information: <BODY BGCOLOR="PINK" onLoad="ONLOADFUN()">

Customization type: O  (Input/Output)

Object type/attribute: FIL DSP

Customization description: Pink background for web pages

F3=Exit  F12=Cancel  F13=Fast Exit
```

3. Enter the CA 2E model and take option 'S' to a function to display the screen for that function. Press F3 to display the EDIT FUNCTION DEVICES screen.

4. Press F17 to display the Edit Screen Element Customization screen. This screen shows the element details (the fact that it is a display file). Type BODY_BG_PINK in the Customization Identifier input field, press Enter and confirm.

```
Element Name: UUA8EFKD    Edit Address             Edit file

Object Type: FIL  Object Subtype: DSP


Display File:  UUA8EFKD    Edit Address             Edit file

Customization Identifier:  BODY_BG_PINK         (Use '?' to select)

Customization Description: Pink background for web pages

F3=Exit  F12=Cancel  F13=Fast Exit
```

5. If you generate a skeleton for this function, the customization data entered in step 2 are generated into the skeleton:

```
</HEAD>
<!-- File-level element customization -->
<BODY BGCOLOR="PINK" onLoad="ONLOADFUN()">
<CENTER>
```

Format-level customization

Format-level customization allows you to give a specific look-and feel to record formats (such as the subfile control format) in an HTML page.

Record-level customization replaces the <TR> tag that is generated for the record format. If the record format covers more than one line, the record-format customization applies to all lines within the record format. Consequently, format-level customization must at least consist of a <TR> tag, which is required by HTML.

Example

1. Use the Work with Element Customization (YWRKELMCST) command to display the various types of element customization.
2. Use F6 to create a new record with the following values:

```
Customization ID: FMT_BG_YELLOW        Sequence no.:   1.00

Customization information: <TR BGCOLOR=”YELLOW”>

Customization type: 0  (Input/Output)

Object type/attribute: FMT ALL

Customization description: Yellow background for record formats

F3=Exit  F12=Cancel  F13=Fast Exit
```

3. Enter the CA 2E model and take option 'S' to a function to display the screen for that function. Press F17 to display the DISPLAY SCREEN FORMATS screen. Select a record format (for instance, the Subfile control format).
4. Press F17 to display the Edit Screen Element Customization screen. This screen shows the element details (the fact that it is a subfile control record format). Type FMT_BG_YELLOW in the Customization Identifier input field, press Enter and confirm.

```
Element Name: ZSFLCTL     Subfile control.

Object Type: FMT  Object Subtype: CTL

Display File:  UUA8EFKD    Edit Address              Edit file
Record Format: ZSFLCTL     Subfile control.

Customization Identifier:  FMT_BG_YELLOW        (Use '?' to select)

Customization Description: Yellow background for record formats

F3=Exit  F12=Cancel  F13=Fast Exit
```

5. If you generate a skeleton for this function, the customization data entered in step 2 are generated into the skeleton:

   ```
   <!-- FORMAT:ZSFLCTL (Subfile control record) -->
   <!-- Format-level element customization -->
   <TR BGCOLOR="YELLOW">
   <TD COLSPAN="002" HEIGHT="020">
   </TD>
   ```

Element-level customization

Note: This assumes we are using the same State code field for which we generated field-level customization. In this case, we want to use it in a different screen and have the following attributes:

1. Display the state names instead of the two-letter state codes
2. Default the option to California
3. Only allow the user to pick California or New York

Example

Use the YWRKELMCST (Work with Element Customization) command to display the various types of element customization.

1. Use F6 to create a new record with the following values:

```
Customization ID: DDL_STATE_2          Sequence no.:   1.00

Customization information:  <select name="&&_fldnam">

Customization type: I  (Input/Output)

Object type/attribute: FLD STS

Customization description: Drop-down list for state codes (CA/NY only)
```

```
F3=Exit  F12=Cancel  F13=Fast Exit
```

2. Use F6 to create further DDL_STATE_2 records so the following data records are created in YELMCSTRFP:

   ```
   DDL_STATE_2  001.00     <select name="&&_fldnam">
   DDL_STATE_2  002.00     <option selected value="CA">California</option>
   DDL_STATE_2  004.00     <option value="NY">New York</option>
   DDL_STATE_2  999.00     </select>
   ```

3. Enter the CA 2E model and take option 'S' to the new function to display the screen for that function. Move the cursor to the State code field and press Enter to display the EDIT SCREEN ENTRY DETAILS screen.

4. Press F17 to display the Edit Screen Element Customization screen. This screen shows the element details (the fact that it is a status field on a particular screen). The Customization Identifier input field is already filled in with DDL_STATE (because it has been implicitly selected at the field-level). Overtype DDL_STATE with DDL_STATE_2, press Enter and confirm.

```
Element Name: AFCD       State code

Object Type: FLD  Object Subtype: STS

Display File:  UUA9EFKD    Edit Address            Edit file
Record Format: ZSFLRCD     Subfile record.

Screen position:
   Screen Row: 010
   Screen Col: 070

Customization Identifier:  DDL_STATE_2          (Use '?' to select)

Customization Description: Drop-down list for states (CA/NY only)

F3=Exit  F12=Cancel  F13=Fast Exit
```

From the EDIT SCREEN ENTRY DETAILS screen, move the cursor to the first line of the Column Headings input field and Press F17 to display the Edit Screen Element Customization screen. This screen now shows the element details for the column heading text itself. Enter *DROP ELEMENT in the Customization Identifier field and press Enter and confirm. Repeat for the second line of Column Headings.

```
Element Name: AFCD        *COLHDG1: 'State code'

Object Type: FLD  Object Subtype: STS

Display File:  UUA9EFKD    Edit Address             Edit file
Record Format: ZSFLRCD     Subfile record.

Screen position:
    Screen Row: 010
    Screen Col: 070

Customization Identifier:  *DROP ELEMENT        (Use '?' to select)

Customization Description: Drop element - do not generate element

F3=Exit  F12=Cancel  F13=Fast Exit
```

If you generate a skeleton for a function where a screen for that function includes the State code field, the customization data you entered in steps 2 and 3 are generated into the skeleton, with the &f in the <SELECT> tag being replaced with _Fnnnn where nnnn is the screen offset of the state field on the screen. Additionally, the heading text that would normally be generated into the subfile control is not generated. For instance:

```
      <!--ROW=010 COL=070-->
        <TD COLSPAN="004" HEIGHT="020">
        <!-- Field-level element customization -->
        <select name="_F0790">
        <option selected value="CA">California</option>
        <option value="NY">New York</option>
        </select>
        </TD>
When the skeleton is merged with live data into an HTML page and sent to the browser,
the drop down list displays *Select* being displayed.
```

Note: If the Web Option control value YVARTYP is set to *ALL or *SHORT, then all short substitution variables within element customization must be specified without an immediately following alpha character (a-z orA-Z). Any other character (including a blank, single quote or double quote) is acceptable.

If it is required to have, for example, "...value=&vxyz..." and the '&v' should be converted to the relevant value, then the YVARTYP control value should be set to *ALL and the &v should be replaced with its equivalent long-form substitution variable as "...value=&&_fldvalxyz...".

Global Customization

The HTML Generation Process

HTML skeletons are generated from the display file DDS, which is itself, generated from data in the model. Each skeleton that is generated is given a unique name in the format H*nnnnnnn* (where **nnnnnnn** is a unique number greater than 0001025) and is created as a source member in the YMLSSRC MLS source file in the Y2WEB Web Option product library.

The HTML generator is data-driven-that is, the generator program has been written so that the generation process is controlled by data held externally to the program. While some of the HTML that is generated is screen-specific (such as the actual fields on a screen and screen titles), much of it is generic (such as screen/font colors and page headings/footings). This has two advantages:

- You can change the HTML code so that all skeletons are generated with the same look-and-feel, possibly including your own corporate logos, internal website links, and so on.
- You can use the HTML generator with different sets of data to generate other markup languages in the future (for example, Extensible Markup Language-XML).

Most of the generic HTML code generated is held in the YMLSSYNRFP Markup Language Skeleton Syntax file; changes made to data held in this file affect all skeletons that are generated. This file is owned by the YMLSHDRRFP MLS Header file, which is actually referenced by the HTML generator itself. As the generator parses the DDS source member for a display file, it determines the type of object (field, record format, and so on) it is processing, and uses the data held in YMLSHDRRFP and YMLSSYNRFP to generate the correct type of data for that object.

All generated HTML files include the following sections (each of which corresponds to a record on YMLSHDRRFP, which itself corresponds to one or more records on YMLSSYNRFP) generated in order:

- ***HTML**-This section consists of the HTML element, which is generated at the top of each skeleton.
- ***HEAD**-This section consists of a number of lines, covering the HEAD element of the HTML skeleton. The HEAD element contains several sub-elements, such as TITLE, STYLE, SCRIPT, and META.
- ***BODY**-This section consists of the <BODY> start tag, whose attributes control the background color of the page, various default colors for hypertext links, and so on.
- ***HEADER**-This section consists of a number of lines (maximum 100) that can be used to provide a standard header for the top of every HTML page. Typically, this section would contain HTML tags to display company logos, hyperlinks (perhaps to pages on the company website), and anything else that you wish to display on all HTML pages. This section can contain certain Web Option tags, such as System tags, Exit program tags, and User-defined text tags.
- ***FORM**-This section consists of the data, HTML tags, and Web Option tags that describe the specific screen for which the skeleton is being generated.
- ***FOOTER**-This section, like the *HEADER section, consists of up to 100 lines of HTML which can be used to provide a standard page footer for every HTML skeleton.

In terms of basic layout, every generated HTML skeleton uses the following format:

```
*HEADER
(generic)


*FORM
(screen-specific)

*FOOTER

(generic)
```

The *HEADER and *FOOTER sections in YMLSSYNRFP are shipped with data and includes an example company logo, text, and sample hyperlinks.

Editing the YMLSSYNRFP File

To make global customization changes, the data in the YMLSSYNRFP file needs to be edited. This can be done using the YEDTMLSSYN (Edit MLS Syntax) command.

Changing Web Option Formatting Control Values

The Web Option YW2EVALRFP Control Values file contains a number of control values specifically used to control the way HTML skeletons are generated and the way JIT screens display.

These formatting values can be edited by using the following command:

YWRKW2EVAL W2EVAL(*ALL) VALTYP(*FMT)

Changes to these values affect any skeletons that are subsequently generated. In some cases, changes also affect the way JIT screens display.

Specific Customization

Editing an HTML Skeleton After Generation

When an HTML skeleton has been generated, it exists as a source member in YMLSSRC in Y2WEB. While it is possible to edit the skeleton in that form using SEU, it is easier to edit it as an HTML document on a PC. To do this, you must convert the skeleton source member into a document in the WEBOPT directory in the IFS, using the YPRCSKL command.

Once the skeleton is available as an HTML file on the PC, you can edit it using a graphical HTML editor or a text editor. You can make almost any changes to a generated skeleton, but there are particular sections of generated HTML that you should leave in the skeleton:

- The message display line on row 24; the HTML for this is shipped as:

  ```
  <FONT COLOR=RED SIZE=3>(_v1842)</FONT>
  ```

  This line shows any error messages, it is generated from the *ERRORLINE section in the YMLSSYNRFP Markup Language Skeleton syntax file. If you change the *ERRORLINE record in YMLSSYNRFP, you **must** ensure that the (_v1842) tag remains.

- Web Option data tags

  The CA 2E data tags in the skeleton correspond to runtime data, which is only available from the 5250 data stream when the application is running. Some of these tags are for text such as the Program Name and time, which appear on a standard CA 2E generated panel; this type of text does not need to remain on the HTML page for the underlying program to function correctly. However, a lot of the tags correspond to field data that is retrieved from the underlying green-screen at runtime; these tags should be left in the skeleton.

  To identify which tags correspond to file data:

  1. Take a screen print of the function running in the browser and convert the HTML skeleton into an HTML file (which you can copy onto your PC).
  2. Open the HTML file and take a screen print so that you can see where all the 2E tags are.
  3. Compare the two views to determine what tags you need to keep.

- The CONFIRM prompt

  Leave in the CONFIRM prompt; it is usually required in an Update function, if confirmation is required when records are added, changed, or deleted. This is normally defined on line 24 and the HTML is generated from the *CONFIRMPROMPT section in the YMLSSYNRFP Markup Language Skeleton syntax file.

After making your necessary changes to the skeleton, use the YPRCSKL command with option *CVTTOSKL, to copy it over to the HTML source file on the IBM i.

Note: If you customize a function skeleton and copy it to YMLSSRC, then you regenerate the skeleton for that function using the YGENMLS command, any changes you added are lost. At that point, you are back to your basic skeleton.

Tips and Techniques

The following sections provide helpful information including:

- Adding buttons to send e-mail
- Accessing Web Option through another website (such as the Jasmine portal)
- Calling PC programs from Web Option (such as MS Word and MS Excel)
- Adding field-specific help buttons

Adding Buttons to Send E-Mail

A useful feature in many websites allows users to click on a button and send an email, automatically passing information from the screen as data (To, Subject, Message, and so on) into the email itself.

A function called SENDEMAIL is included in the default YSCRIPT.JS file, which has five arguments (parameters): TO, CC, BCC, SUBJECT, and BODY. This function performs all the processing necessary to create an email.

Use the following code to create a button that sends e-mail:

```
<INPUT TYPE=”BUTTON” VALUE=”button-value”
 onclick=”SENDEMAIL(to-value, cc-value, bcc-value, subject-value, body-value)”>
```

To summarize:

- *button-value* is the text to display on the button (for example, “Send e-mail to us”)
- *to-value*, *cc-value*, *bcc-value*, *subject-value* and *body-value* are the values to use when sending the email.

Notes:

- If you need to use hard-coded values, enclose them in single quotes.
- You can specify any of the values as blanks, by not including a value between two quotes.

- You can include any information from elsewhere in the form by specifying the value from that field (as long as that field is named in the form), for example:

```
<FORM...>
...
<INPUT TYPE=”TEXT” NAME=”COMMENTS” VALUE=” ” SIZE=”100”>

<INPUT TYPE=”BUTTON” VALUE=”Send us your comments”
onclick=”SENDEMAIL('firstname.lastname@companyname.com' , '' , '' , 'My comments
on your website' , this.form.COMMENTS.value)”
...
</FORM>
```

  Clicking the Send us an email button opens an e-mail window with:

  – The To: box already set as 'firstname.lastname@companyname.com'

  – The Cc: and Bcc: boxes blank

  – The Subject: line already set to 'My comments on your website'

  – The body of the e-mail containing the text that the user entered in the COMMENTS text-box.

  In general, you can use any field value, by specifying it as:

  `this.form.field-name.value`

  Where field-name is the name of another field on the same form

  – field-name must be in the same case as the field name on the HTML page

  – this.form and value should be in lowercase (as in the 'this.form.COMMENTS.value' previous example)

Accessing Web Option Through Another Website

The URL that you should normally use to initiate a Web Option session:

http://WEB2EDOC/

However, it is possible to sign directly onto Web Option using the following methods:

- Single user sign-on

  In this mode, a call to Web Option using the previous URL results in the router always opening a session with a single specified user id and password (bypassing the normal sign-on screen).

  The user ID and password are specified in the YDFTUSR Web Option control value. To set these values, run the following command:

  ```
  YWRKW2EVAL W2EVAL(YDFTUSR)
  ```

  Then, enter the 20-character string as *'userid   password'*

  Where *userid* is a user ID on that IBM i and *password* is the password for that user ID.

Note: If the user ID is less than 10 characters, it must be padded with blanks, so that the password starts at character 11 of the VALUE parameter. For example, if the user ID and password chosen are DFTUSER and DFTPASS, then the actual YDFTUSR value is:

```
'DFTUSER   DFTPASS   '
```

Setting an YDFTUSR value severely limits the use of Web Option, but it does ensure the strictest level of security (assuming that the specified user has only extremely limited authority on the IBM i. For instance, this is useful if only external customers were intended to use the Web Option.

- Specific user sign-on

  In this mode, you can append a parameter to the basic Web Option URL to allow a specified user to sign on, as follows:

  http://WEB2E?*external-user-name*external-password

  Note that a question mark separates external-user-name from the basic URL (signifying that the following data is the parameter) and a leading asterisk precedes both external-user-name and external-password.

  The external-user-name and external-password are not an actual IBM i user ID and password. Instead, they are external values that are compared to internal values held on file. They can each be up to 15 characters long, and are case-sensitive. When the router receives a parameter that commences with an asterisk, it calls the YW2EUSRRXR user exit program. This program is shipped with the product along with its source, enabling customers to modify it as they wish. YW2EUSRRXR parses out the external-user-name and external-password from the parameter and checks for a record on the YW2EUSRRFP External Users file. A related internal IBM i user ID and password are used to start an IBM i session depending on whether a record exists, whether the external-password is correct, and whether the specified user is currently authorized.

  A typical YW2EUSRRFP record might be:

```
External user ID                  15   username1
External password                 15   password1
Internal user ID                  10   ABC
Internal password                 10   ABCPASS
User description                  50   User Name

Flag 1                             1   N
Flag 2                             9.0        0
Flag 3                            10
Error page                         8   YNOAUTH
```

  In the above case, an external user with the user ID username1 and password password1 has been defined so that they can sign on directly to the IBM i using the user id ABC and password ABCPASS.

  The Flag 1, Flag 2 and Flag 3 fields are generic fields that can be used when you customize YW2EUSRRXR to provide further validation links to your own files. Use the Error page field to specify what error page to display if the user has been de-authorized or does not pass other customer-specific validity checks.

  Information about this exit program is included with the source of the YW2EUSRRXR program in the QRPGLESRC source file in the Web Option product library.

Notes:

- It is possible to pass only an external-user-name, for example:

  http://myas400:4100/WEB2E?*external-user-name

  in which case,YW2EUSRRXR looks for a record on YW2EUSRRFP for the specified external-user-name with an external-password of *NONE. However, an internal IBM i user ID and password must both be specified.

- If the internal IBM i user ID, password, or both are incorrect or do not exist, the user is presented with a sign-on screen, unless you explicitly add user-validation.
- If you specify the internal user ID as *DFTUSR and the internal IBM i password as blank, the current value for the YDFTUSR is used.
- Internal IBM i user IDs and passwords are currently held in plain-text format in YW2EUSRRFP.

Therefore, the following three strategies are possible:

- Everyone automatically uses the default user ID and password (bypassing the sign-on screen).
- Specific users can sign on automatically (bypassing the sign-on screen). All others are either presented with the sign-on screen, automatically sign-on using the default user ID and password (bypassing the sign-on screen), or are presented with the authority rejection screen.
- All users are presented with the sign-on screen.
- All users are presented with the authority rejection screen.

Note: Processing the YDFTUSR and YUSRTYP Web Option control values is done using real-time values-that is, a change to either of these values takes effect immediately and any jobs that access Web Option.

Calling PC Programs from Web Option (Such as MS Word, MS Excel)

Depending on your needs, calling a PC application such as MS Word can be as simple as navigating between HTML pages.

In general, an application such as MS Word has a document suffix associated with it (for example, .doc for Word documents and .xls for MS Excel spreadsheets). Specifying a link to an object with one of these suffixes, therefore, loads that document into the browser window. The document displays exactly as it would in the given application; it cannot be edited in this format.

There is a function called LOAD within the YSCRIPT.JS default JavaScript file in Web Option. Calling this function (passing the IBM i name and a document name) opens a new browser window and loads the specified document, by concatenating “http://”, the passed IBM i name, “/WEB2EDOC/” and the document name. When this new window is closed, control passes back to the original window.

For instance, if you have a Word document called USERDOC.DOC in the Y2WEBDOC folder within QDLS on your IBM i, you could add the following HTML to a page to display that document:

```
<INPUT TYPE=”BUTTON” VALUE=”Display User Doc”
onClick=”LOAD('myas400','userdoc.doc')”>
```

When the button is clicked, a new window opens with the following url:

```
http://myas400/WEB2EDOC/userdoc.doc
```

Although the LOAD function explicitly assumes the document is in Y2WEBDOC (for which the external name is WEB2EDOC), you can change the LOAD function to retrieve documents from other locations. The source for the LOAD function (in YSCRIPT.JS) is:

```
function LOAD(as400,doc)

 new_win = window.open("http://" + as400 + "/WEB2EDOC/" + doc,
     "New_window",
'menubar=no,toolbar=no,location=no,resizable=yes');
 new_win.focus();
}
```

However, you can change this (if you are familiar with JavaScript) to perform document-specific checking, perhaps by loading documents from subfolders within Y2WEBDOC for specific document types.

Using a hyperlink is a simpler method of loading a PC document into a separate window, as in the following HTML example:

```
<HTML>

Click <A HREF=”/WEB2EDOC/userdoc.doc” target=”_blank”>here</A> to see the user
documentation.

</HTML>
```

When the word here is selected in the previous HTML section, a MS Word Document comes up.

Adding Field-Specific Help Buttons

On a green-screen (either dumb-terminal or using emulator software), it is possible to create field-specific help, which can be accessed by pressing a key (such as F1 or the HELP key) when the cursor is at a particular screen location.

This can be mimicked within a generated skeleton by the addition of field-help buttons, if a field is generated as follows:

```
<!--Dynamic Field-->
<TD BGCOLOR="#FFFFFF" COLSPAN="050" HEIGHT="20">
(_f0737)
</TD>
```

The following addition adds a small ? button next to the field in question, that, when pressed, sends a signal to the (IBM i) IBM i to make it believe that the HELP key was pressed and the cursor was situated over the field in question:

```
<!--Dynamic Field-->
<TD BGCOLOR="#FFFFFF" COLSPAN="050" HEIGHT="20">
(_f0737)&nbsp;<INPUT TYPE=”SUBMIT” VALUE=”?” NAME=”_KHL0737”>
</TD>
```

The &nbsp is an HTML non-breaking space that creates a small space between the input field and the button. You can change what is displayed on the button by changing the value of the VALUE attribute Help! in the following example:

```
<INPUT TYPE=”SUBMIT” VALUE=”Help!” NAME=”_KHL0737”>
```

You can experiment by adding Help buttons using various offsets to mimic the cursor-sensitive functionality of the underlying green-screen, even where that cursor-sensitive help is not related to a specific input field.

System Request Functionality

System is available in the web option runtime as follows:

This functionality is made available by a user-defined field called W2E_SYSREQ, which must be added to skeletons either through element customization or manually post-generation. The easiest way is to use element customization and add the shipped customization element *SYSRQS to a screen element on the function Header, which ensures the W2E_SYSREQ field is added to the generated skeleton.

The W2E_SYSREQ field can be defined as either a text field or (ideally) as a select box drop-down list. The W2E_SYSREQ field can take any valid value from the System Request menu or the special value *, which causes the System Request menu to be displayed. If W2E_SYSREQ field exists on a page that has a blank value, it is ignored.

An example of element customization to display an abbreviated System Request menu is as follows:

```
<select name="W2E_SYSREQ" onchange="document.submit();"> <option value="
">Option...<option value="*">Menu<option value="3">Display job<option
value="4">Messages<option value="90">Sign off</select>
```

This element customization has been added to the shipped YELMCSTRFP file in Y2WEBVENG, with a CSTID value of *SYSREQ.

Appendix E: Scripting

This chapter describes the scripting processing that has been developed for Web Option.

The purpose of scripting support is:

- To allow users of Web Option to simplify much of the repetitive input that can be a feature of IBM i interactive programs

  For example, many popular emulators (including IBM's Client Access) allow the creation of small 'macros', which are essentially a series of commands, run automatically when the macro is executed. A macro might perform auto-sign-on or might simply run a standard series of commands on an IBM i command line.

- To allow users of Web Option to mimic web-type functionality

  For example, in an internet shopping application, a customer can often move back and forth between sections of the online 'shop', the different departments, their shopping cart, and so on. By contrast, an IBM i interactive application does not normally have this type of functionality. Green-screen applications tend to be linear in style, that is, users move from screen A to screen B to screen C and can only come back to screen A by passing through screen B again.

Scripting support for Web Option allows the same level of functionality as provided by these macro languages. A Web Option script can:

- Perform screen-specific functionality - entering data into input fields
- Save data from one screen to use on another screen
- Loop through screens

A benefit of Web Option scripting is that it is invisible to the user; script processing is handled within the Web Option server itself, without the screen processing functionality being involved. For example, a script that provides a 'fast path' to back out through several levels of a menu does not display the intervening menu screens in the browser window.

Scripts are held as database source file members in file YSCRIPT in the Web Option product library and can be edited using the IBM Source Entry Utility (“SEU”) or any equivalent source editor.

A Web Option scripting language has been developed, which consists of a number of commands, each of which performs specific processing. An individual script can contain one or more commands. A Web Option script is executed when a script-execution button on a generated skeleton is pressed.

This section contains the following topics:

Web Option Script-Execution Button Format

A script SUBMIT button has the following HTML format:

```
<INPUT
TYPE=”SUBMIT”
VALUE=”button-face-value”
NAME=”_Sscript-name”
>
```

where *button-face-value* is the text displayed on the button itself and *script-name* is the name of the script (up to 10-characters in length). Note that the first two characters of the NAME attribute are '**_S**' (an underscore followed by a capital S).

Script-execution buttons are not generated into Web Option skeletons automatically. The necessary HTML must be added to a generated skeleton to create a script-execution button. Also the executed script that results when the button is pressed must be written.

Example

To define a button to execute a script called MYSCRIPT, you add the following HTML to a generated skeleton:

```
<input type=”submit” value=”Main Menu” name=”_SMYSCRIPT”>
```

Alternatively, an image can be used to submit a script. For example, the following HTML also executes the MYSCRIPT script when the image is selected:

```
<input name="_SMYSCRIPT" type="image" border="0" src=" /WEB2EDOC/image.gif"
alt="Main Menu">
```

Web Option Scripting Control Values

A number of Web Option Control Values are provided to take advantage of scripting support.

- YSCPRCV - Script recovery option
- YSCPERR - Script error page

These control values are described in detail in the appendix Control Values.

Web Option Script Processing

When a script-execution button is selected:

- The page is submitted to the Web Option server
- The Web Option server updates the screen buffer of the application program with any data entered in the fields on the HTML page
- The Web Option server invokes the script-processor
- The script processor processes commands in the script, updating the screen buffer if necessary
- When a SEND command is found, the updated screen buffer is sent to the application program. When a response is received from the application program, the script processor retrieves the new screen buffer from the application program and continues updating the new screen buffer as necessary
- When a QUIT command is found, the script processor ends, the screen buffer is formatted into an HTML page and control is returned to the client.
- If a screen-error occurs as a result of script processing (such as a character value being entered into a numeric field, or an incorrect command key being pressed), script processing ends and the script-error processing occurs.

Web Option Scripting Language

A scripting language has been developed that uses a limited number of commands. All commands in a Web Option script take the following format:

```
command [parameters]
```

Additionally:

- Each command must be on a separate line.
- Different commands can have zero, one, or two parameters, depending on the command.

- Each parameter must be separated by one or more spaces.
- Commands can be entered in lowercase or uppercase.
- Parameters should be entered exactly as they should be processed.
- Comments can be included in a script, if they are on their own line and start with a '#' in the first position on the line.
- Blank lines can be included in a script to aid readability.

The specific commands that follow have their own syntax.

The SEND command

The SEND command is used to send a command key to the IBM i and optionally quit the script. It is used to mimic the user pressing either a command key, the ENTER key, or one of the other 'standard' keys. Both the required command-key parameter and the optional QUIT parameter are case-insensitive. It has the following format:

```
SEND command-key [QUIT]
```

Acceptable command-key values are:

```
01 - 24
ENTER
PAGE-UP
PAGE-DOWN
PRINT
HELP
```

Examples

```
SEND ENTER
SEND 03 QUIT
send page-up
```

Usage

If the command-key value is not supplied or is not one of the above acceptable values, it is assumed that ENTER should be sent (that is, both “SEND” and “SEND JUNK” are equivalent to “SEND ENTER”).

A SEND command sends the current screen buffer, which can contain data changed as a result of previous script commands, for processing by the application program. If the optional QUIT is not specified, the script is still considered to be 'active' and when control returns to the Web Option server from the application program, it resumes script processing from the point following the SEND command. If the optional QUIT is specified, the script is deactivated, and normal interactive processing occurs when control returns to the Web Option server.

Thus, the following two script fragments result in the same overall processing, but the second results in an additional processing loop:

```
SEND ENTER QUIT
```

```
or
```

```
SEND ENTER
QUIT
```

It must be emphasized that if a SEND command is not included in the script, any processing performed by the script to change data on the screen is ignored and is not processed by the application program.

The QUIT command

The QUIT command is used within a script to signify that the script should be ended. Note that if the QUIT command is encountered within an IF-group, the script ends immediately without executing any further commands. It has the following format:

```
QUIT [error-page]
```

Examples

```
QUIT
QUIT YERROR
```

If error-page is specified, that error page displays to the user and their interactive IBM i job is ended. If error-page is not specified, the script ends normally and control passes back to the user.

The SET-FIELD command

The SET-FIELD command is used to set a field on the screen to have a particular value. It has the following format:

```
SET-FIELD field-name field-value
```

where field-name has the following format:

```
Fnnnn (where nnnn defines a screen offset)
```

and field-value can be optionally enclosed in quotes. It must be enclosed in quotes if it contains preceding blanks. The value parameter is left-adjusted.

Examples

```
SET-FIELD _F0453U         CUST001
SET-FIELD _F1242N 12.34
SET-FIELD _F0336 Michael R. Mouse
SET-FIELD _F0336 ' Michael R. Mouse'
```

Usage

The SET-FIELD command allows the user to change the screen data that was submitted as a result of the script-execution button being pressed before the screen is sent to i OS to be processed by the application program. Essentially, it mimics the user typing data into a field. Remember that unless a script contains the SEND command, any processing done within the script is not processed.

The SET-CURSOR command

The SET-CURSOR command is used to set the cursor to a specific screen position (normally prior to sending a command key like F4 or Help). It has the following format:

```
SET-CURSOR cursor-position
```

where cursor-position has the following acceptable formats:

RnnCnn	(where Rnn Cnn defines a screen row and column)
Fnnnn	(where nnnn defines a screen field)

Examples

```
SET-CURSOR _F0453

SET-CURSOR R06C53
```

Usage

The two different options allow users to place the cursor on a specific input field or elsewhere on the screen.

The SET-CURSOR command only has meaning if followed by a SEND command with a cursor-sensitive command-key (such as 04 or HELP).

The IF, IFNOT, ELSE, ENDIF and END-IF commands

The IF and IFNOT commands are used to start an IF-group to provide processing, based on the actual contents of the screen. An IF-group must be ended with an ENDIF or an END-IF command.

If an IF-group contains an ELSE command, then commands between the IF command and the ELSE command are referred to as an IF-group and the commands between the ELSE command and the END-IF command are referred to as an ELSE-group.

The IF command and the IFNOT command are syntactically equal and perform inverse processing. All the rules which relate to the IF command are equally valid for the IF command.

The IF command takes the following general format:

```
IF expression
```

```
where expression is in the general format
```

```
item    value
```

where item can be any of the following:

- A field name in the format _Fnnnn
- An area on the screen
- A counter variable
- The special value '*SID'
- The special value '*USER'

and value is the value with which the item is compared.

If an IF-group expression is satisfied, every command up to the END-IF is executed, followed by the command immediately following the END-IF command. If an IF-group expression is not satisfied, the script processes the ELSE group (if specified) followed by the next command following the END-IF command.

The END-IF command takes either of the following formats:

```
END-IF
ENDIF
```

Item-format 1 - Web Option Screen Identifier (*SID)

```
*SID screen-id | *BLANK
```

The screen is analyzed and if it has previously been identified, the IF-group is processed.

Example 1

```
IF *SID C2E0001472

END-IF
```

The IF-group is processed only if the screen contains a Screen Identifier with a value of C2E0001472.

Example 2

```
IF *SID *BLANK

END-IF
```

The IF-group is processed only if the screen has not previously been identified.

Item-format 2 - Field name

```
field-name value | *BLANK
```

If the given field appears on the screen and it has the specified value, the IF-group is processed.

Example 1

```
IF _F0453U USER001

END-IF
```

The IF-group is processed only if the field appears on the screen and had the value 'USER001' when the script-execution button was pressed.

Example 2

```
IF _F0453U *BLANK

END-IF
```

The IF-group is processed only if the field appears on the screen and was blank when the script-execution button was pressed.

Item-format 3 - Screen area

```
area[length] value | *BLANK
```

If the specified area on the screen has the specified value, the IF-group is processed. The optional length value is ignored unless the value is *BLANK. In all other cases, the comparison length is determined by the length of value.

Example 1

```
IF R02C03 CUSTINQ01

END-IF
```

The IF-group is processed only if the nine characters at row 2, column 3 on the screen are CUSTINQ01.

Example 2

```
IF R02C03L12 *BLANK

END-IF
```

The IF-group is processed only if the 12 characters at row 2, column 3 (that is, the screen area from row 2, column 3 to row 2 column 14) are blank.

Usage

Any commands can be used within an IF-group, except another IF-group. Thus the following is an acceptable script fragment:

```
# If on main menu page, press F3 to drop one level and quit
IF *SID C2E0001472
SEND 03 QUIT
END-IF
```

Item-format 4 - Web Option client session user (*USER)

```
*USER value
```

If the IBM i user profile of the user running the Web Option client session has the specified value, the IF-group is processed. Note that value must be specified in uppercase if *USER is specified.

Example

```
IF *USER CUSTOMER

END-IF
```

The IF-group is processed only if the Web Option session is being run by a user called CUSTOMER.

The GOTO and LABEL commands

The GOTO command is used to provide a simple loop mechanism within scripts. A label must start with a colon (:) followed by the label name. The label name can be a maximum of 10 characters:

```
:label-name
```

`The GOTO command takes the following format:`

```
GOTO label-name
```

where label-name specifies a label defined elsewhere in the same script.

The label identified by label-name can occur before or after the GOTO command. A maximum of 10 labels can appear within a script. If the label specified in the GOTO does not exist in the script, an error occurs.

Examples

```
# Drop back to main screen (identified by C2E0001042)
:LOOP1
SEND 03
IF *SID C2E0001042
QUIT
END-IF
GOTO LOOP1
```

Usage

Users should be careful when using the GOTO command with a prior label to ensure that an infinite loop does not occur within the script. For instance, if users add a script-execution button to a skeleton to execute the above example script, they must be sure that repeatedly pressing F3 from that screen would eventually display the screen that has the C2E0001042 Web Option Screen Identifier on it.

The GET and PUT commands

The GET and PUT commands allow the user to retrieve data from an input or output field on one screen or from a Web Option control value or substitution variable and place it into an input field on the same or a different screen.

The GET command can take one of the following two formats:

```
GET RxxCyyLzz $variable-name
GET _Fnnnn $variable-name
GET substitution-variable $variable-name
```

The PUT command takes one of the following formats:

```
PUT _Fnnnn $variable-name
PUT _Fnnnn value
PUT substitution-variable $variable-name
PUT substitution-variable value
```

where RxxCyyLzz is a row/column/length identifier used where the data being retrieved is either an output field or a piece of static text and $variable-name is a (maximum 10-character) temporary variable name into which the retrieved value is placed. All variable-names must begin with a dollar-sign ($) and can be a maximum of 11 characters, including the initial dollar-sign. Variables are local to a given script and are only active during the processing of that script. Variable values themselves can be a maximum of 228 characters in length.

Substitution variables must begin with two ampersands (&&). If a substitution variable is specified in the GET statement, it can be one of the following:

- A Web Option control value (beginning with a 'Y')

  ```
  GET &&YDFTBGC $BACKCOLOR
  ```

- A user-defined substitution value

  ```
  GET &&_MYVAR1 $MYVAR
  ```

- A text constant from the MLS Text file YMLSTXTRFP

  ```
  GET &&1001060 $modelname
  ```

- A language-specific special value, such as the '#' symbol

  ```
  GET &&HASH $hashsign
  ```

If a substitution variable is specified in the PUT statement, it can be one of the following:

- One of the client-session specific control values YLSTLIB, YLSTFIL, or YPMTLIB

```
PUT &&YLSTLIB FRNLIB
PUT &&YLSTFIL $MYFIL
PUT &&YPMTLIB FRNLIB
```

- A user-defined substitution value

```
PUT &&_MYVAR1 $MYVAR
```

User-defined substitution values can be used within scripts and can be output directly into generated skeletons. For more details about user-defined substitution variables see the User-Defined Substitution Variables section in the appendix “Control Values”.

Examples

```
# save displayed customer ID from screen
# save order number from screen input field
GET R02C03L08 $CUSTID
GET _F1013  $ORDNBR

:LOOP2
SEND 03
# put saved customer ID into field on screen
IF *SID C2E0002436
PUT _F0762 $CUSTID
SEND ENTER
# put saved order number into field on screen
PUT _F0222 $ORDNBR
SEND ENTER QUIT
END-IF
GOTO LOOP2
```

In the above example, when the script-execution button is pressed, the eight characters on the screen at row 2, column 3 on the screen are saved as a local variable called $CUSTID and the contents of the _F1013 field are saved as a local variable called $ORDNBR. F3 is then repeatedly pressed until the screen that has the C2E0002436 Web Option Screen Identifier on it is encountered, at which point the saved $CUSTID value is entered into a field on the screen and Enter is pressed. Then the saved $ORDNBR is entered into a field on that screen, Enter is pressed again and the script ends.

The ADD, DIV, INZ, MULT and SUB ('counter') commands

The ADD, DIV, INZ, MULT and SUB commands are collectively known as 'counter' commands, since they can be used to mimic a named numeric counter within a script. This processing can be used to provide complex iteration within a script.

The counter commands all take the following format:

```
ADD|DIV|INZ|MULT|SUB counter-name counter-value
```

The INZ command is used to initialize (and optionally create) the counter. The INZ command must be used before any of the other counter commands. If the counter already exists (because it was created in a previous INZ command) it is simply initialized. If a counter-value is not specified for the INZ command, the counter is initialized to zero.

The ADD, DIV, MULT and SUB commands are used to modify the value of a counter (which has already been created and initialized with the INZ command). The counter-value must be specified and must be an integer between -32767 and 32767.

Counters can be checked using the IF and IFNOT commands.

Counter names are not case-sensitive

A maximum of 50 counters can be used in any script. Counter values are not saved between scripts

Examples

```
# Use counter to simulate pressing F3 four times
INZ $MYCOUNT
:LOOP
IF $MYCOUNT1 4
QUIT
END-IF
ADD $MYCOUNT1 1
SEND 03
GOTO LOOP
```

Support for User Defined Macros (UDM)

CA 2E has a feature that gives you the ability to record and play User Defined Macros (UDM) from a Web Option page at run time. User Defined Macros allow any user to pre-record keystrokes for any screen at any position. There can be multiple macros defined for the same screen or field combination.

Note: *Macro* and *Script* are used synonymously in this appendix. A User Defined Macro is simply a script by another name, but is not the same as a Macro button defined in the section Web Option HTML Input Element Format.

From versions of Web Option prior to r8.5, the following changes were made to the way scripts are stored as a result of the implementation of UDM support:

1. All users on the system have their own members in YSCRIPT. The members can contain multiple scripts. Any script that a user creates is stored in this member.
2. A default member called DEFAULT also exists in YSCRIPT. This member can contain multiple scripts and allows the users to add scripts only manually. The scripts contained in this member are available to all users for running. Users cannot edit scripts in the DEFAULT member. If a user selects any of these scripts for the Delete or Edit option, the action is ignored. However, you can copy these scripts between the DEFAULT and user-specific members for editing purposes.

Note: If you have added a new script member for a user to YSCRIPT, you must end and restart the web option server for the scripts to be available for that user in a web option session.

User Interface

You can add UDM support to the skeleton through a new system tag (_xSCPLST). Use the the customization element named *SCPLST to add to a screen element on the function Header. This ensures the _xSCPLST tag is inserted in the skeleton. Otherwise, you can make a change to the MLS Syntax file or edit the skeleton after generation.

Note: If you are adding the _xSCPLST tag manually, ensure you add it after the <form method....> HTML line.

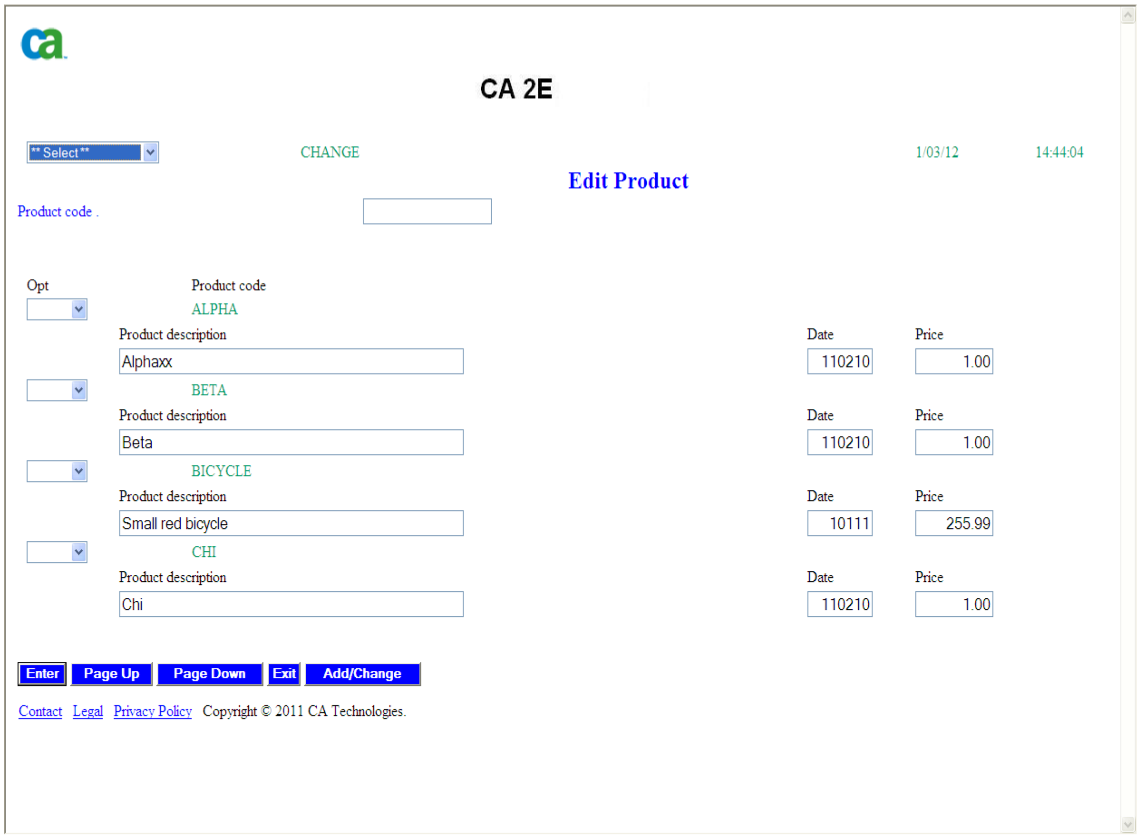
When the Web Option Server processes a skeleton containing the (_xSCPLST) tag, the following additional processing is initiated:

1. The tag processor parses the DEFAULT member in YSCRIPT and loads all the scripts available for the screen being processed.
2. The tag processor parses your member in YSCRIPT and loads all the scripts available for the screen being processed.
3. The scripts are sorted in name order.
4. If you have *ALL authority, certain control options are also loaded. Control options that are loaded depend on the current script processing status.

   For more information about the available control options, see Control Options (see page 176).
5. The *SCPHDR object group (in the MLS Syntax file) is written to the HTML output.
6. The *SCPRCD object group is written repeatedly to the HTML, once for each available script. The Script name and Script description are made available in the &&_scpnam and &&_scptxt substitution variables respectively.

7. The *SCPFTR object group (in the MLS Syntax file) is written to the HTML.

The following example shows the Script Selection DDL:



CA 2E

** Select ** CHANGE 1/03/12 14:44:04

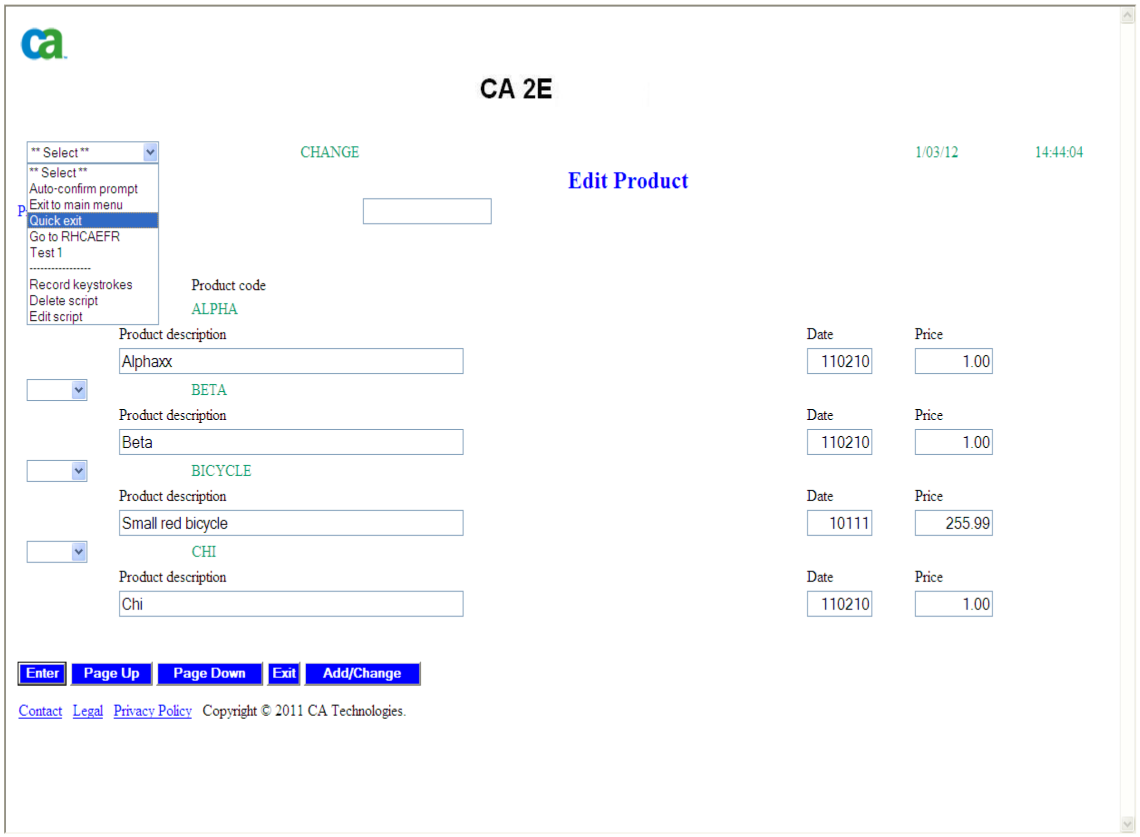
Edit Product

Product code .

Opt	Product code	Product description	Date	Price
	ALPHA	Alphaxx	110210	1.00
	BETA	Beta	110210	1.00
	BICYCLE	Small red bicycle	10111	255.99
	CHI	Chi	110210	1.00

Contact Legal Privacy Policy Copyright © 2011 CA Technologies.

You can also expand the Script Selection DDL:



CA 2E

** Select ** | CHANGE | 1/03/12 | 14:44:04

** Select **
Auto-confirm prompt
Exit to main menu
Quick exit
Go to RHCAEFR
Test 1

Record keystrokes
Delete script
Edit script

Edit Product

Product code	Product description	Date	Price
ALPHA	Alphaxx	110210	1.00
BETA	Beta	110210	1.00
BICYCLE	Small red bicycle	10111	255.99
CHI	Chi	110210	1.00

Enter | Page Up | Page Down | Exit | Add/Change

Contact Legal Privacy Policy Copyright © 2011 CA Technologies.

Note: In this example, you have *ALL authority. As a result, the Record Keystrokes, Delete Script, and Edit Script options are shown at the bottom of the Script Selection DDL.

Control Options

There are some control options that can be displayed in the UDM User Interface when a list of scripts is built as follows:

Record Keystrokes

Starts the keystroke recording process. This option is only available if keystrokes are not currently being recorded.

Save Script

Lets the user specify a name and textual description for the script that will be created from the recorded keystrokes. This option is only available if keystrokes are currently being recorded or if recording has been paused.

Pause Recording

Pauses the keystroke recording process. This option is only available if keystrokes are currently being recorded.

Resume Recording

Resumes the keystroke recording process. This option is only available if keystroke recording is currently paused.

Cancel Recording

Cancels the keystroke recording and deletes any keystrokes already recorded. This option is only available if keystrokes are currently being recorded or if recording has been paused.

Delete Script

Lets the user select a script to be deleted. This option is only available if keystrokes are not currently being recorded.

Edit Script

Lets the user select a script to be edited. This option is only available if keystrokes are not currently being recorded.

The control options are only displayed in the Script Selection DDL if the user has *ALL authority, as defined in their member in YSCRIPT.

Selecting either Save Script, Delete Script, or Edit Script prompts the user for a name for the script (and, in the case of Save Script, the textual description of the script). The user is prompted for the script name using a popup window.

Control Data

Control data is used to delineate separate scripts and define the attributes for each specific script within a member in YSCRIPT (either the DEFAULT member or a user's member). Control data begins with '%%' (double-percent-sign) followed by a data-identifier, as follows:

%%NAME:script-name

Indicates the name of the script. Each script name must be unique within a member. Each script name must be a valid system name and must not contain any underscores.

%%TEXT:script-description

Indicates the textual description of the script.

%%SCREEN:screen-identifier/*GRPn/*ALL

Indicates the screen for which the script is valid. Screens are identified by their screen identifiers. You can use the special value *ALL to indicate the script is screen-independent. You can use the special value *GRPn to indicate the script is valid for multiple screens. If you are using *GRPn, you must include the related GROUP definition at the top of the member.

%%SCPRCV:script-recovery-option

Indicates a script-specific script recovery option is used to override the global script recovery option held in the YSCPRCV Web Option control value.

%%SCPERR:script-error-page

Indicates a script-specific error page is used to override the global script error page held in the YSCPERR Web Option control value.

In addition to these script-specific control data, you can specify *global* control data-identifiers at the top of the member, as follows:

%%USER:user-name

Indicates the name of the user for whom the member was created.

%%AUTH:*ALL/*USE

Indicates whether the user has authority to create, delete or edit scripts. If set to *USE, the drop-down list of available scripts does not include any control options. If set to *ALL, the drop-down list of available scripts includes control options in addition to the available scripts. For more information about available control options, see Control Options.

%%GROUP:group-identifier list-of-screen-identifiers

Indicates a *group* of screens for which certain scripts are available. By grouping screens, individual scripts can easily refer to the group-identifier in their %%SCRIPT: control datum. Multiple %%GROUP: control-data-identifiers can be present. If an individual script refers to a group, the script is available on any screen within that group.

In the following example, you can see the heading data defining two screen-groups; Group1 contains 6 screens and Group2 contains 3 screens. There are two separate scripts, the second script uses *GRP1 as its screen group (indicating that it is available on any of the 6 screens that make up Group 1):

```
%%================================================================
%%USER:Mickey Mouse
%%AUTH:*ALL
%%SCPRCV:*RECOVER
%%SCPERR:YSCPERR
%%GROUP:1 C2E0005132,C2E0005133,C2E0005134
%%GROUP:1 C2E0005135,C2E0005136,C2E0005137
%%GROUP:2 C2E0001000,C2E0001024,C2E0001995
```

```
%%================================================================
%%NAME:UUAGEFR
%%TEXT:Go to UUAGEFR
%%SCREEN:C2E0005134
:LOOP
SEND 03
IF *SID C2E000006
  IF R01C02 MAIN
    SET-FIELD _F1607U 1
    SEND ENTER
    SET-FIELD _F0181NZ 1
    SEND ENTER
    QUIT
  END-IF
  SEND 03
  GOTO LOOP
END-IF
%%================================================================
%%NAME:DLTTOPRCD
%%TEXT:Delete first subfile record
%%SCREEN:*GRP1
SET-FIELD _F0722U 4
SEND ENTER
SET-FIELD _F1913U Y
SEND ENTER
```

For more information about the control options that can be displayed when a list of scripts is built, see Control Options.

Note: If you are changing an existing user's authority (using %%USER and %%AUTH) specified in the user's member in YSCRIPT, you must end and restart the web option server for the changes to take effect.

Record a Script

You can record your keystrokes by selecting the Record Keystrokes option. This causes the Web Option server to start the keystroke recording module and refresh the screen. In addition, the Script Selection DDL displays **RECORDING** as its top (selected) option. You can then enter data and use command keys as usual. After you finish recording the keystrokes, select the Save Script option to save the recorded keystrokes as a script. You are prompted for a script name and a textual description.

The script is written to your own member in YSCRIPT. By default, all recorded scripts have the following attributes:

1. The %%SCREEN: control-data-identifier is automatically set to the screen on which the Record Keystrokes option was selected, irrespective of the screen on which the user has selected the Save Script option.
2. The %%SCPRCV: and %%SCPERR: control-data-identifiers are set to the values of the current YSCPRCV and YSCPERR Web Option control values.

At any time while recording keystrokes, you can use the Pause Recording and Resume Recording functionality to pause and resume recording. However, you must ensure if you pause keystroke recording, you can only restart recording on the same screen from which you selected the Pause Recording option, otherwise undetermined results can occur.

Note: To support recording that involves navigating away from the initial screen on which the recording was started, ensure the XSCPLST tag exists in the initial and the final screen.

Edit a Script

When you select the Edit Script option, you are prompted for a script name that exists in your member in YSCRIPT). The YSCPEDT shipped skeleton is then presented, showing current statements for the specified script. You can edit, delete, or add new statements. Any change results in the script being updated.

Run a Script

To run a script from the UDM drop-down list, select the appropriate script from the list and confirm. The script runs until completion.

Backwards Compatibility

No automatic processing is available to convert the existing scripts of a customer (each in their own member in YSCRIPT) into separate scripts in the DEFAULT member. However, existing script processing continues to work as before. A conversion program may be written in the future to perform this task.

Limitations

Scripts that are created by recording keystrokes can only include the SET-FIELD and SEND script statements. If more complex functionality is required in a script, you must use the Edit Script option to edit the script manually.

For more information about recording the keystrokes, see Recording a Script and for more information about the Edit Script option to edit the script manually, see Editing a Script.

Appendix F: Generating HTML for Non-CA 2E Display Files

This chapter describes the way Web Option Markup Language Skeletons can be generated for DDS that is not generated from CA 2E.

This functionality has been included with Web Option for users with application suites that contain both CA 2E display files and also non-CA 2E display files to generate Markup Language Skeletons (MLS) for all the display files. An MLS generated for a non-CA 2E display file uses the same Web Option control values and has a similar 'look-and-feel' to an MLS generated from a CA 2E display file.

The Web Option HTML generator uses the same processing 'rules' as the CA 2E display file DDS generator uses to generate the record format and field names for the display file. Thus, when processing a CA 2E display file, the HTML generator can establish the names and relationships between record formats before it starts to process the DDS.

However, hand-written display file DDS (or DDS created by using IBM's Screen Design Aid (SDA) can use any naming scheme (or lack thereof) it chooses for both record formats and fields. Additionally, whereas CA 2E-generated DDS has all fields explicitly defined (in terms of field length and type) and positioned on the screen by row/column, hand-written DDS may have fields referenced to external fields in a field reference file (FRF) and positioned relative to other fields. Because of this lack of formalized rules across hand-written DDS, there is no easy way for the HTML generator to determine the relationships between record formats within a display file.

To provide the Web Option HTML generator with sufficient information to generate a skeleton (or skeletons) from a non-CA 2E display file, a one-time manual process is involved prior to first-time generation, where certain specially formatted comment lines, known as instructions, are added to the display file DDS. These instructions are used by the HTML generator to determine what skeletons to generate for a given display file and which record formats each skeleton should contain. Once these instructions have been added to the display file DDS, they normally do not need to be changed, unless there is a requirement to generate differently named skeletons or to use different record formats.

This section contains the following topics:

Web Option Instructions

Web Option instructions are specially formatted comment lines that are inserted into the DDS source for a non-CA 2E display file. When the Web Option HTML generator processes the file, it uses the information in these comment lines to determine how many skeletons to generate for the display file and which record formats are used for each skeleton.

All Web Option instructions must be added into the source before any data lines. As soon as a DDS data record is found in the source (that is, a record without an asterisk in column 7), record format analysis stops and MLS generation starts, using any instruction lines already processed.

Within the 80-character SRCDTA field in QDDSSRC, Web Option instruction lines all take the following format:

Start Column	End Column	Data	Description
1	6	Comment	Any user-defined comment
7	11	*#W2E	Web Option instruction control code
12	80	Data	Instruction specific data

Thus every Web Option instruction can be identified by *#W2E in columns 7 through 11. However, there are three different types of Web Option instruction line, each of which is differentiated by the two characters in columns 12 and 13, as follows:

- Definition instruction (#D)
- Format instruction (#F)
- Identification instruction (#I)

Definition Instructions

Definition instructions are identified by having #D in columns 12 and 13. They specify the name and description of the skeleton to be generated and also a relative skeleton number within the display file. A definition instruction must precede any format instructions using the same skeleton number. The skeleton name must begin with a letter from A through W.

Instruction-specific data is as follows:

Start Column	End Column	Data	Description
12	13	#D	Web Option instruction type code

Start Column	End Column	Data	Description
14	16	Number	Display file skeleton number
18	27	Name	Skeleton name
32	61	Text	Skeleton description

Example

```
A*. 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7
*#W2E#D001 ORDINQKEY Order inquiry key screen
```

The above line defines to the HTML generator that a new skeleton should be generated called ORDINQKEY, with a description of 'Order inquiry key screen'. For the skeleton to actually be generated, the Definition instruction must be followed by one or more record instructions, which have the same skeleton number.

Format Instructions

Format instructions are identified by having #F in columns 12 and 13. One or more format instructions must immediately follow a definition instruction and are related to it by having the same skeleton number. A format instruction defines the name and description of a record format within the display file that should be included within the skeleton, which is named within the preceding definition instruction. If more than one format instruction follows a definition instruction, the relevant record formats are included in the skeleton in the order the format instructions appear.

Format instructions can include 2 Y/N 'flags', as follows:

- Command key flag
- This Y/N flag is held in column 29 and defines whether the record format contains command key text. This information is used by the HTML generator to build the list of command key buttons and their text.
- Controlling format flag

  This Y/N flag is held in column 30 and defines whether the record format is the controlling format; whether it is the write/read record format that contains the screen identification text that is used at runtime to identify the screen. Only one record format within a skeleton can be defined as the controlling format.

Instruction-specific data is as follows:

Start Column	End Column	Data	Description
12	13	#F	Web Option instruction type code

Start Column	End Column	Data	Description
14	16	Number	Display file skeleton number
18	27	Name	Record format name
29	29	Y/N flag	Command key flag
30	30	Y/N flag	Controlling format flag
32	61	Text	Record format description

Examples

```
A*. 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7
*#W2E#F001 RCDFMT1    YY Order inquiry key screen
```

The above line defines to the HTML generator that the skeleton defined in a preceding definition instruction should include the RCDFMT1 record format. It also defines the RCDFMT1 record format to the HTML generator as containing command key text and as being a controlling format.

```
A*. 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7
*#W2E#F002 SFLCTL1    NY Subfile control
*#W2E#F002 SFLRCD1    NN Subfile record
*#W2E#F002 CMDKEYS    YN Command keys
```

The above lines define to the HTML generator that the skeleton defined in a preceding definition instruction should include the SFLCTL1, SFLRCD1 and CMDKEYS record formats from the display file. Moreover, the SFLCTL1 record format is the format that contains the screen identifying text and the CMDKEYS record format contains command key text.

Identification Instructions

Identification instructions are identified by having #I in columns 12 and 13. A single identification must immediately follow the format instruction, which is defined as the controlling format (that is, the format instruction that has 'Y' in column 29). An identification instruction defines the text on the controlling format that should be used to identify that format.

Instruction-specific data is as follows:

Start Column	End Column	Data	Description
12	13	#I	Web Option instruction type code

Start Column	End Column	Data	Description
14	16	Number	Display file skeleton number
18	21	Name	Offset to identifying text
32	61	Text	Identifying text

Example

```
A*. 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7
*#W2E#I001 0031          Work with orders
```

The above line defines to the HTML generator that the record format defined in a preceding format instruction as the controlling format can be uniquely identified by having the text 'Work with orders' at offset 31 (row 1, column 31). Note that the offset is defined as a 4-digit number including leading zeroes.

Extended example

```
A*. 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7
*#W2E#D001 CSTINQKEY     Customer inquiry key screen
*#W2E#F001 RCDFMT1     YY Key screen
*#W2E#I001 0031          Select customer
*
*#W2E#D002 CSTINQDTL1    Customer details screen
*#W2E#F002 SFLCTL1    NY Subfile control
*#W2E#I002 0032          Customer details
*#W2E#F002 SFLRCD1    NN Subfile record
*#W2E#F002 CMDKEYS    YN Command keys
*
*#W2E#D003 CSTINQDTL2    Customer address screen
*#W2E#F003 SFLCTL2    NY Subfile control
*#W2E#I003 0102          Customer address
*#W2E#F003 SFLRCD2    NN Subfile record
*#W2E#F003 CMDKEYS    YN Command keys
*
```

The above lines define to the HTML generator to generate the following skeletons:

1. A skeleton called CSTINQKEY, which contains a single record format (RCDFMT1). The generator creates a screen cross-reference record for the skeleton showing that the skeleton is uniquely identified by the text 'Select customer' at offset 31 and that it contains command key text on it, which should be parsed to determine the command key button names to generate.
2. A skeleton called CSTINQDTL1, which contains three record formats:
   - SFLCTL1 - the controlling record format and can uniquely be identified at runtime by the text 'Customer details' at offset 32
   - SFLRCD1
   - CMDKEYS - record format parsed by the HTML generator for command key text to place on the command key buttons
3. A skeleton called CSTINQDTL2, which contains three record formats
   - SFLCTL2 - the controlling record format and can uniquely be identified at runtime by the text 'Customer address' at offset 102
   - SFLRCD1
   - CMDKEYS - record format is parsed by the HTML generator for command key text to place on the command key buttons

Note that more than one skeleton can include the same record format (in the example above, the CMDKEYS record format is used in both the CSTINQDLT1 and CSTINQDTL2 skeletons).

Appendix G: Web Option Limitations

A limitation exists with Web Option where it can display a Just-In-Time (JIT) page, even if a skeleton was previously generated for the screen. This limitation only occurs when:

- The function is a window function.
- The function has a confirm prompt (that normally appears in its own small window).
- The function displays with the confirm prompt, but was not displayed immediately before and without the confirm prompt.

For example, if a window function WinFun (with a confirm prompt) displays, and then, as a result of action on the WinFun page, a different function SelFun is displayed. In this case, when the user leaves SelFun, WinFun is immediately redisplayed with the confirm prompt, and the WinFun page displays in JIT mode.

This limitation occurs because although Web Option is able to determine that a confirm prompt (in its own window) is being displayed, it is unable to retrieve required details of the underlying window, such as its original window offset and attributes, which are required to process field offsets.

This limitation does not effect full-screen functions. It does not effect window functions that initially display without the confirm prompt and then redisplay with the confirm prompt when the user presses Enter.

There are two workarounds to this limitation:

- Ensure that window functions that have the option to show the confirm prompts always display without the confirm prompt. Require a command key such as Enter to be pressed before the confirm prompt displays. This is essential in situations such as the previous example, where control might have gone to a different function and can then return to the window function.
- When window functions do not have the option to display the confirm prompt, consider changing the window function to be a full-screen function.

Note: If a window function displays in JIT mode, this is only while the confirm prompt is displayed.When the window function displays without the confirm prompt, the generated skeleton is used. If the window function displays in JIT mode, normal application functionality is retained. However, functions specific to the generated skeleton, such as drop-down lists, are not available.

Index