CA 2E

Release Notes

r8.5

Contact CA

Contact Technical Support

For your convenience, CA provides one site where you can access the information you need for your Home Office, Small Business, and Enterprise CA 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

Provide Feedback

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

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

Contents

Chapter 1: New Features and Enhancements 7

Service Program Design and Generation . . . 7
- Service Program Overview . . . 8
- Service Program Functions . . . 9
- Edit Function Details Panel . . . 11
- Adding Modules and Procedures . . . 12

Web Service Creation . . . 17
- Installation Requirements . . . 17
- Architecture . . . 22
- Sample Flow . . . 25
- Web Services Limitations . . . 32
- New Commands . . . 32
- Web Service Remote Deployment . . . 36
- References . . . 40

Impact Analysis Enhancements . . . 40
- Include inactive code: *YES . . . 41
- Include inactive code: *NO . . . 41
- Include inactive code: *IGN . . . 41
- Examples . . . 41
- Hanging References . . . 45

Improved Model Selection and Positioning . . . 48

Web Option Environments . . . 51
- Upgrade Existing Web Option Libraries . . . 53
- Web Option Commands . . . 55
- Create Web Option Environment Command (YCRTW2EENV) . . . 56

Additional Enhancements and Updates . . . 58
- Default Target HLL Language for New Models is RPGIV . . . 58
- YCVTCNDVAL and YCVTMDLMSG Enhancement . . . 58
- New Versioning User Exit Program . . . 58
- New Reason Code for DSPMDLUSG and DSPMDLREF . . . 59
- Rebranding . . . 59
- EJB Option No Longer Supported . . . 59
- New Model Value YTRNAPI - Convert case API . . . 64
- IPv6 Compatibility . . . 64
- CL ILE EXCUSRPGM Functions . . . 64
- LDO Modification . . . 65

Chapter 2: Fixes to CA 2E r8.5 67

2E 67
CA 2E Toolkit 74
Web Option 75

Chapter 1: New Features and Enhancements

This chapter describes the new features and enhancements for CA 2E r8.5.

This section contains the following topics:

Service Program Design and Generation (see page 7)
Web Service Creation (see page 17)
Impact Analysis Enhancements (see page 40)
Improved Model Selection and Positioning (see page 48)
Web Option Environments (see page 51)
Additional Enhancements and Updates (see page 58)

Service Program Design and Generation

In the CA 2E model, a new function type called Service Program (*SRVPGM) is defined. Service Programs can be copied, deleted, renamed, and so on, just like any other function. Service Programs have no parameters or action diagram, and they can be defined over any file. A Service Program has an associated source member, whose name is automatically allocated, but which can be changed. The source member name will be used for the final *SRVPGM object name.

Once a Service Program is created, a developer can, through a series of screens, define which existing module functions should be bound into the Service Program. Additionally, they can define one or more external (non-2E) *MODULE objects to be bound into the Service Program. For each bound module, the developer can specify which exported procedures from the module should also be exported from the Service Program.

When the Service Program is generated, a source member is created which includes both the export list, as well as several compile preprocessor directives which define how the *SRVPGM object should be created.

Service Program Overview

Within the CA 2E product, external functions can be defined with an object attribute of either PGM or a MOD. When compiled, these functions become one of two i OS object types-respectively, a *PGM (executable program) or a *MODULE (non-executable module).

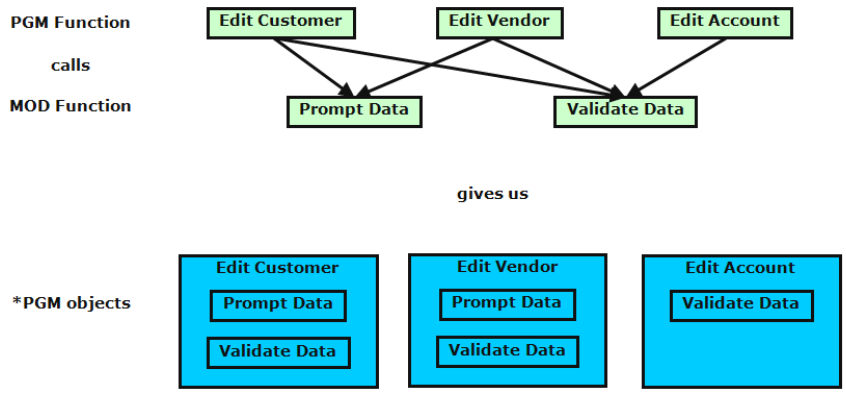
A *PGM object can be directly called, either from a command-line or from another program (assuming the correct parameters are passed). By contrast, a *MODULE object must first be 'bound' into an executable program.

Earlier releases of CA 2E provided the following ILE-specific functionality:

- An external function could be given an attribute of MOD (instead of PGM). This function would compile into a i OS *MODULE rather than a *PGM.
- All MOD functions are automatically added to the 2E model's default binding directory YBNDDIR during the compilation process.
- The default compilation command for ILE programs forces the compiler to search YBNDDIR for called bound modules.
- Calls to a MOD function would be generated (in RPG) as a CALLB (call-bound) rather than a CALL.

Note: Developers could then define certain external functions (typically ones that would never be called from a command line, such as SELRCD's) as modules. Any programs that called these functions (EDTFIL's, etc.) would include a copy of the module object. In ILE terms, this is known as *bind-by-copy*.

For example, the following diagram shows the use of three PGM functions calling two MOD functions and the resulting *PGM objects containing copies of the *MODULE objects:



This has a number of benefits:

- Fewer *PGM objects required at runtime can simplify application deployment.
- Calls to bound modules are typically quicker than calls to other programs, due to significantly reduced object pointer resolution overhead.

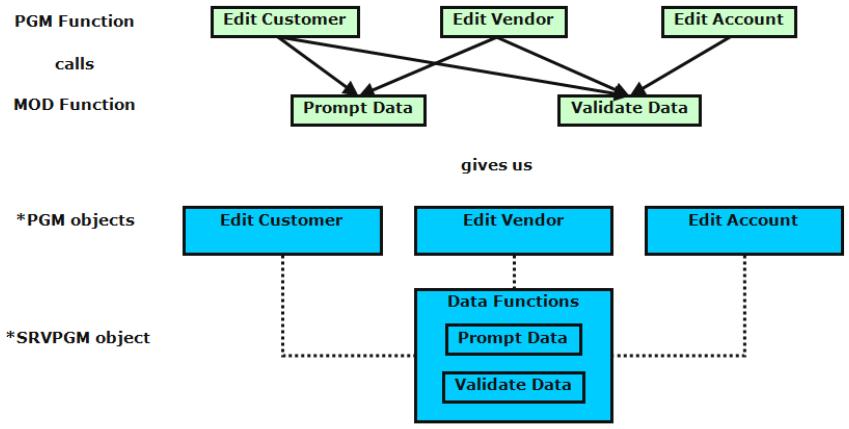
However, if a module is bound into a program and the code within the module function needs to be changed, the *MODULE must be recompiled and then any programs that bind that module must be changed to include the new copy of the module-typically this would be done by regenerating/recompiled the PGM function in a 2E model, although the i OS UPDPGM command can be run from outside the model environment.

Note: Although there are fewer *PGM objects, they are larger than if they did not contain a copy of the module.

Service Program Functions

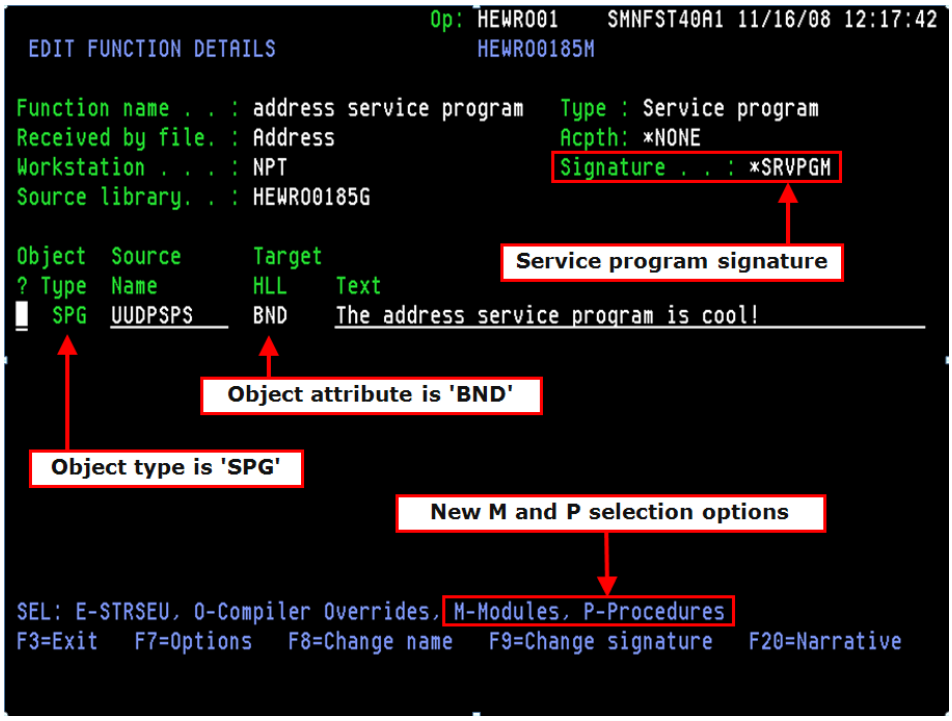
The new Service Program Support allows developers to create service program functions, which are functions that equate to i OS *SRVPGM objects and can be thought of as *containers* for one or more *MODULE functions. As with modules, you cannot directly call service programs. However, when a program function is compiled into a *PGM object, if it finds the service program (which contains the module) in the binding directory before it finds the module itself, it performs a *bind-by-reference*, wherein it simply contains a reference (link) to the module object in the service program and does not itself contain a copy of the module object. At runtime, the program resolves (calls) to the copy of the module contained in the service program (so the service program needs to be available).

Using service programs in this manner provides the benefit of improved performance, due to one-time object resolution, but ensures that a change in any of the modules contained in the service program does not require a change in the calling program, as detailed in the following diagram.



Edit Function Details Panel

This existing panel displays when you take option *Z* (Details) against a function. For a Service Program, this screen contains certain Service Program-specific fields, subfile options, and function keys, as you can see in the following example:



The *Signature* field displays the current signature for the Service Program. It can take the following values:

- *SRVPGM - The signature is the *SRVPGM object name
- *GEN - The signature is system-generated
- User-defined - the signature is a user-specified 16-byte string

Note: The signature is determined at generation time (or in the case of *GEN, at compile-time).

You can take option M (Modules), to display the list of modules currently bound into the Service Program (and to add more modules if you authority).

You can also take option *P* (Procedures) against the Service Program, to display the list of procedures which will be exported from the *SRVPGM object (and, if you have sufficient authority, to reorder that list).

Use function key F9 to make the Signature field input-capable

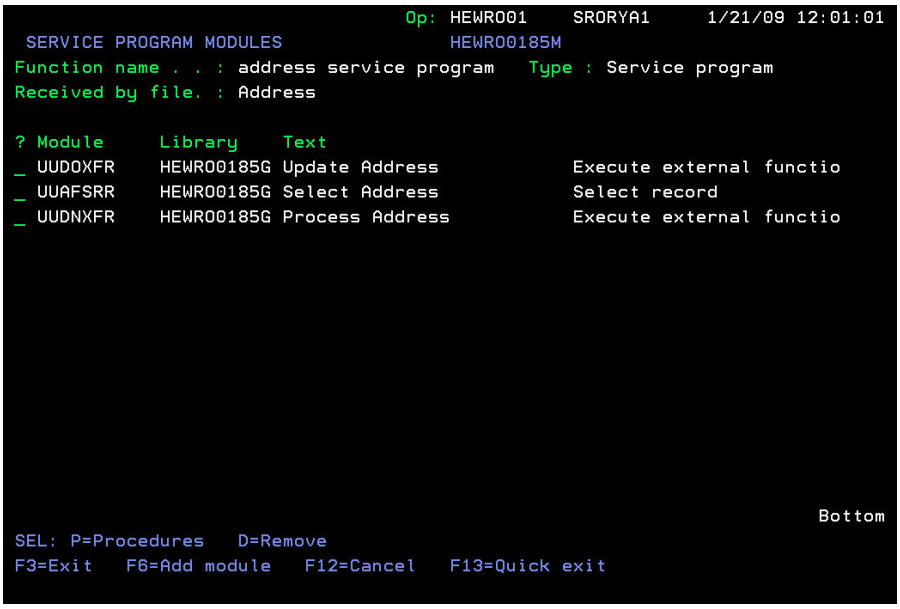
There is one function option for a Service Program function - *Add to Binding Directory*. This determines whether or not this service program should automatically be added to the model's default binding directory. Whether set to Y or N, all constituent modules will be removed from the default binding directory when the service program is created.

Setting this to N can be useful where you create multiple service programs that contain one or more of the same modules, which therefore export the same procedures, since this would give rise to errors when subsequently compiling programs.

Adding Modules and Procedures

The Service Program Modules panel is called when option M is taken from the EDIT FUNCTION DETAILS panel (above).

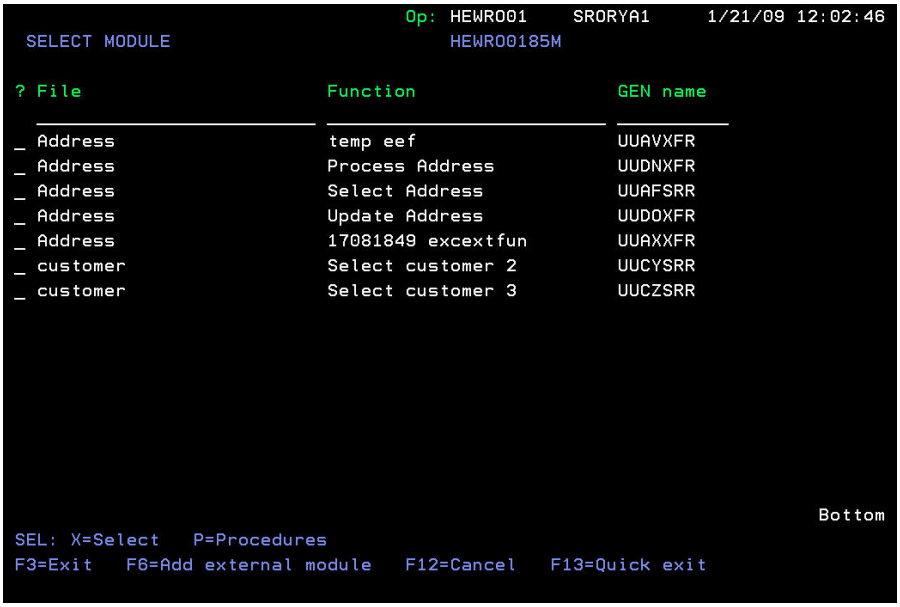
Service Program Modules



From this screen, you can see which modules are currently bound into the Service Program. This list may include both 2E modules and external (non-2E) modules.

You can also press F6 to go to the SELECT MODULE screen

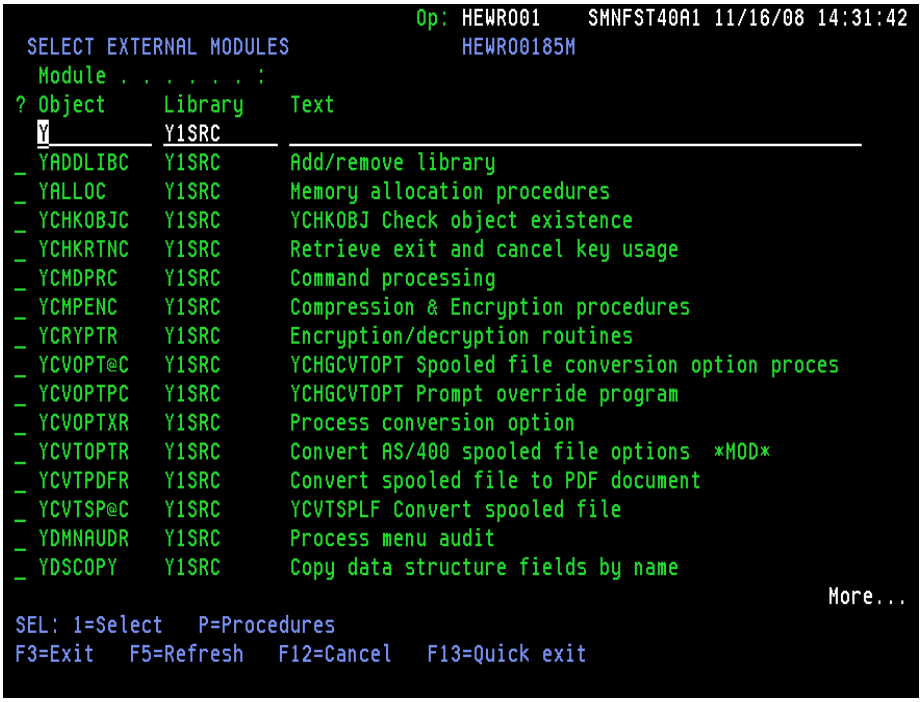
Select Module



You can select one or more modules to bind into the Service Program, by using subfile option X (displayed) or 1 (not displayed, but active anyway). This automatically makes all the procedures from the module exported from the Service Program. Alternatively, you can take option P against a module and select which procedures they would like to export from the Service Program.

You can also press F6 to display the SELECT EXTERNAL MODULE screen

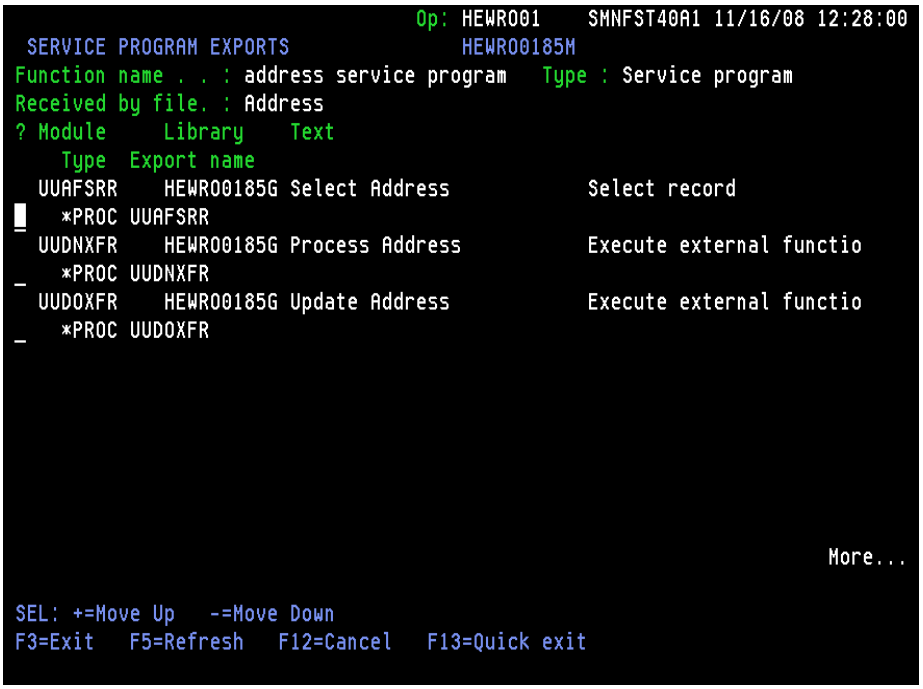
Select External Module



You can select one or more external (non-2E) modules to bind into the Service Program. The initial screen is displayed empty, and you can specify subfile control criteria so sub-select the modules to display. As with the SELECT MODULE screen, option P is available to sub-select procedures from within the module to export from the Service Program.

Service Program Exports

The Service Program Exports panel is displayed when option P is taken from the EDIT FUNCTION DETAILS panel:



This allows you to display the list of procedures which are exported from the *SRVPGM object, and if necessary, re-order them.

Note: Although this is only a requirement where, due to modules being removed and added, the list may have changed-typically, once a Service Program has been generated and compiled into a *SRVPGM object, this list would never be changed).

Service Program Generation Mode

When option G or J is taken against a Service Program, the generation program creates/updates the source member (in file QSRVSRC) in the 2E model generation library (*GENLIB). This source member contains both the export list associated with the *SRVPGM object, as well as several compile preprocessor directives which will be used by the CA 2E Toolkit to actually create the *SRVPGM object. These include the following (in order):

1. The actual CRTSRVPGM command, specifying the bound modules and also specifying the same source member as the export list definition member
2. An ADDBNDDIRE command, to add the*SRVPGM object to the YBNDDIR binding directory (Only if function option Add to Binding Directory is set to Y)

3. Multiple RMVBNDDIRE commands (one for each 2E module bound into the Service P), to remove that module from the YBNDDIR binding directory.
4. When you compile a Service Program function using the YSBMMDLCRT command, the 2E model generation/compile processing executes the YEXCOVR command against the source member. This command invokes the Toolkit compile preprocessor, which processes the compile directives in the source member.

An example of an Service Program source member is as follows:

```
/*Y: CRTSRVPGM SRVPGM(HEWRO0185G/UUDPSPS) +                          */
/*Y: MODULE(HEWRO0185G/UUAFSRR HEWRO0185G/UUDOXFR) +                 */
/*Y: EXPORT(*SRCFILE) SRCFILE(HEWRO0185G/QSRVSRC) OPTION(*DUPVAR +   */
/*Y: *DUPPROC *NOWARN) BNDDIR(QC2LE YBNDDIR) TEXT('The address +     */
/*Y: service program is cool!')                                      */

/*Y: ADDBNDDIRE BNDDIR(HEWRO0185G/YBNDDIR) OBJ((HEWRO0185G/UUDPSPS + */
/*Y:  *SRVPGM)) POSITION(*FIRST)                                     */

/*Y: RMVBNDDIRE BNDDIR(HEWRO0185G/YBNDDIR) OBJ((*LIBL/UUAFSRR +      */
/*Y: *MODULE))                                                       */
/*Y: RMVBNDDIRE BNDDIR(HEWRO0185G/YBNDDIR) OBJ((*LIBL/UUDOXFR +      */
/*Y: *MODULE))                                                       */

            STRPGMEXP  PGMLVL(*CURRENT) SIGNATURE(*GEN)

            EXPORT     SYMBOL('UUAFSRR')
            EXPORT     SYMBOL('UUDOXFR')

            ENDPGMEXP
```

Processing this source member results in the UUDPSPS service program being created in library HEWRO0185G and being added to the top of the YBNDDIR binding directory, with *MODULE objects UUAFSRR and UUDOXFR being removed from the same binding directory.

Note: The shipped defaults for the CRTSRVPGM, ADDBNDDIRE, and RMVBNDDIRE commands can be customized by doing the following:

1. CRTSRVPGM - the defaults for this command are held in a shipped model message, *CRTSRVPGM, with message id Y2U1033:

```
*CRTSRVPGM                EXC   Y2U1033   Y2USRMSG
```

2. ADDBNDDIRE - the defaults for this command are held in message Y2R0130 in Y2MSG.
3. RMVBNDDIRE - the defaults for this command are held in message Y2R0137 in Y2MSG.

Web Service Creation

This feature fulfills a requirement to provide a mechanism to expose CA 2E server-side programs via a web services, and to model this exposure.

The runtime functional deliverable is an automatically created and deployed Web Service(s); its operations invoke 2E server-side ILE service programs.

The feature creates a Web Service containing one-to-multiple operations, where each operation corresponds to a single procedure in a module within a 2E ILE Service Program. Note that CA 2E Service programs can also contain modules developed outside of a 2E model, as described in the earlier sections of these Release Notes.

Approach:

IBM's i 6.1 (and V5R4 with PTFs) provides a Web Services Server.

IBM states, "The Web services server provides a convenient way to externalize existing programs running i OS®, such as RPG and COBOL programs, as Web services."

The IBM Web Administration Interface provides a web-based, wizard like approach to creating and deploying a Web Service that can invoke an RPG ILE or COBOL ILE program.

The CA 2E support reduces the reliance on the Web Administration Interface Web Services wizard, with a programmatic invocation of the IBM shipped scripts which perform the pertinent Web Service administration tasks:

- Install a Web Service (i.e. automatically create and deploy).
- Uninstall a Web Service

Install and uninstall Web Service administration tasks are available as new 2E commands.

Additionally, a CA 2E user can model Web Services within a 2E model enabling 2E facilities such as impact analysis to be applied to web services.

Installation Requirements

To enable Web services provider and requestor support you must have the following products installed:

- Extended Base Directory Support (5761-SS1 Option 3) V5R4, or later.
- IBM HTTP Server for i5/OS (5761-DG1) V5R4, or later.
- IBM Developer Kit for Java™ (5761-JV1 Option 8) V5R4, or later

- Qshell (5761-SS1 Option 30) V5R4, or later
- Portable Appl Solutions Environment (5761-SS1 Option 31) V5R4, or later
- Digital Certificate Manager licensed program (5761-SS1 Option 34) V5R4, or later (needed only for service requestor support)

Note: 5722 is the product code for i5/OS options and products, prior to V6R1.

To see the list of installed products on your machine, you should run the following command:

```
GO LICPGM
```

Then select option 10.

RPG and COBOL source must be is compiled with the PTF's listed below to include the information necessary to generate Web services programs or service programs.

Required IBM PTFs

In order to correctly support CA 2E r8.5 (particularly Web Service Support) you need to ensure that you have the latest IBM Cumulative Package and Latest HTTP Group PTFs installed.

Latest Cumulative Package:

- V5R4
- V6R1

Latest HTTP Group PTF:

- V5R4
- V6R1

Note: There are additional PTFs required. At the time of writing this document, the following PTFs are not yet included in the Latest Cumulative packages and must be ordered separately:

V5R4

5722SS1 - SI34862

V6R1

5761SS1 - SI34865

For more information on PTF's, see IBM's website.

PCML in Module

At V5R4, *after* the PTF's are installed, but *before* the module source is compiled by CA 2E, the following statements *must* exist in the module source. (The PGMINFO statement does not have to be manually added at V6R1 and beyond.)

RPG

H PGMINFO(*PCML:*MODULE)

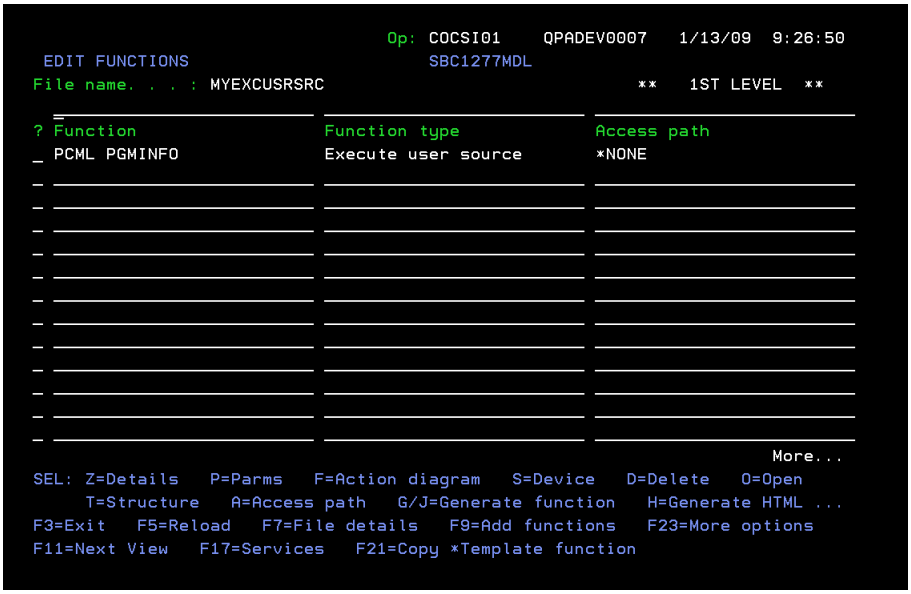
COBOL

PROCESS OPTIONS PGMINFO(PCML MODULE)”

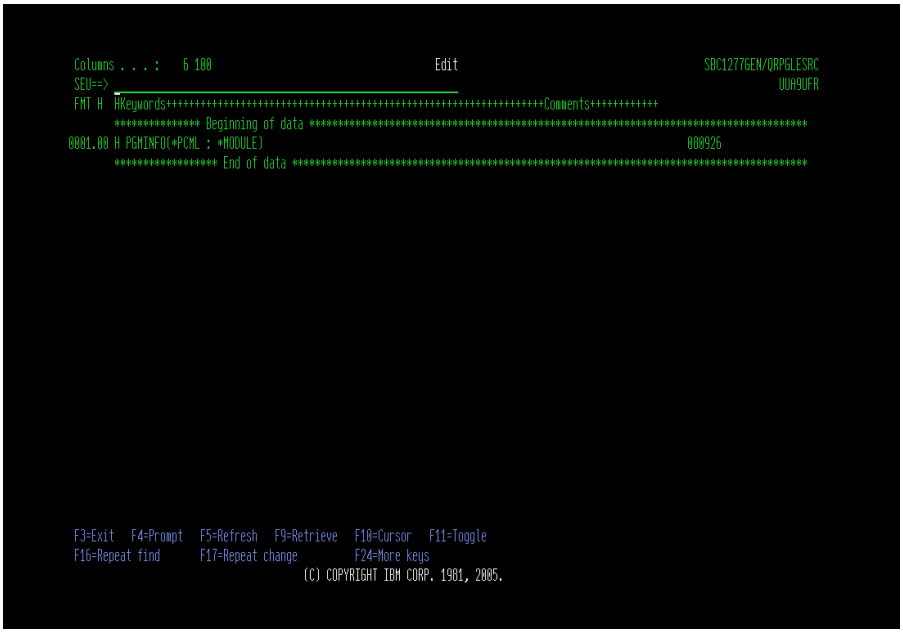
Note: 2E does not automatically generate these PGMINFO source lines.

An easy method to accomplish generation of PGMINFO into V5R4 source is to create an EXCUSRSRC member containing the PGMINFO statement and then include that statement in the AD of the module, as in the following examples:

1. “PCML PGMINFO” is a function of type EXCUSRSRC.

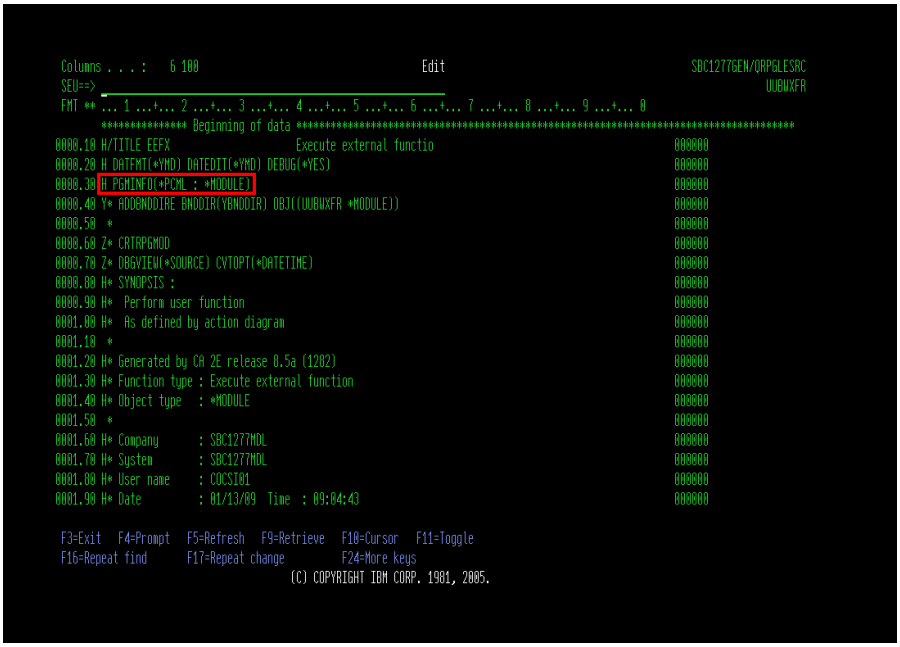


2. Source member for the EXCUSRSRC function contains "H PGMINFO(*PCML : *MODULE)".



3. AD of function EEF shows a call to the user source.

The generated source for the module shows the included PGMINFO line:



Note: At V6R1, it is not necessary to add the PGMINFO statement, but you must ensure that the Compiler Overrides for the Module have the PGMINFO parameter set to PGMINFO(*PCML *MODULE).

In addition to modifying CA 2E functions using EXCUSRSRC to include the keyword PGMINFO(*PCML : *MODULE), the YRP4HS2 model value can be used to alter H spec generation (for an RP4 module) on a model-wide basis.

YRP4HS2 (*MODULE) ships with a value of 'H DATFMT(*YMD) DATEDIT(*YMD) DEBUG(*YES)'.

Changing the value, as below, ensures any generated module is generated with contained PCML:

```
YCHGMDLVAL MDLVAL(YRP4HS2) VALUE('H DATFMT(*YMD) DATEDIT(*YMD) DEBUG(*YES)

PGMINFO(*PCML : *MODULE)')
```

Architecture

The fundamental components of this solution are the new 2E commands that encapsulate programmatic invocation of IBM's Web Service Administration scripts.

IBM ships the following scripts.

Script	Purpose
installWebService.sh	Create and deploy a Web Service
listWebServices.sh	List Web Services deployed to a Web Services Server
startWebService.sh	Start a deployed Web Service
stopWebService.sh	Stop a deployed Web Service
uninstallWebService.sh	Uninstall a deployed Web Service
createWebServicesServer.sh	Create Web Services Server*
deleteWebServicesServer.sh	Delete Web Services Server*
startWebServicesServer.sh	Start Web Services Server*
stopWebServicesServer.sh	Stop Web Services Server*

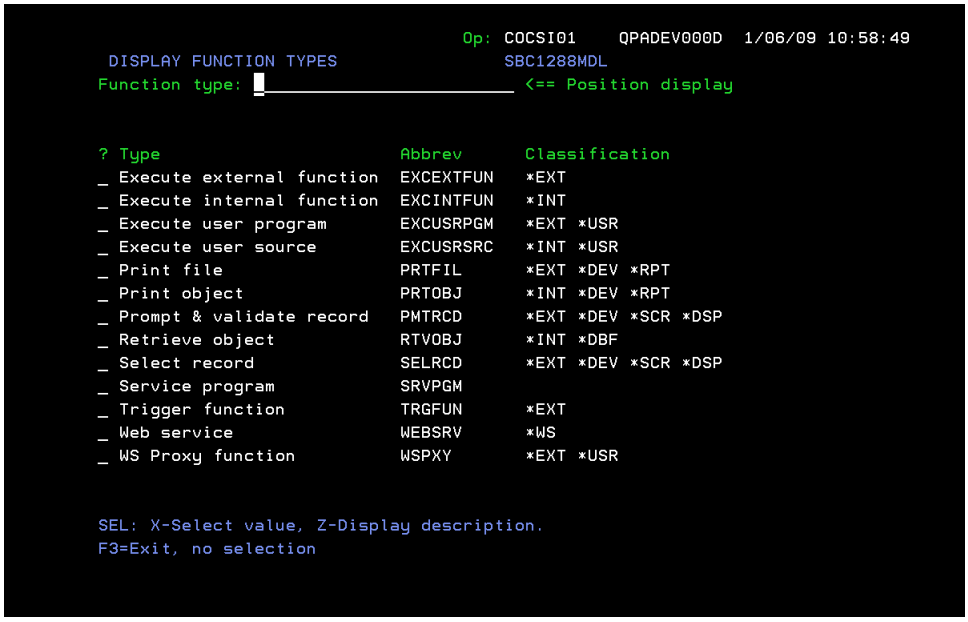
Note: It is *not* the intention of this phase of the CA 2E Web Service support to mimic all functionality available within the Web Administration interface. This release allows a Web Service to be modeled and to be installed to/uninstalled from a Web Services Server.

Two new 2E commands mirror and invoke functionality of two of the scripts:

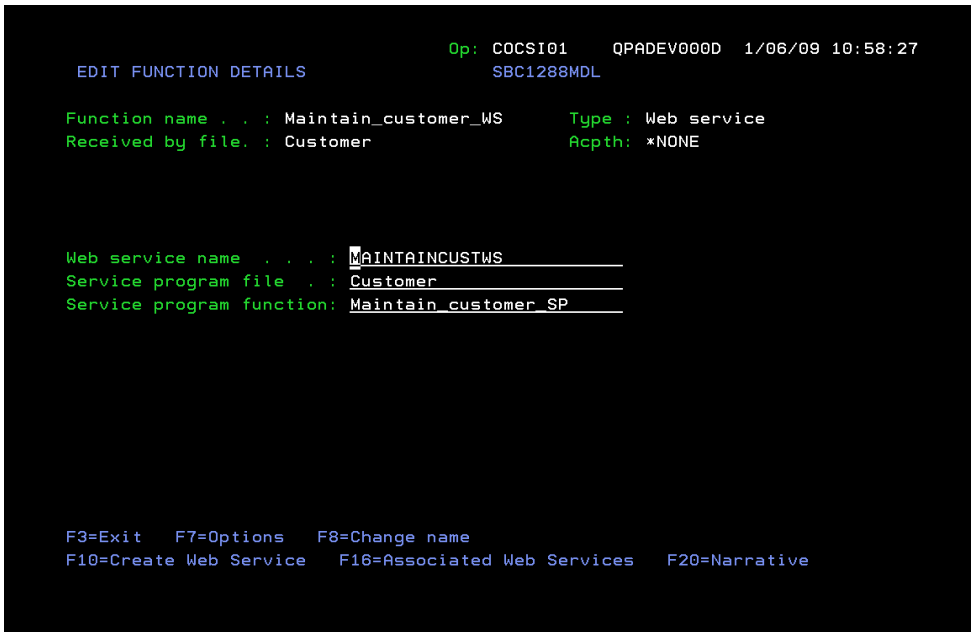
2E command	Purpose
YCRTWS	Create and deploy a Web Service (installWebService.sh)
YUNSWS	Uninstall a deployed Web Service (uninstallWebService.sh)

Note: The Web Service Administration interface (and therefore) scripts cannot operate on a remote environment; i.e. all WS work is for local machine.

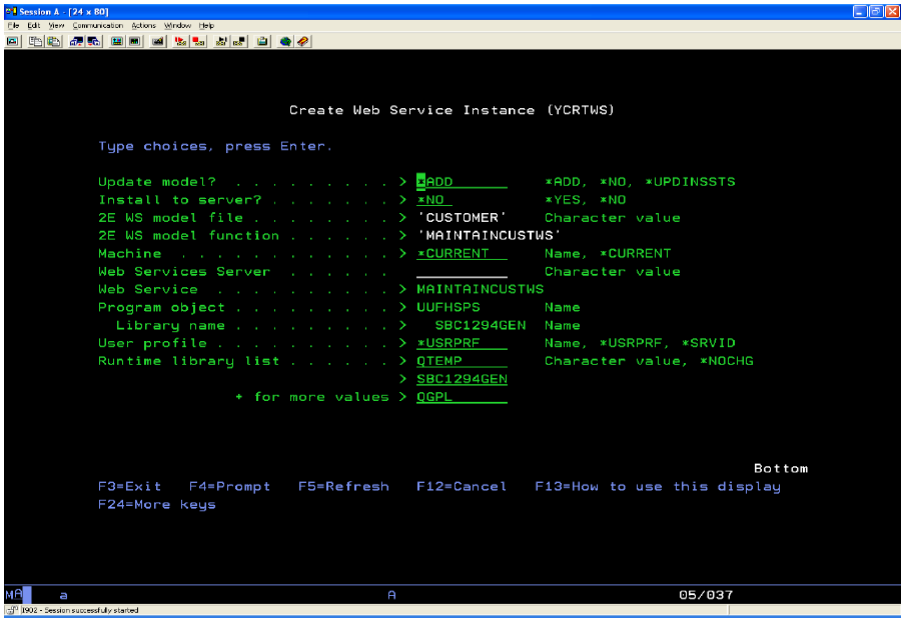
A new 2E function type of Web Service (WEBSRV) is available within the model.



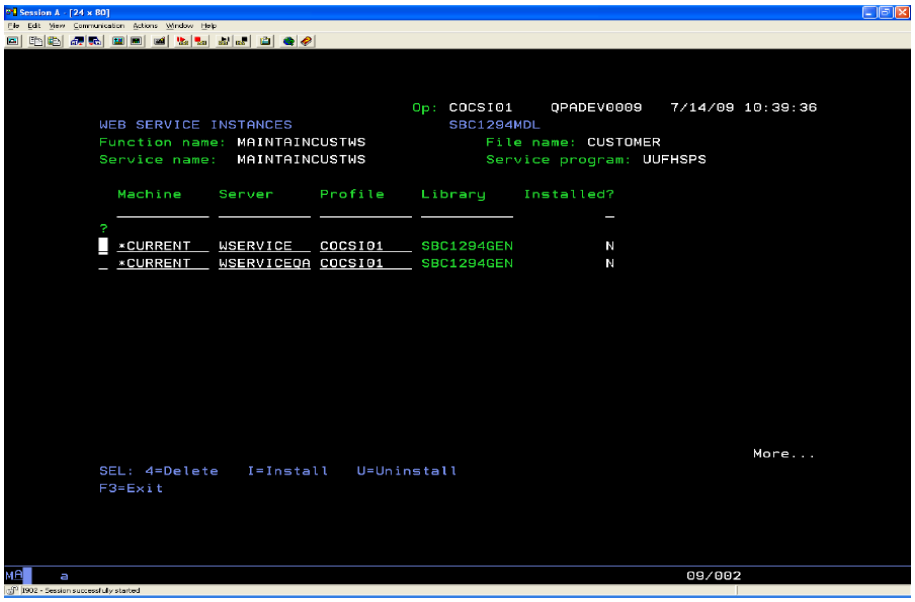
A function of type Web Service cannot be generated and does not have an action diagram. Its purpose is to serve as an umbrella for any number of Web Services Instances. A Web Service instance corresponds to a web service that is or can be deployed to a Web Services Server. The function of type web service has a customized EDIT FUNCTION DETAILS panel. This panel stores the default name of any created web service instance, as well as the 2E file/function that represents the target Service Program to be exposed.



The EDIT FUNCTION DETAILS panel provides option F10 to invoke YCRTWS.



The EDIT FUNCTION DETAILS panel also provides option F16 to invoke WEB SERVICE INSTANCES PANEL:

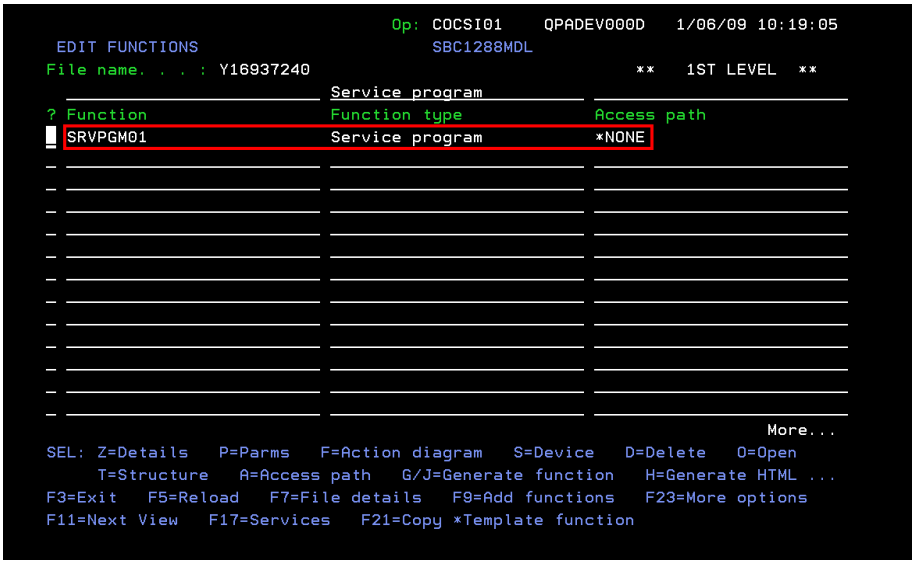


Sample Flow

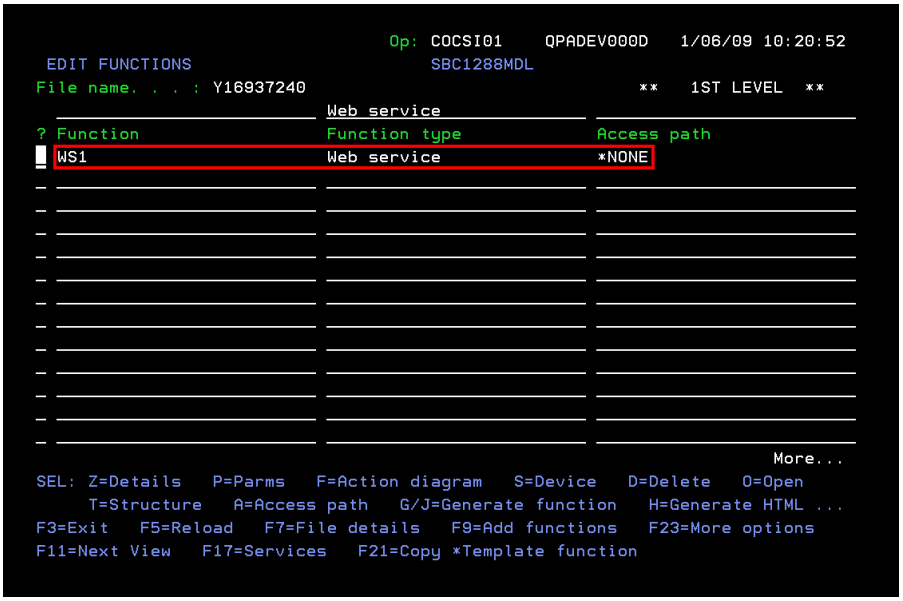
This section shows an example of how you might use these new features in CA 2E. Some settings and selection might vary depending on your system configuration.

To define and deploy a Web Service from CA 2E

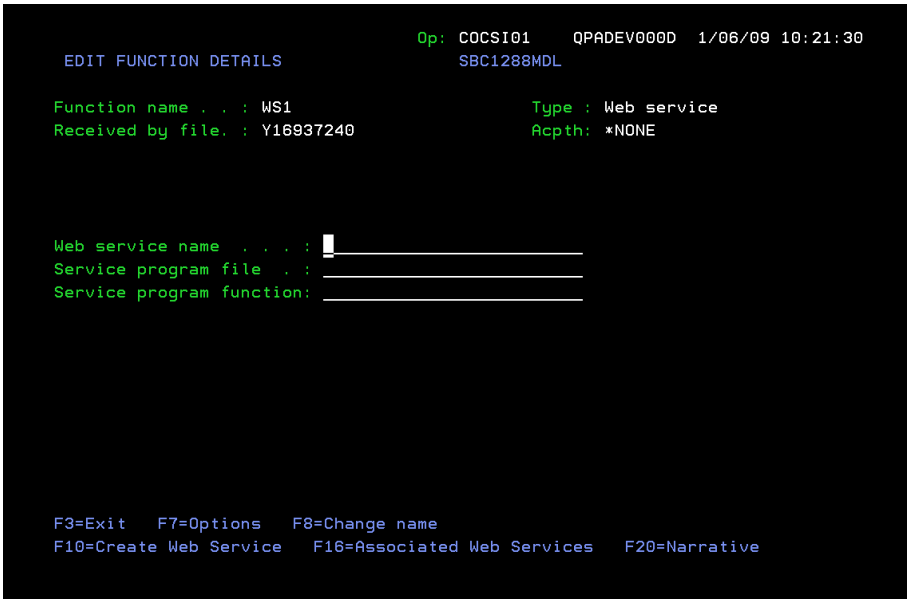
1. Identify a Service Program that has module operations you want to expose via a web service.



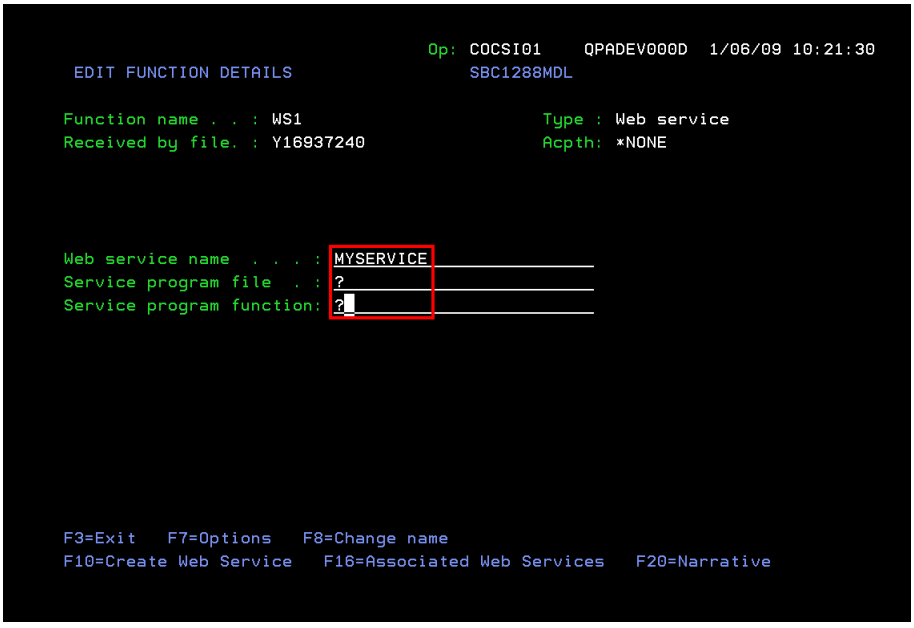
2. Create a new Web Service type function.



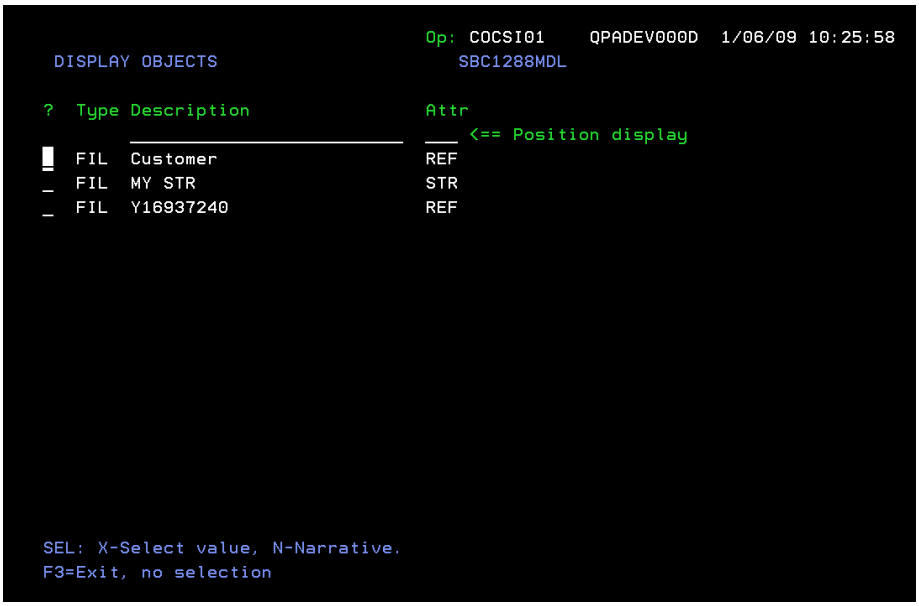
3. Zoom into the Web Service function to Edit Function Details.



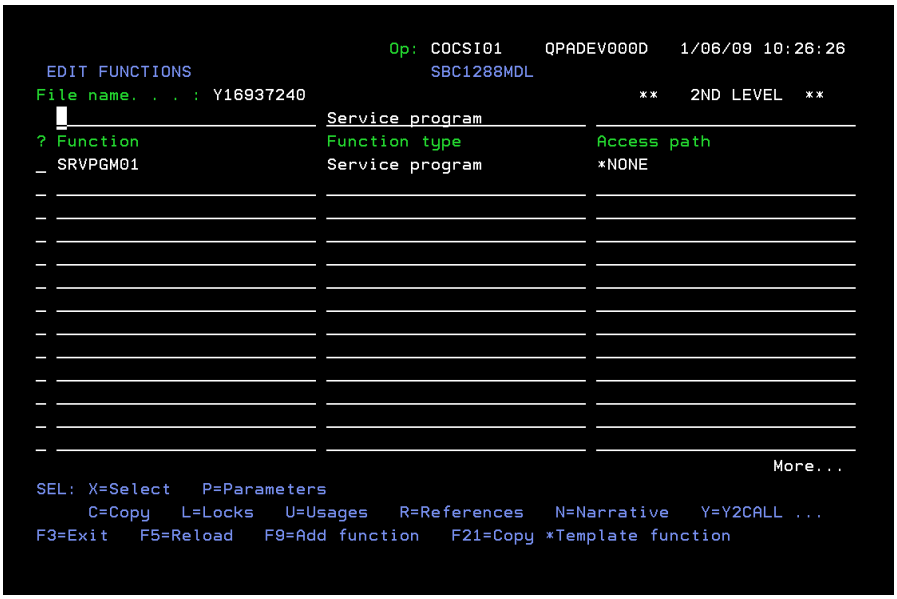
4. Specify the Web Service Name and Select Service Program.



Note: On Service program file and Service Program function allows user to choose file, as in the following example:

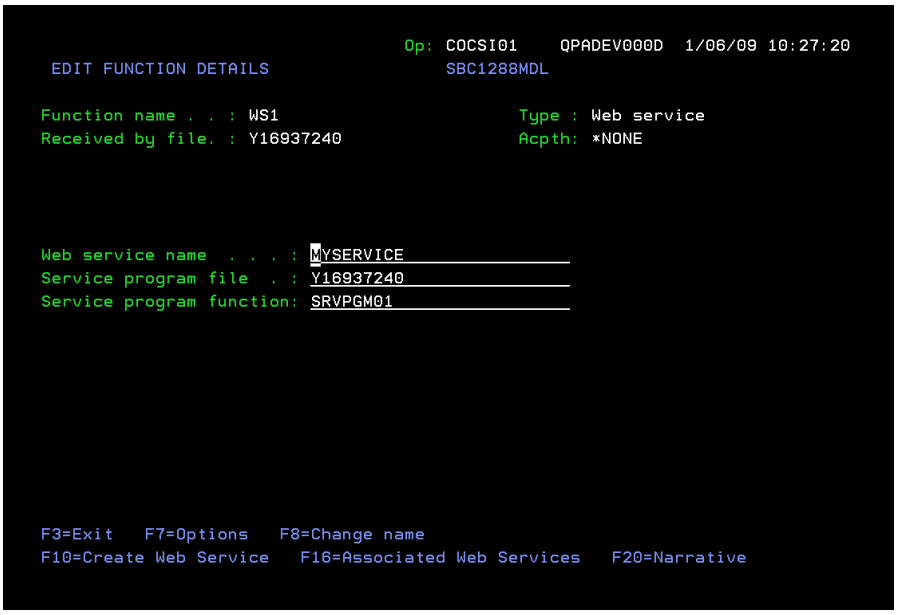


Users can also choose function, as in the following example:



5. EDIT FUNCTION DETAILS

   Auxiliary details populated.



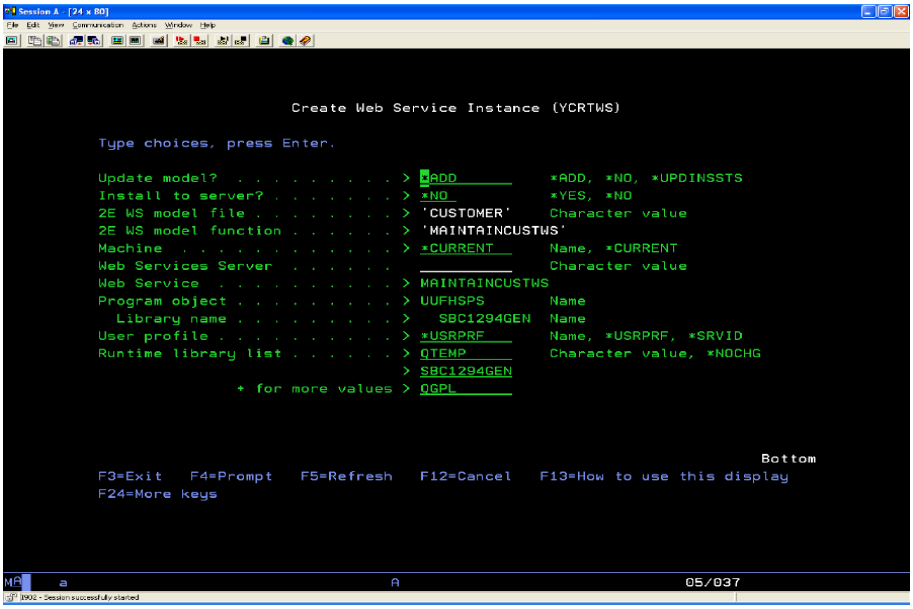
Note: Any number of Web Service Instances can be created from the EDIT FUNCTION DETAILS panel. Each Web Service will be associated with the 2E Web Service function in the panel header.

All the associated Web Services can be viewed using F16 to invoke the WEB SERVICE INSTANCES panel.

To create a Web Service Instance with a different Web Service name, you should change the Web Service Name on Edit Function Details, then press F10.

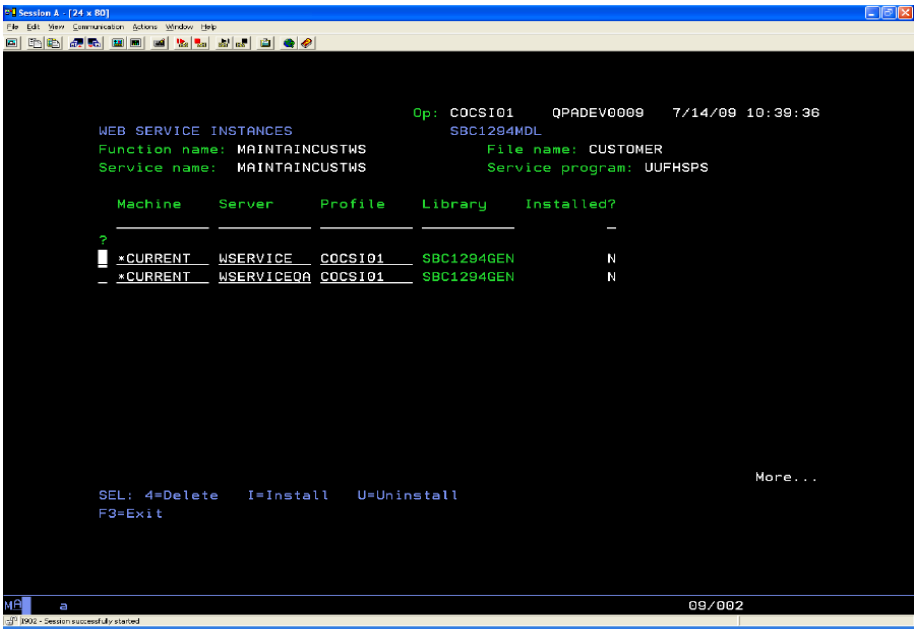
Note: If any of the instances have been installed to a server and the Installed Flag is set to Y, then the Web Service Name, Service Program File and Service Program Function all become Output only. This is to ensure that the Web Service details in the model, and the installed Web Service Instance, are in synch.

6. Create the Web Service Instance (YCRTWS)



Note: See the section *New Commands* for the command parameter descriptions.

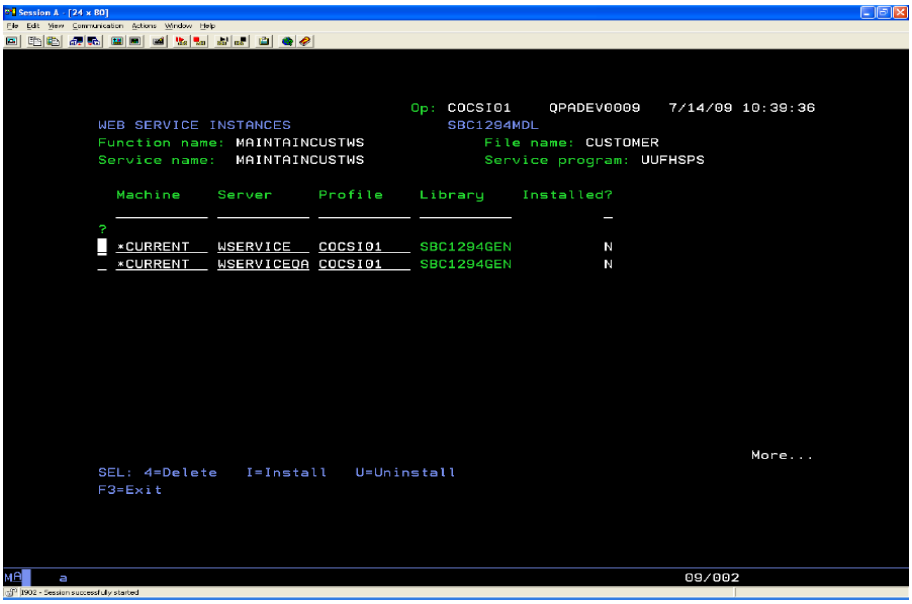
7. Work with Web Service Instances:



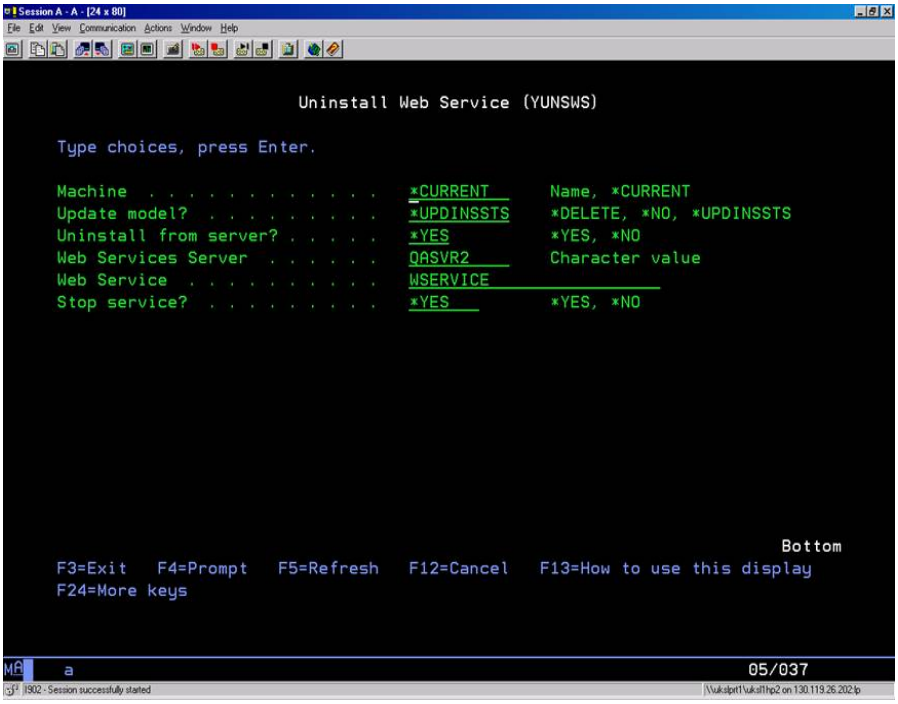
A machine name of *CURRENT indicates that the Web Service instance is to be deployed on the local machine. If you enter any other machine name other than the local one, the Installed flag will always be set to ?, since the machine is not available.

You can only take option I - Install for a Web Service Instance with Machine = *CURRENT. To install Web Service instances to a remote machine, you must use the Web Service Remote Deployment feature. For further details of this please refer to the corresponding section in this chapter.

8. Use option I to install a modeled Web Service to a Web Services Server.

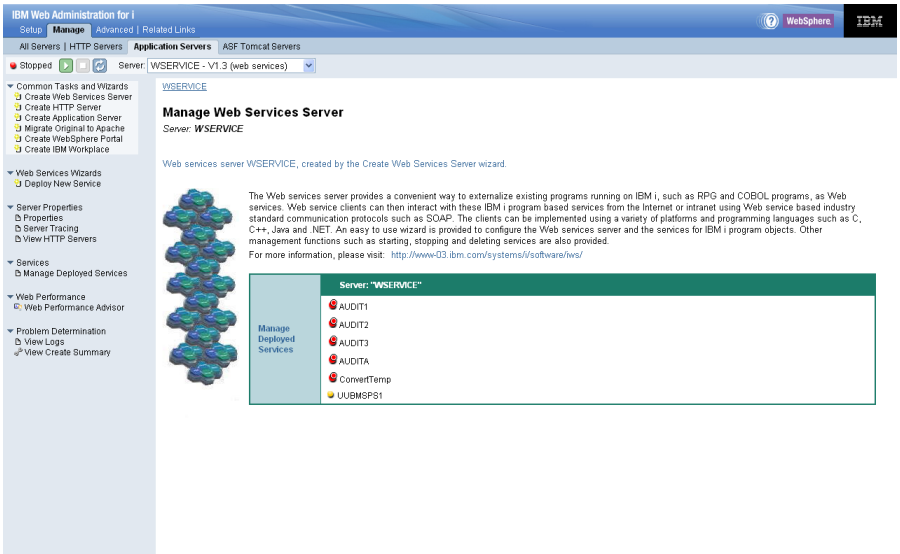


9. Use option U to uninstall a modeled WS from a Web Services Server.



10. Use IBM Web Administration interface to start/stop/test deployed Web Service.

    `http://{your_machine}:2001`



Web Services Limitations

This feature relies on the Web Services Server which is part of the IBM i operating system. Only 2E service programs generated as RPG ILE or COBOL ILE are candidates for exposure. The Web Service client portion is *not* created by this feature.

IBM states: "There are a few limitations within the Web services server regarding the deployment of services. To retrieve the most current information on restrictions, refer to the document located at */QIBM/ProdData/OS/WebServices/V1/server/docs/readme.txt*."

One of these limitations you need to be aware of is this: When a module is defined with a RCD parameter definition, in the generated source a data structure is used as the parameter for the *ENTRY PLIST. If some of the fields on the RCD parameter data structure are not being used as parameters, then due to an IBM limitation in the PCML generation, in the IBM Test client interface you will see redundant parameters (or subfields) named "_unnamed_1", "_unnamed_2", etc. These can be ignored and do not cause any problems.

Full details of the restrictions can be viewed in IBM's documentation.

Note: To call a deployed web service on a web services server, the user ID that is on the Properties for the web service needs to satisfy the following criteria: 1. The user ID must have the necessary authority to this program object and any other additional program objects. 2. The server profile QWSERVICE must have *USE authority to the user ID.

New Commands

In addition to the install/uninstall functionality in the EDIT FUNCTION DETAILS and WEB SERVICE INSTANCES panels, both of the new 2E Web Service commands can be invoked from the command line using the commands listed below:

Command	Function
YCRTWS	Installs a Web Service instance.
YUNSWS	Uninstalls a deployed Web Service instance.

YCRTWS (Create Web Service Instance)

The Create Web Service Instance (YCRTWS) command is used to install a web service to the IBM Web Services Server that contains an operation to invoke the RPG ILE or COBOL ILE program specified.

Update model? (UPDMDL)

Specifies if and how the model is updated.

***ADD**

New WS instance is added to the model (it must not already exist).

***NO**

The model is not updated at all.

***UPDINSSTS**

For a WS instance that already exists, the Installed status is updated.

Install to server? (INSTALL)

***YES**

The WS is installed to a Web Services Sever. The WS instance name must be unique to the specified Web Services Server.

***NO**

The web services server is not updated at all.

2E WS model file (MDLFIL)

Model-file-name

Specify the name of the 2E WS model file that owns the 2E WS model function.

2E WS model function (MDLFUN)

Model-function-name

Specify the name of the 2E WS model function to which the web service instance will be associated.

Machine (MACHINE)

Specifies the name of the machine onto which the web service instance will be deployed.

***CURRENT**

Refers to the local machine.

name

Specify the machine name. This can be the local machine or a remote machine. The machine name is not validated and the machine need not exist on the local, or indeed any network.

SERVER (char (10))

The name of the web services server in which the service will be installed.

server-name

Specify a web services server name.

SERVICE (char(25))

The name of Web service to be installed.

***PGMOBJ**

The program object name will be used.

service-name

Specify the name of the web service to be installed.

PGMOBJ (*PNAME)

The path to the ILE program or service program.

server-name

Specify the path to the ILE program or service program.

USRPRF (*VNM)

The user profile that the Web service will run under.

***USRPRF**

The web service will be created to run under the user profile that is invoking the YCRTWS command.

user-profile

Specify the user profile that the Web Service will run under.

Note: This user profile is granted access to all the Web service files and directories. If the service user ID is different from the server user ID, the server user ID must be given *USE authority to the service user ID.

***SRVID**

The Web services server user ID is used to run the service.

RTLIBL

A list of libraries, that will be added to the library list prior to invoking the Web service.

library-list

Specify the list of libraries.

***NOCHG**

No user-specified libraries will be added to the run-time library list.

YUNSWS (Uninstall Web Service)

The Uninstall Web Service (YUNSWS) command is used to uninstall a web service from the IBM Web Services Server that contains an operation to invoke the RPG ILE or COBOL ILE program specified.

The command can also update the installed status of the WS in the model or delete the modeled service entirely.

Update model?

This specifies if and how the model is updated.

***DELETE**

The WS instance is deleted from the model.

***NO**

The model is not updated at all.

***UPDINSSTS**

For a WS instance that already exists, the Installed status is updated

Uninstall from server? (UNINSTALL)

***YES**

The WS is uninstalled from the Web Services Sever.

***NO**

The web services server is not updated at all.

Web Services Server

The name of the web services server from which the service will be uninstalled.

server-name

Specify a web services server name

Web Service

The name of Web service to be uninstalled.

service-name

Specify the name of the web service to be installed.

Stop Service?

An indication whether the service should be stopped before an uninstall.

***YES**

The service is stopped before uninstall.

***NO**

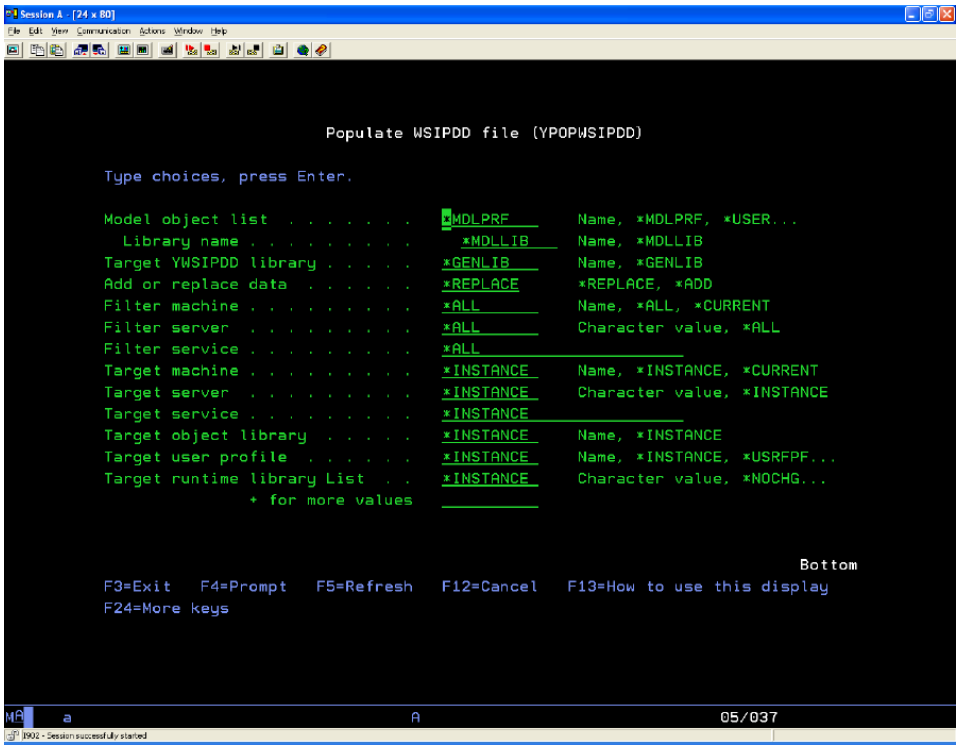
The service is not stopped before uninstall. An error will be returned if the service is active.

Running YUNSWS will delete or update the record in YWSICTLRFP file, providing UPDMDL is not *NO.

Web Service Remote Deployment

With Web Service Support in 2E, it is possible to bundle up Web Service instances that need to be ported and deployed to a remote machine. To do this you will need to use the Web Service Remote Deployment feature. The following new commands were created for this feature:

YPOPWSIPDD (Populate WSIPDD file)



Populates a Web Service Instance Portable Deployment Data file (WSIPDD). Once populated, a WSIPDD file can be moved to a remote machine, where the related YEXCWSIPDD (Execute WSIPDD) command can process the file to deploy web service instances on that remote machine.

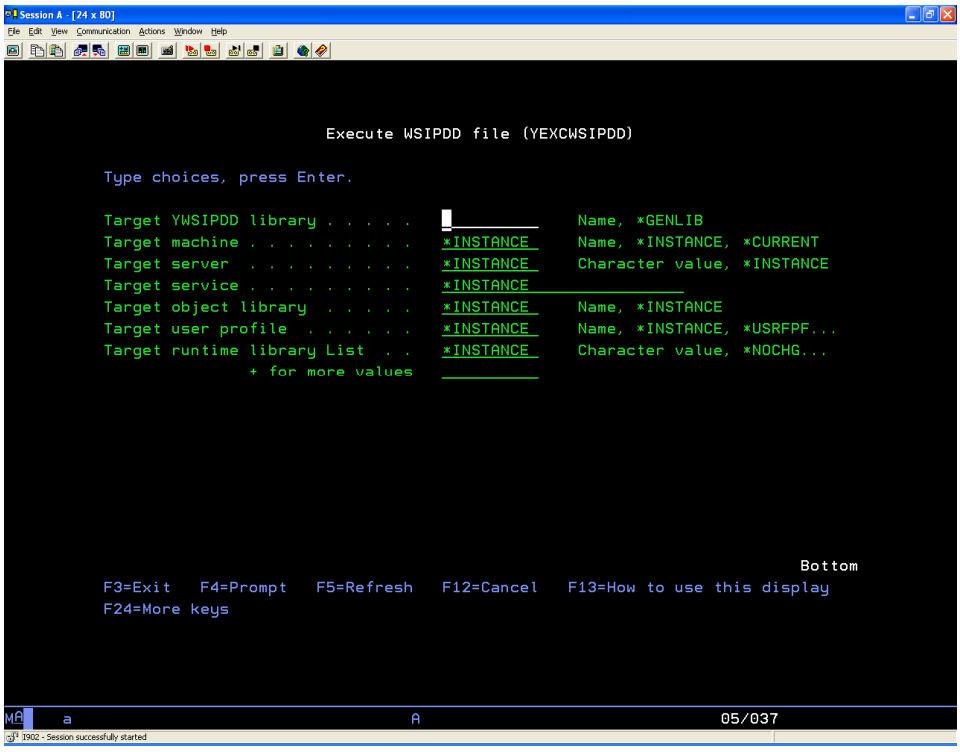
Note: Portable deployment does not require CA 2E or 1E to exist on the remote machine. See YEXWSIPDD for details.

The YPOPWSIDD command takes a model list as input. The model list is processed and for each function of type Web Service, additional processing occurs (items in the list that are not Web Service functions are ignored). For each Web Service function, all its web service instances are processed. Where a web service instance is not excluded due to filtering arguments on the YPOPWSIPDD command, an instance record will be created in the target WSIPDD file.

Note: The target WSIPDD file is always called YWSIPDDRFP, but the location is specified on the WSIPDDLIB parameter.

The Target parameters on the YPOPWSIPDD command allow the modelled web service instance data to be overridden when populated to the WSIPDD file.

YEXCWSIPDD (Execute WSIPDD file)



This command executes a Web Service Instance Portable Deployment Data file (WSIPDD), to deploy web service instances. The WSIPDD file should have been populated by the CA 2E YPOPWSIPDD command. Typically the WSIPDD file is then moved to a different machine to be executed by the YEXCWSIPDD command.

Note: Portable deployment does not require CA 2E or 1E to exist on the remote machine, on which the YEXCWSIPDD command is running. However, the YEXCWSIPDD command does require certain application objects to exist on the machine on which the command is running. These objects can be created in a target library using the YDUPAPPOBJ command (see the *WS argument for the DUPOPT) parameter. The YEXCWSIPDD command takes a WSIPDD file as an input.

Note: The input WSIPDD file is always called YWSIPDDRFP, but the location is specified on the WSIPDDLIB parameter.

For each record in the WSIPDD file with an ACTION flag of 'I' a web service instance will be deployed by the YCRTWS command. The Target parameters on the YEXCWSIPDD command allow the WSIPDD web service instance data to be overridden when deploying.

Note: If a web service instance is successfully deployed as a result of the YEXCWSIPDD command instance record's Action flag is updated to BLANK. The YINZWSIPDD command can be used to reset the WSIPDD Action flag.

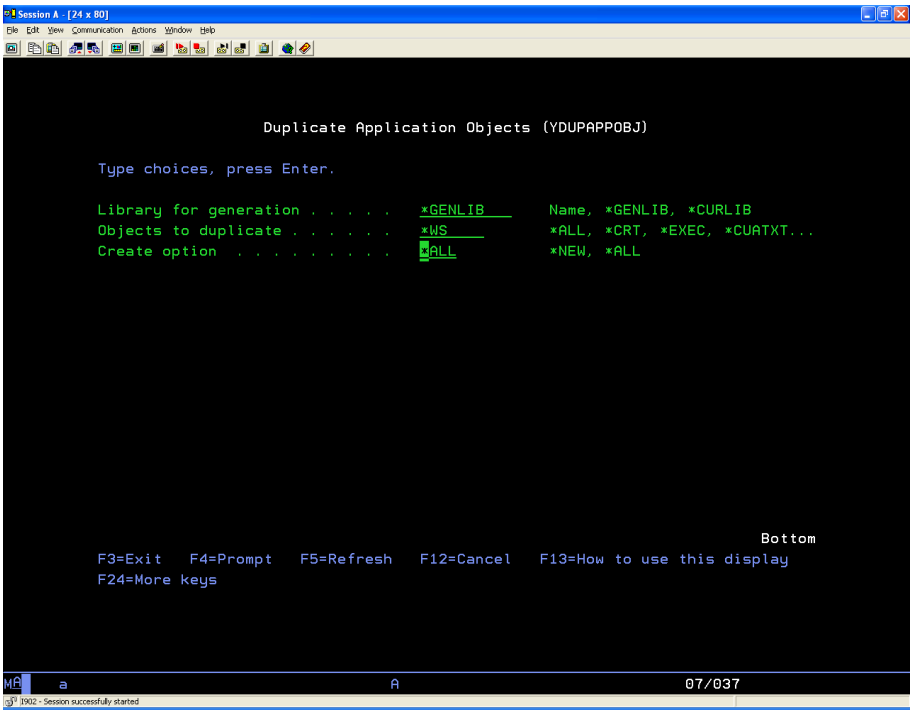
YINZWSIPDD (Initialise WSIPDD file)

The YINZWSIPDD command can be used to reset the Action flag on records in a WSIPDD file. See the YEXCWSIPDD command for more information.

To bundle up and deploy your remote Web Service instances

1. Create the Web Service Application objects using YDUPAPPOBJ

   There are a number of objects that need to be created in order to use Web Service Remote Deployment. These should be created by running the YDUPAPPOBJ command as follows:



   You can either create these objects into the Model Generation library, or into a new library to be used for deployment.

2. Build a model list containing all the Web Service functions whose modelled Web Service instances you want to portably deploy.

3. Use the YPOPWSIPDD (Populate WSIPDD file) command to populate the target YWSPDD file.

   See the command description above for details.

4. Copy the library that contains the WSIPDD file (this will be the library that you created your WS application objects into) to the remote machine.
5. On the remote machine, use the YEXCWSIPDD (Execute WSIPDD file) to deploy each of the Web Service instances in the portable deployment file. See command description above for details.

The YEXCWSIPDD command calls the YCRTWS command for each record in the portable deployment file, which installs the WS Instance on that machine. If a Web Service already exists on the machine, the deployment for that Instance record will fail, and the Action field will be left as 'I'.

References

- IBM Information Center
- Integrated Web Services for i
- Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language", 26 June 2007, W3C
- Web Services Architecture, W3C Working Group Note", 11 February 2004, W3C
- Standards and Web Services

Impact Analysis Enhancements

This feature enhances the 2E model impact analysis (Usages and References) processing, which allows you to control the identification or suppression of objects that have been commented out (deactivated) in an Action Diagram. To provide this, the Display Model Usages and Display Model References panels have a new field: Include Inactive Code.

Additionally, we improved the accuracy of impact analysis results associated with the deleted fields. For more information, see the section Hanging References in this chapter.

Include inactive code: *YES

When Include inactive code is set to *YES, the usage and reference expansion generally behaves as it does at previous releases.

An additional field Wrn (Warning) in the subfile record indicates, with an '*' in the second character of the field, when the corresponding Action Diagram call is deactivated (commented out). Additionally, any object that appears in the usage/reference report *only* by virtue of deactivated code is also marked with an '*'.

Note: The first character of the field Wrn (Warning) in the subfile record is currently unused.

Include inactive code: *NO

When Include inactive code is set to *NO, the usage and reference expansion will consider the deactivated (commented out) status of action diagram calls. This has 2 major effects:

1. Any deactivated action diagram calls within the reference/usage expansion of a target object will be ignored.
2. No objects will be included by virtue of the deactivated call to the object. In other words, the commented out function is not expanded.

Include inactive code: *IGN

When Include inactive code is set to *IGN, the usage and reference expansion behaves exactly as it does at previous releases, and there is no differentiation between active code and deactivated (commented out) code – i.e. all code is treated as active, even if it is commented out.

The additional field Wrn (Warning) in the subfile record is not relevant when this option is used.

Examples

The two examples in this section depict the new impact analysis for objects containing deactivated (commented out) code. Consider the following scenario:

- Function_A calls Function_B.
  Function_A **also** calls Function_C.
  Function_B calls Function_C.
- Function_B's call to Function_C is deactivated (commented out)
- Function C is analyzed for usages.

Example 1–Include Inactive Code = *YES

```
*Scope  . .  *NOMAX  Reason . .  *ALL*
Opt Object                Typ Atr Owner                Lvl Reason  WRN
    Function_C            FUN RPG 17009615             000 *OBJECT
    Function_B            FUN RPG 17009615             001 *ACTION  *
    Function_A            FUN RPG 17009615             002 *ACTION  *
    Function_A            FUN RPG 17009615             001 *ACTION
```

Note: Function_B's record has a '*' in character 2 of the WRN field, to indicate that the action diagram call to Function_C has been deactivated (commented out). Hence the Function_A (LVL 002) which calls Function_B) is also marked with a '*' in the WRN field. However the Function_A (LVL 001) which calls Function_C directly) is not marked with a '*' in the WRN field.

Example 2–Include Inactive Code = *NO

```
*Scope  . .  *NOMAX  Reason . .  *ALL*
Opt Object                Typ Atr Owner                Lvl Reason  Wrn
    Function_C            FUN RPG 17009615             000 *OBJECT
    Function_A            FUN RPG 17009615             001 *ACTION
```

Note: Because Function_B's call to Function_C is deactivated (commented out)* Function_B is not included in the usages report and not expanded, therefore Function_A (LVL 002) is also not included.

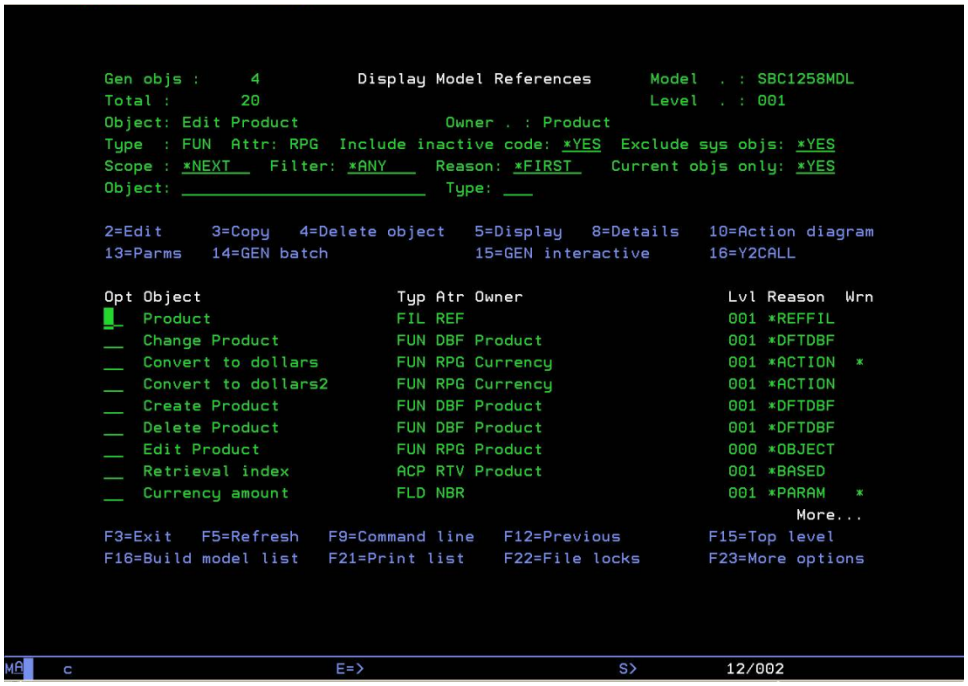
Example 3–Include Inactive Code = *IGN

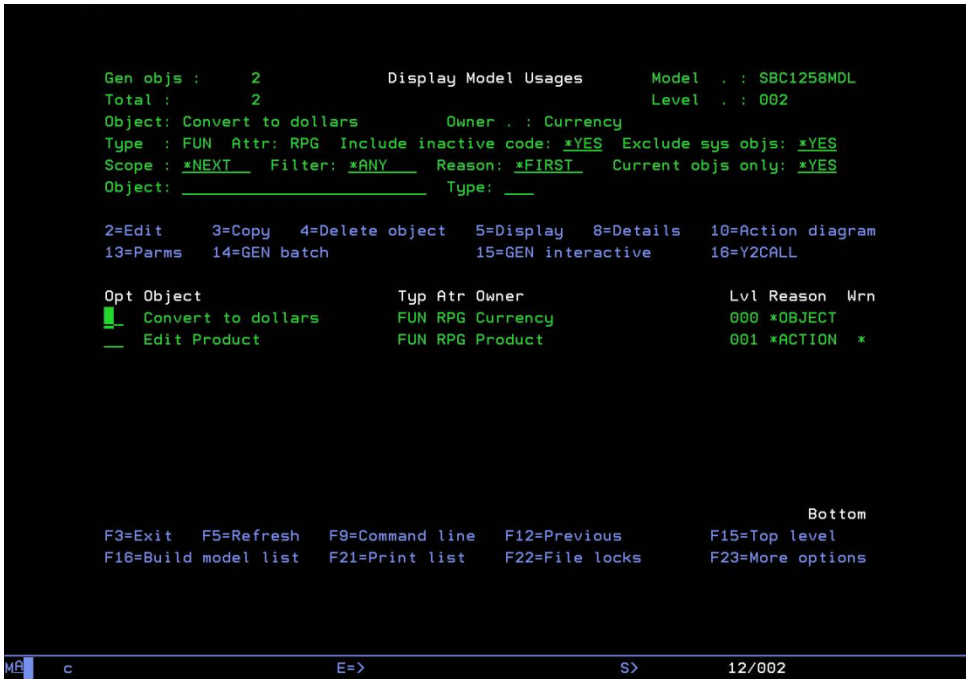
```
*Scope . . *NOMAX Reason . . *ALL*
Opt Object Typ Atr Owner Lvl Reason WRN
Function_C FUN RPG 17009615 000 *OBJECT
Function_B FUN RPG 17009615 001 *ACTION
Function_A FUN RPG 17009615 002 *ACTION
Function_A FUN RPG 17009615 001 *ACTION
```

Note: These are the same results that you would see with 8.1 SP2. They also look the same as with Include Inactive Code=*YES but the WRN field is not used, since there is no differentiation between active and deactivated code.

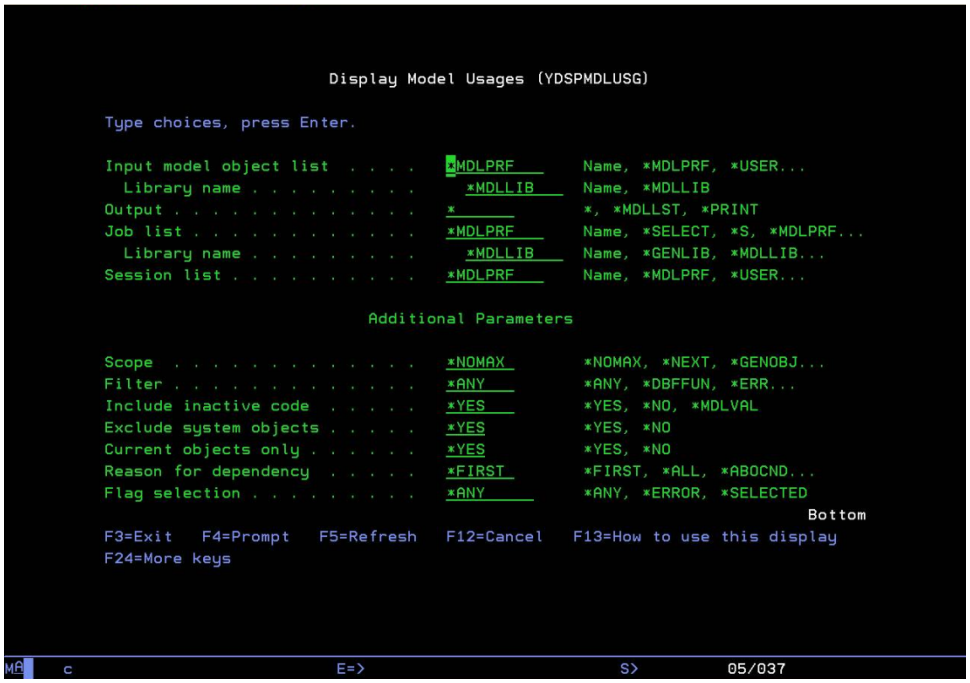
The following screens show the new field:

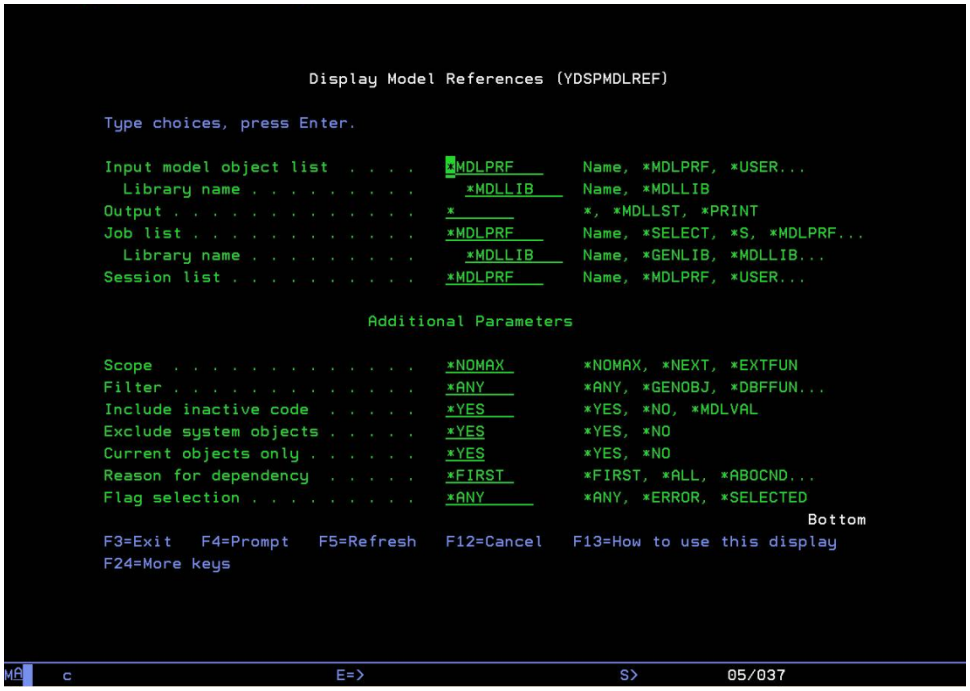
YDSPMDLREF:


Gen objs : 4 Display Model References Model . : SBC1258MDL
Total : 20 Level . : 001
Object: Edit Product Owner . : Product
Type : FUN Attr: RPG Include inactive code: *YES Exclude sys objs: *YES
Scope : *NEXT Filter: *ANY Reason: *FIRST Current objs only: *YES
Object: Type:
2=Edit 3=Copy 4=Delete object 5=Display 8=Details 10=Action diagram
13=Parms 14=GEN batch 15=GEN interactive 16=Y2CALL
Opt Object Typ Atr Owner Lvl Reason Wrn
Product FIL REF 001 *REFFIL
Change Product FUN DBF Product 001 *DFTDBF
Convert to dollars FUN RPG Currency 001 *ACTION *
Convert to dollars2 FUN RPG Currency 001 *ACTION
Create Product FUN DBF Product 001 *DFTDBF
Delete Product FUN DBF Product 001 *DFTDBF
Edit Product FUN RPG Product 000 *OBJECT
Retrieval index ACP RTV Product 001 *BASED
Currency amount FLD NBR 001 *PARAM *
More...
F3=Exit F5=Refresh F9=Command line F12=Previous F15=Top level
F16=Build model list F21=Print list F22=File locks F23=More options
MA c E=> S> 12/002

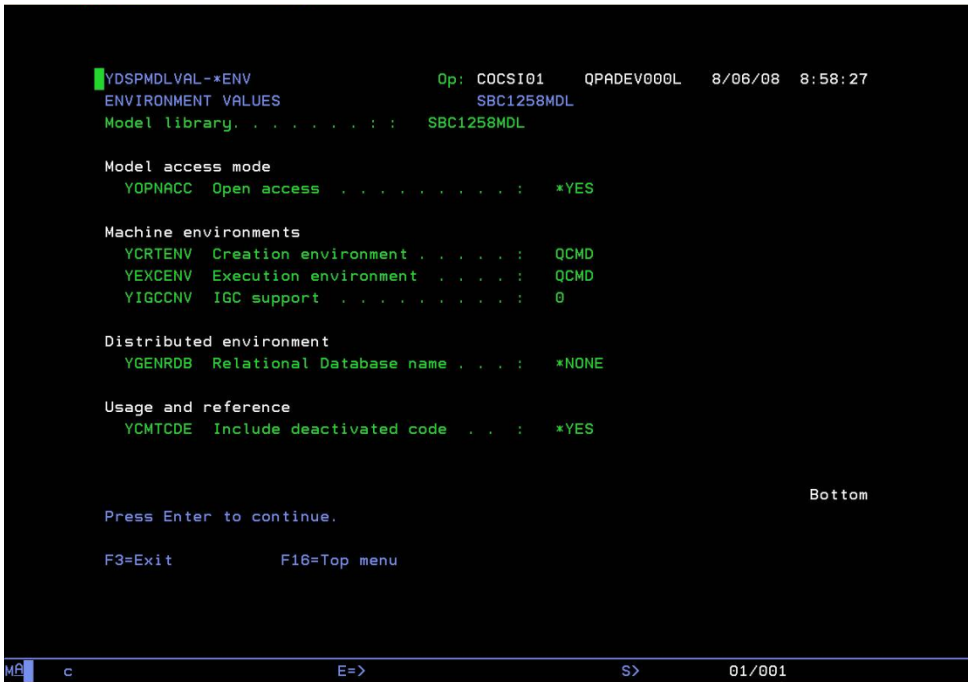

Gen objs : 2 Display Model Usages Model . : SBC1258MDL
Total : 2 Level . : 002
Object: Convert to dollars Owner . : Currency
Type : FUN Attr: RPG Include inactive code: *YES Exclude sys objs: *YES
Scope : *NEXT Filter: *ANY Reason: *FIRST Current objs only: *YES
Object: Type:
2=Edit 3=Copy 4=Delete object 5=Display 8=Details 10=Action diagram
13=Parms 14=GEN batch 15=GEN interactive 16=Y2CALL
Opt Object Typ Atr Owner Lvl Reason Wrn
Convert to dollars FUN RPG Currency 000 *OBJECT
Edit Product FUN RPG Product 001 *ACTION *
Bottom
F3=Exit F5=Refresh F9=Command line F12=Previous F15=Top level
F16=Build model list F21=Print list F22=File locks F23=More options
MA c E=> S> 12/002

YDSPMDLUSG:


Display Model Usages (YDSPMDLUSG)
Type choices, press Enter.
Input model object list . . . . *MDLPRF Name, *MDLPRF, *USER...
Library name . . . . . *MDLLIB Name, *MDLLIB
Output . . . . . * *, *MDLLST, *PRINT
Job list . . . . . *MDLPRF Name, *SELECT, *S, *MDLPRF...
Library name . . . . . *MDLLIB Name, *GENLIB, *MDLLIB...
Session list . . . . . *MDLPRF Name, *MDLPRF, *USER...
Additional Parameters
Scope . . . . . *NOMAX *NOMAX, *NEXT, *GENOBJ...
Filter . . . . . *ANY *ANY, *DBFFUN, *ERR...
Include inactive code . . . . . *YES *YES, *NO, *MDLVAL
Exclude system objects . . . . . *YES *YES, *NO
Current objects only . . . . . *YES *YES, *NO
Reason for dependency . . . . . *FIRST *FIRST, *ALL, *ABOCND...
Flag selection . . . . . *ANY *ANY, *ERROR, *SELECTED
Bottom
F3=Exit F4=Prompt F5=Refresh F12=Cancel F13=How to use this display
F24=More keys
MA c E=> S> 05/037


Display Model References (YDSPMDLREF)
Type choices, press Enter.
Input model object list . . . . *MDLPRF Name, *MDLPRF, *USER...
Library name . . . . . *MDLLIB Name, *MDLLIB
Output . . . . . * *, *MDLLST, *PRINT
Job list . . . . . *MDLPRF Name, *SELECT, *S, *MDLPRF...
Library name . . . . . *MDLLIB Name, *GENLIB, *MDLLIB...
Session list . . . . . *MDLPRF Name, *MDLPRF, *USER...
Additional Parameters
Scope . . . . . *NOMAX *NOMAX, *NEXT, *EXTFUN
Filter . . . . . *ANY *ANY, *GENOBJ, *DBFFUN...
Include inactive code . . . . . *YES *YES, *NO, *MDLVAL
Exclude system objects . . . . . *YES *YES, *NO
Current objects only . . . . . *YES *YES, *NO
Reason for dependency . . . . . *FIRST *FIRST, *ALL, *ABOCND...
Flag selection . . . . . *ANY *ANY, *ERROR, *SELECTED
Bottom
F3=Exit F4=Prompt F5=Refresh F12=Cancel F13=How to use this display
F24=More keys
MA c E=> S> 05/037

The value of this field is controlled by a new model value, YCMTCDE:



This is shipped with a value of *YES to ensure current processing remains unchanged. Objects that were commented out are still picked up in Usages and References processing. If you want to suppress the expansion and listing of Inactive code, then you need to set this Model Value to *NO.

Note: You can also change this field on the Display Model Usages and Display Model References to tailor the display to suit your requirements at that time.

Hanging References

There is new processing in the YCHKMDL command that ensures the removal of dangling surrogate records from various internal files. These surrogate records were causing outdated *DEVATR and *FUNPDT usages to display for a field.

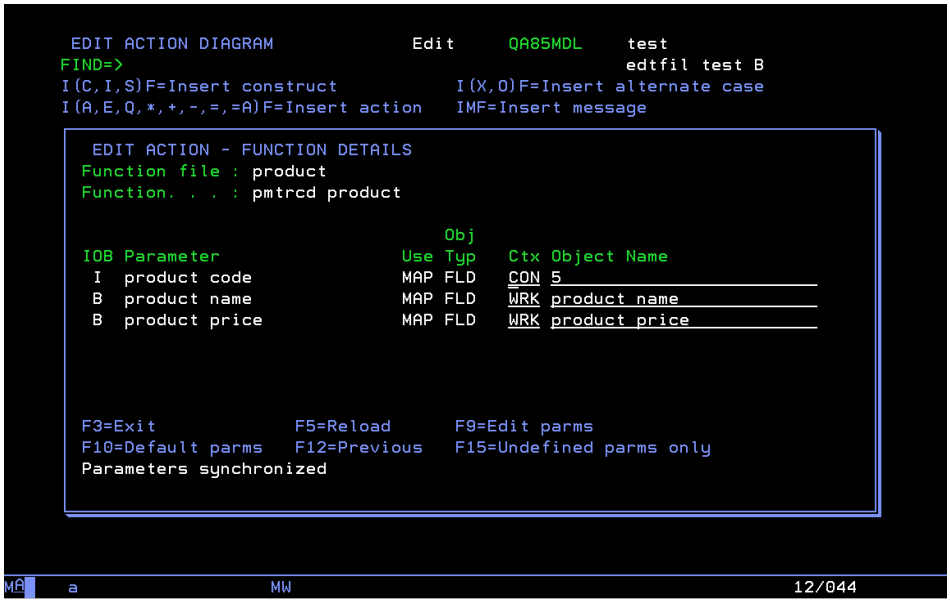
Notes:

- This change does not address all dangling surrogate problems; only the ones mentioned in this section.
- Due to this change, YCHKMDL may take longer to run than at 8.1 SP2.

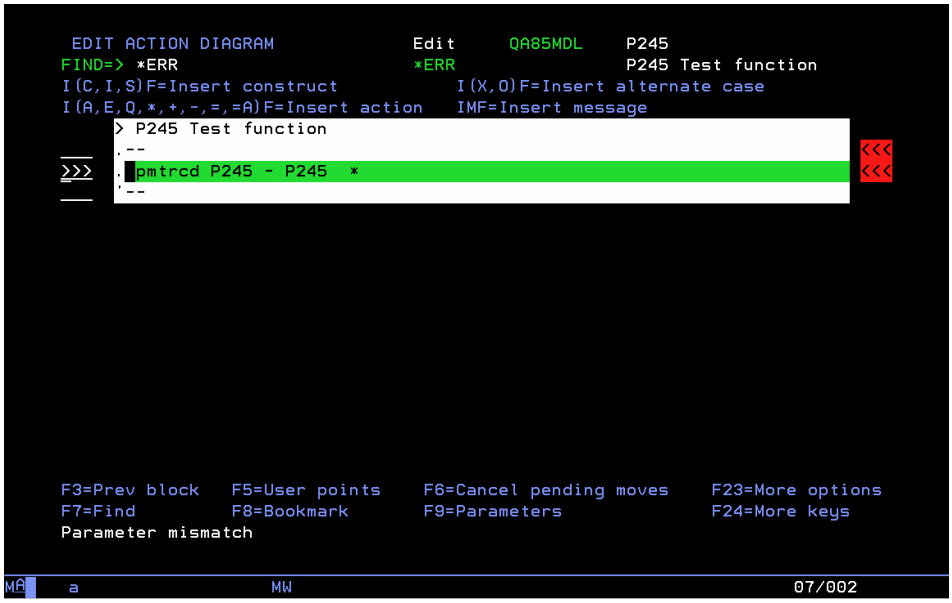
Also, the Action Diagram editor was enhanced to remove any outdated *PARAM references for the parameter interface when you FF on a function call.

These 'hanging' references most likely resulted from parameters being dropped from a function at some point in the past. Prior to r8.5, when parameters were dropped from a function, the YMSGACTRFP record for the dropped parameter was never removed.

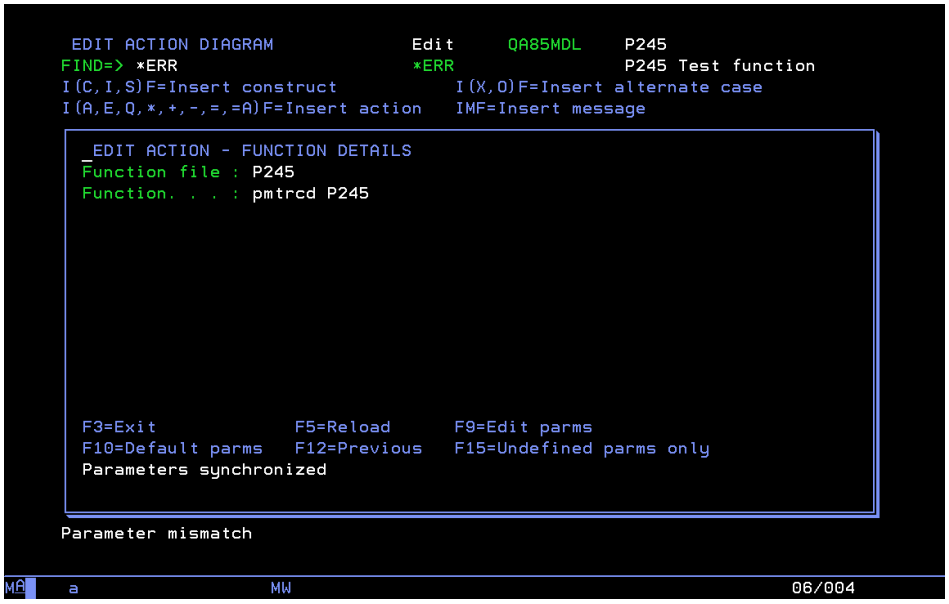
With r8.5, if you FF on a function call and there is a parameter mismatch due to a change in the parameters defined for the called function (such as a dropped parameter), the parameters are automatically synchronized and message Y2V0347 (Parameters synchronized) displays:



The Action Diagram Find Services picks up these parameter mismatch errors when Find Option 2-Error is used. This is signaled with message Y2V0348 (Parameter mismatch):



You can FF into the function call to synchronize the parameters:



These parameter mismatch errors are also picked up by the YCHKFUNACT command and message Y2V0348 written to the spooled file, or displayed on screen if you are in *EDIT mode.

Be aware that after the parameters are synchronized, you must save the function when exiting the Action Diagram Editor. This ensures that the synchronization changes are written to file (if you have the YACTUPD model value set to *CALC, the Action Diagram Editor flags the function as changed.

You might notice that this message is displayed frequently if you are running YCHKFUNACT over very old functions. This is due to called functions in the Action Diagram having had parameters dropped at some point.

Improved Model Selection and Positioning

This enhancement provides changes to positioning and selection in two ways:

- EDIT FUNCTIONS (*ARRAYS) (Array Field) and EDIT DATABASE RELATIONS (Object field) have been enhanced to allow ?{value} to be entered as a positioner value. {value} is passed to the secondary panel.

Example 1

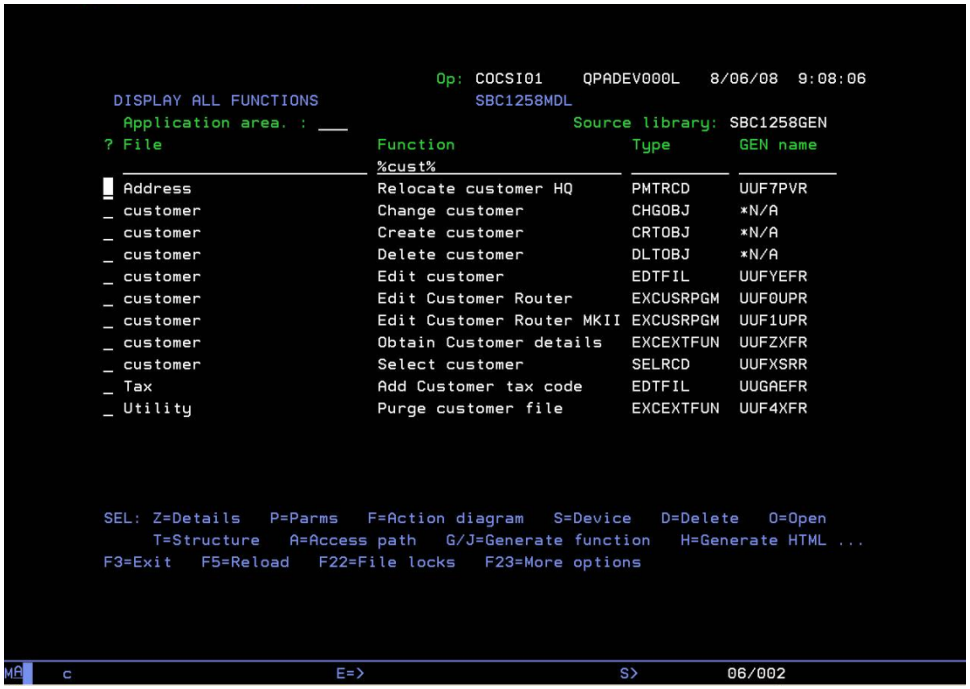
- Go to EDIT Functions over *ARRAYS file (Primary Panel).
- Enter ?%ABC% (Search+Filter term) in the Array positioner field and press ENTER; displays EDIT ARRAY (Secondary Panel) displaying only records that contain ABC in the Array name.

Example 2

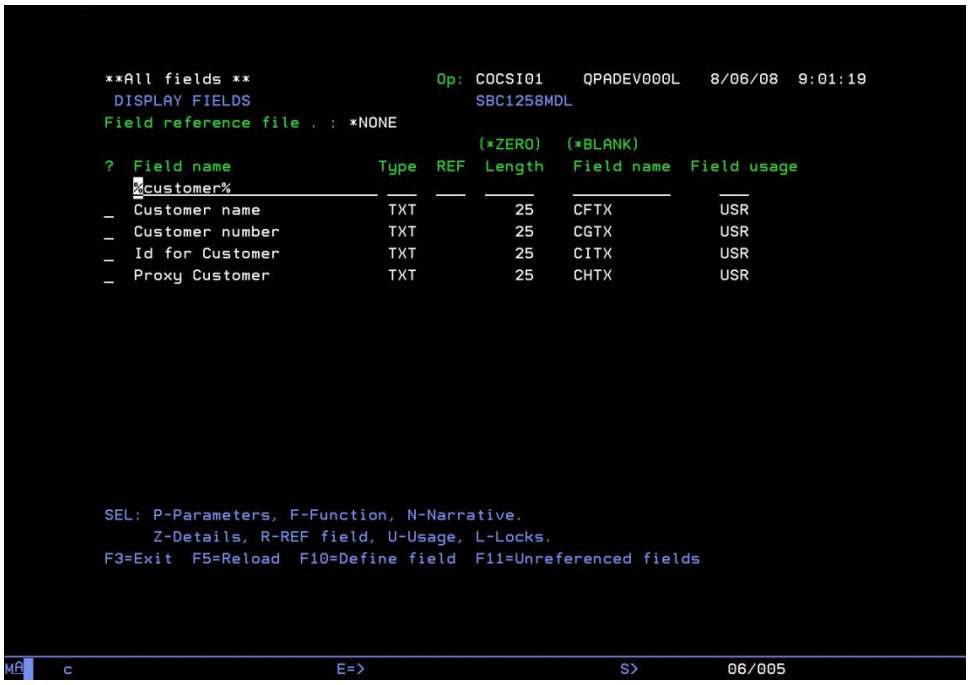
- Go to EDIT Functions over *ARRAYS file (Primary Panel).
- Enter ?ABC (Search + Position term) in the Array positioner field and press ENTER; displays EDIT ARRAY (Secondary Panel) with subfile record pointer moved to the first record that is Equal to or Greater Than 'ABC'...all subsequent records will be displayed.
- Within the model, design-time, subfile search/positioning has been enhanced to include a 'contains' search. %generic-name% : Generic selection ('contains'). By specifying a '%' as the first and last character of the positioning value, you might obtain a list of only those CA 2E objects that have the generic-name anywhere in the name. For instance, a value of '%Book%' will cause only the objects that refer to object names that contain 'Book' anywhere in the name to be displayed e.g. 'Book', 'User Book', 'New bookings'.

Note: This *contains* search is not case sensitive.

For example, on the Display all Functions panel, you can search for all functions with the text 'cust' in the name as follows:



Similarly, you can search for all fields with field name containing %customer% in Display Fields:



The following table shows which panels and fields have the new 'contains' search support:

Panel	Field
EDIT FUNCTIONS (*ARRAYS)	Function
EDIT FUNCTIONS (*ARRAYS)	Array
EDIT FUNCTIONS (FILE)	Function
EDIT FUNCTIONS (FILE)	Access Path
EDIT ARRAY	Array
DISPLAY OBJECTS	Description
EDIT DATABASE RELATIONS	Object
EDIT DATABASE RELATIONS	Referenced Object
DISPLAY ALL FUNCTIONS	File
DISPLAY ALL FUNCTIONS	Function
DISPLAY ALL ACCESS PATHS	File
DISPLAY ALL ACCESS PATHS	Access Path

Panel	Field
DISPLAY ALL FIELDS	Fields

Web Option Environments

Prior to r8.5, a web option set up consisted of two web option product libraries-Y2WEB and Y2WEBVENG. CA 2E r8.5 introduces the concept of Web Option Environments, which enables you to separate user data from non-data objects.

A Web Option environment consists of up to three different libraries:

- Web Option product library-this library is shipped as Y2WEB and contains the non-data objects required to run Web Option. It also contains a number of data objects that contain control data, such as YMLSSYNRFP.
- Web Option LDO library-the default installed library is called Y2WEBVENG (for English), but other languages are available. This library contains the language-specific data objects. Depending on user requirements, this library can be *merged* with the product library.
- Web Option Environment Library-this library is not installed, but is created using the *Create Web Option Environment* (YCRTW2EENV) command, which is new for r8.5. When created, this library contains copies of certain data objects from both the product and LDO libraries, as well as data areas linking it to those product and LDO libraries.

  Note: For detailed information on this command, see the section *Web Option Commands*.

These are the benefits of having a separate data library:

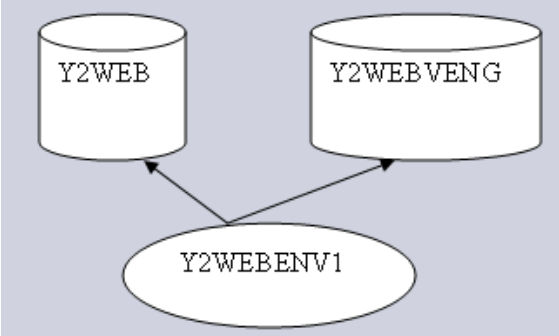
A single *set* of product library and LDO library can be used with multiple different data libraries, providing the separation of your unique data from the system defaults.

- A simpler upgrade process, which ensures that your unique user data is not overwritten by the shipped data.

  Note: This simplification of the upgrade will not take effect until future releases, when upgrading from r8.5 to a newer release. Upgrading from r8.1 SP2, or earlier, to r8.5 still requires the original upgrade process.

A web option environment library is linked to the product libraries when it is created by means of the YCRTW2EENV command. The following scenarios detail how this might look:

Scenario 1



In Scenario 1, one web option environment library, Y2WEBENV1, was created from Y2WEB and Y2WEBVENG. To create this environment library, you can use the following command:

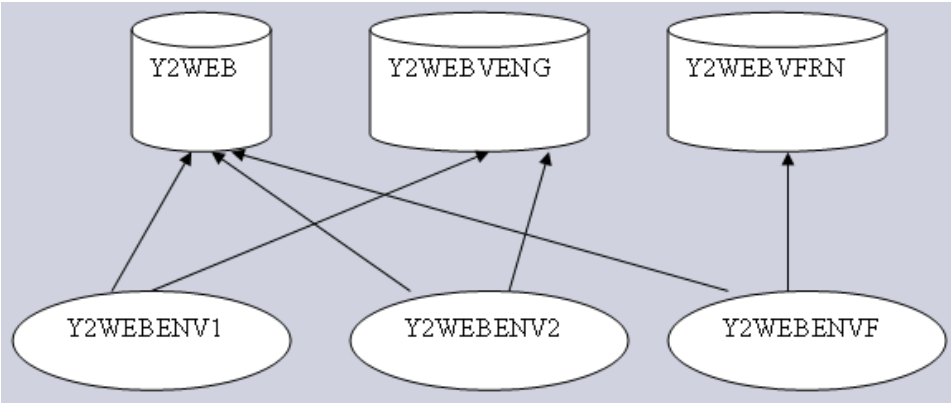
```
YCRTW2EENV W2EENV(Y2WEBENV1) W2ELIB(Y2WEB) LDOLIB(Y2WEBVENG)
```

Two data areas in Y2WEBENV1 link the library to Y2WEB and Y2WEBVENG:

- YW2ELDORFA (the Web Option LDO library)
- YW2EPRDRFA (the Web Option library)

It is also possible to create multiple Environment libraries all linked to the same Y2WEB library (and pointing to the same or different LDO libraries). In this way you can keep user data (such as cross-reference data and skeletons) for different applications, or test environments, separate. This is shown in the following diagram:

Scenario 2



Y2WEBENV1 and Y2WEBENV2 both point to the same Product libraries, but the cross reference records and generated skeletons are stored separately in the relevant environment library.

If you wanted to do any customization to the MLS Syntax file, or add any customization elements to the Element Customization file, then any changes you make are contained in the Y2WEBENV1 or Y2WEBENV2 libraries and the YMLSSYNRFP or YELMCSTRFP files in Y2WEB are not affected. In this case, you can retain unaltered, default product libraries.

The third environment library Y2WEBENVF is for French applications and so this environment points to Y2WEBVFRN (the French LDO library) and Y2WEB.

These new concept of environments provide you with some flexibility in how your data is stored and managed.

Upgrade Existing Web Option Libraries

To upgrade your existing SP2 (or earlier) Web Option libraries, the process is much the same as it was, in that you still need to run the YINZW2EENV command to upgrade your product libraries. This command now calls the YCRTW2EENV command.

Note that if you have 2 sets of Web Option Product libraries, maybe one for Test and one for Production, it is necessary to upgrade each set of libraries separately, as follows:

Test Product Libraries:

Y2WEBTST

Y2WEBVTST

Production Product Libraries:

Y2WEBPRD

Y2WEBVPRD

To upgrade the Test environment

1. Restore the new 8.5 Product libraries as e.g. Y2WEB85T and Y2WEBV85T
2. Change the YW2ELDORFA data area and the WEB2E_SVR job description in Y2WEB85T to refer to the new library names
3. Run the following command to upgrade Y2WEBTST:

   ```
   YINZW2EENV CURW2ELIB(Y2WEBTST) W2ELIB(Y2WEB85T)
   ```

   This process runs through the same processing as it did at SP2, with the exception that before the command finishes, the YCRTW2EENV command (Create Web Option Environment) is prompted and should be run as follows:

   ```
   YCRTW2EENV W2EENV(Y2WEBENV) W2ELIB(Y2WEB85T) LDOLIB(Y2WEB85VT)
   ```

   Where Y2WEBVENV is the name you want to call your Environment library.
4. Change the HTTP server configuration to point to the YROUTER program in Y2WEBENV rather than the YROUTER program in Y2WEBTST. This can be done from a green-screen as follows:
   a. End the HTTP server:

      ```
      ENDTCPSVR SERVER(*HTTP) HTTPSVR(Y2WEBSVR)
      ```

   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 one of the following two lines:

      either:

      ```
      <Directory /QSYS.LIB/Y2WEBTST.LIB/YROUTER.PGM>
      ```

      or:

      ```
      <Directory /QSYS.LIB/Y2WEBTST.LIB/>
      ```

   d. Change the line to reference the environment library:

      ```
      <Directory /QSYS.LIB/Y2WEBENV.LIB/>
      ```

   e. Page down until you find the following line (at or near the bottom of the file):

      ```
      ScriptAliasMatch ^/WEB2E$ /QSYS.LIB/Y2WEBTST.LIB/YROUTER.PGM
      ```

   f. Change the line to reference the version of YROUTER in the environment library

      ```
      ScriptAliasMatch ^/WEB2E$ /QSYS.LIB/Y2WEBENV.LIB/YROUTER.PGM
      ```

   g. Press F2 to save your changes and press F3 to exit.
   h. Restart the HTTP server

      ```
      STRTCPSVR SERVER(*HTTP) HTTPSVR(Y2WEB)
      ```

5. Change any 2E models that used the Y2WEBTST library so that their job descriptions include the new environment library in the library list above the new LDO and product libraries. Also change them so that the YW2ELIB model value refers to the environment library rather than the product library:

   ```
   CHGMDLVAL MDLVAL(YW2ELIB) VALUE(Y2WEBENV)
   ```

Repeat these steps for the Production libraries, thereby creating a 2nd set of 8.5 product libraries, and a new Environment library that links to these Production product libraries.

Web Option Commands

If you prompt certain Web Option commands, you will notice a new parameter called W2EENV, instead of W2ELIB, and the text says "Web Option Environment" rather than library. These changed commands are as follows:

Command	Description
YCRTW2EHTP	Creates the Web Option HTTP server instance
YENDW2ESVR	End Web Option server
YGENMLS	Generate Markup Language Skeleton
YPRCSKL	Process HTML skeleton
YSTRW2ESVR	Start Web Option server
YWRKW2EVAL	Work with Web Option Control Values

Going forward, for these commands you should use the name of the Web Option environment library rather than the Web Option product library, as you would have with previous releases. For example, to start the Web Option server using the Y2WEBENV environment, use the following command:

```
YSTRW2ESVR W2ELIB(Y2WEBENV)
```

There is no difference at runtime–your screens all look the same. However, when new skeletons are generated, they are generated into the YMLSSRC file in Y2WEBENV, and the cross-reference records in YSCRXRFP and YMLSXRFP are created in the Y2WEBENV copies of these files.

Create Web Option Environment Command (YCRTW2EENV)

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).

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:

Y2WEB

	*INIT	*COPY	*MOVE
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.

Additional Enhancements and Updates

The following minor enhancements and updates are included in CA 2E r8.5.

Default Target HLL Language for New Models is RPGIV

The default language used when a new 2E model library is created was changed to be RPGIV. The following two control data areas were changed:

- Y2SY/YHLLGENSYA
- Y2SYMDL/YTGTLANRFA)

The PromptOverride program for the YCRTMDLLIB command was changed to allow HLL language codes longer than 4 bytes (which only allowed *RPG and *CBL).

Note: This change only affects the default generation language; you can override it to any other valid language.

YCVTCNDVAL and YCVTMDLMSG Enhancement

The Convert Condition Values (YCVTCNDVAL) and Convert Model Messages (YCVTMDLMSG) commands have been enhanced to include a Model List (MDLLST) parameter, whereby you can convert a subset of the condition values or model messages in the model rather than all of them.

These commands are further defined in the *Command Reference Guide*.

New Versioning User Exit Program

When a version of a function is made current, its source member name is switched with the previously-current object's source member name. This is not obvious from the CA 2E user interface; therefore, we made the following changes to address this:

When "Making Current" a version of a function:

1. Send message Y2C0314 "Warning: Current object's source member name switched with previously-current object."
2. Call a new exit program YRDROBJR2C.

   YRDROBJR2C, which ships inY2SY, and the source is shipped in Y2SYSRC/QCLSRC.

Note: The exit program could be used to make post-switch amendments.

New Reason Code for DSPMDLUSG and DSPMDLREF

We created the following new Reason Code for DSPMDLUSG and DSPMDLREF:

***DSLDBF**

This indicates the underlying usage of a CRTOBJ, CHGOBJ or DLTOBJ by an EDTRCD, if the Create, Change or Delete Record function options are set to N. It also means that if the CRTOBJ or DLTOBJ function is called in the EDTRCD action diagram, and the Create Record/Delete Record is set to N, the usage is picked up.

Rebranding

The CA 2E product and user documentation are updated to reflect IBM's current branding the IBM Power Systems hardware (formerly AS/400, iSeries and System i) and the IBM i operating system (formerly OS/400 and i5/OS)

The full name of the toolkit product is now CA 2E Toolkit (not CA 2E 400 Toolkit).

EJB Option No Longer Supported

EJB Option is no longer shipped in a CA 2E r8.5 - so this add-on product can no longer be used to create PCML or Java stubs for a CA 2E function.

While there is currently no way within CA 2E to automatically generate Java stubs (that EJB Option would have generated)…it is possible to automatically generate PCML.

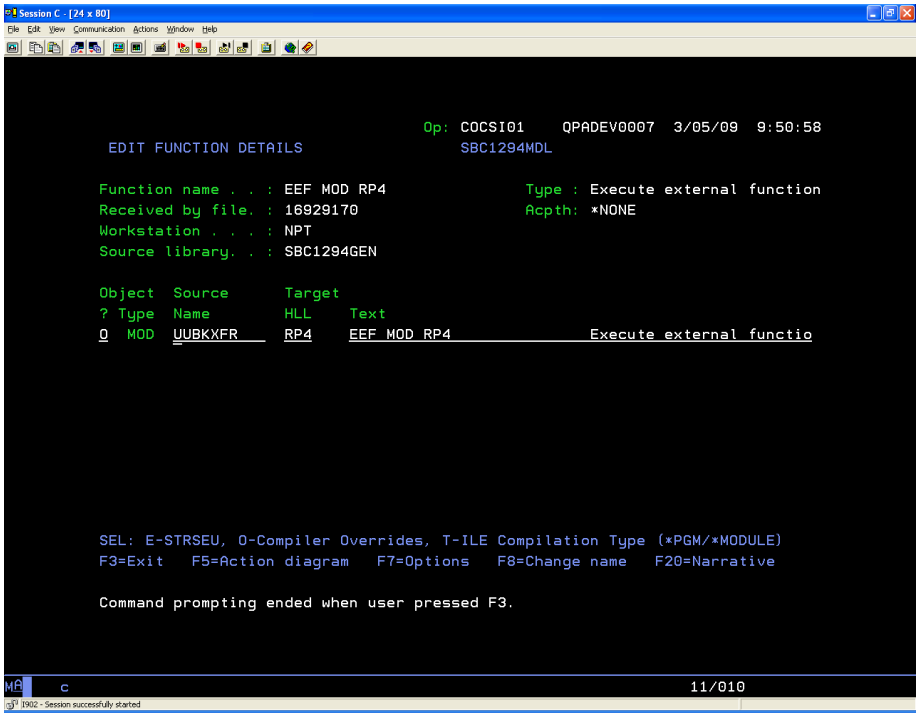
At i5/OS V5R4 and i5/OS V6R1 an external PCML file can be generated for RPG ILE and COBOL ILE modules and programs during the compilation step using the PGMINFO and INFOSTMF parameters on the compilation commands.

NOTE: PCML information can also be created within a *MODULE, as described in the Web Service Creation feature.

Example

The following is an example of using Compiler Override in a CA 2E model to create an external PCML file for a 2E RPG ILE module.

1. From EDIT FUNCTION DETAILS select O-Compiler Overrides:



2. Change the overrides as shown in the following screenshot.

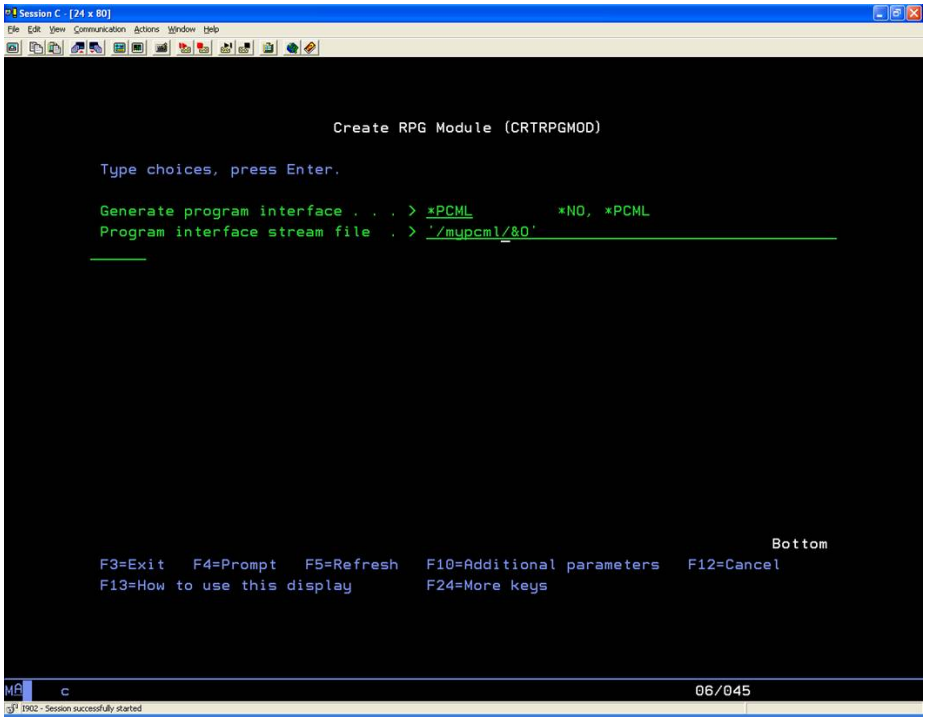
   a. Change Generate program interface (PGMINFO) to *PCML.

   b. Change the Program interface stream file (INFOSTMF) to specify the location of the PCML to be generated.

      NOTE: As shown in the example below, substitution variables are supported, e.g. &O can be used to represent Object Name. This is especially useful when changing the compilation command defaults on the model level

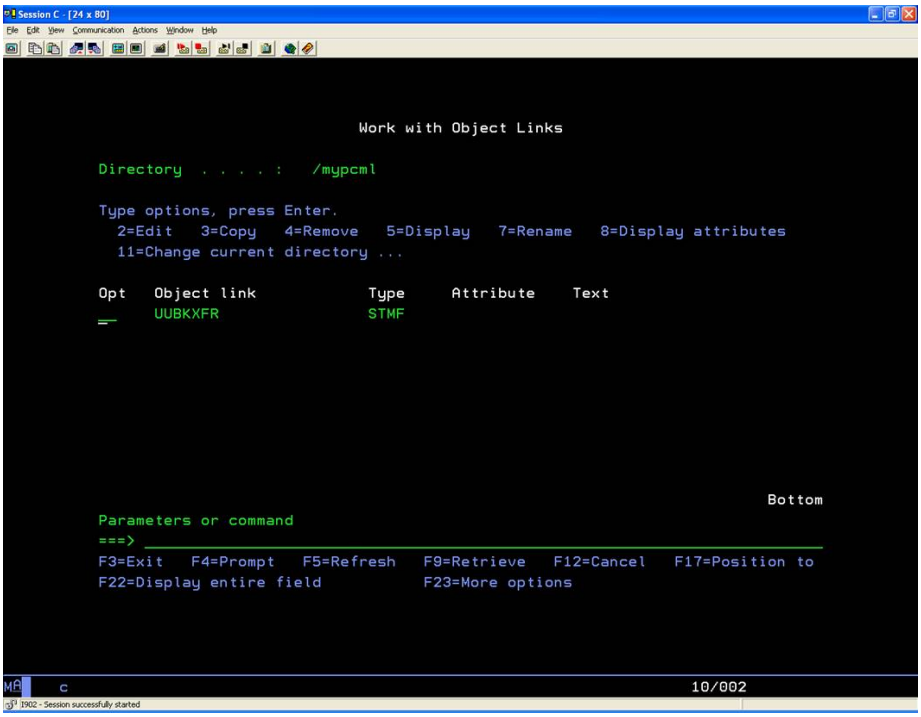
```
      (see *MESSAGES:

         *CRTBNDCBL                EXC   Y2U1028   Y2USRMSG
      *CRTBNDRPG                EXC   Y2U1022   Y2USRMSG
      *CRTCBLMOD                EXC   Y2U1029   Y2USRMSG
      *CRTCBLPGM                EXC   Y2U1006   Y2USRMSG)
…instead of compiler override at a function level.
```

In the following screenshot the command override at a function level has been overridden to specify *PCML and the PCML generation will be a stream file (STMF) called the same name as the object since &O resolves to, in this case, UUBKXFR:

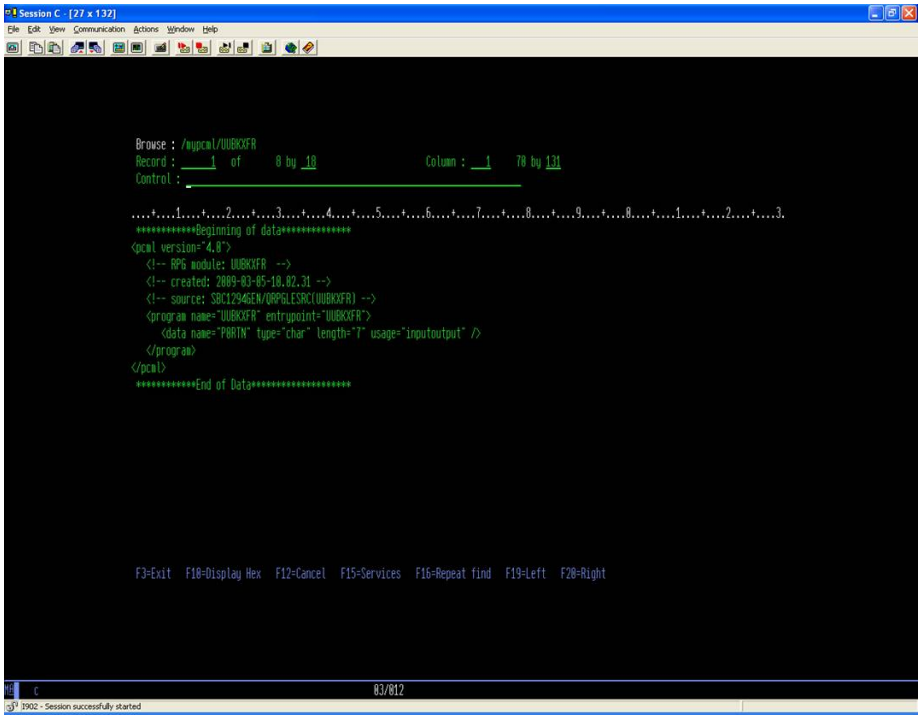


3. Following successful compilation of the module, the PCML file has been created in the target location.

   For example, use wrklnk '/mypcml/*' to verify.



Example of PCML contents:



IBM Information centre (PCML creation), references

In order to correctly support CA 2E r8.5-particularly Web Service Support-is important to ensure that the latest IBM Cumulative Package and Latest HTTP Group PTFs are installed:

For V5R4

- IBM i5/OS Information Centre (V5R4)
- Program Call Markup language
- Create RPG Module (CRTRPGMOD)
- Create Bound RPG Program (CRTBNDRPG)
- Create COBOL Module (CRTCBLMOD)
- Create BOUND COBOL Program (CRTBNDCBL)

For V6R1

- IBM i5/OS Information Centre (V6R1)
- Program Call Markup language
- Create RPG Module (CRTRPGMOD)

- Create Bound RPG Program (CRTBNDRPG)
- Create COBOL Module (CRTCBLMOD)
- Create BOUND COBOL Program (CRTBNDCBL)

New Model Value YTRNAPI - Convert case API

To support control field filters, CA 2E generates calls to the IBM API QDCXLATE in 2E DSPFIL and SELRCD functions. The old IBM API QDCXLATE does not correctly handle case conversion for certain code pages; for example, CCSID 273 German for upper/lower case Umlaut.

As a workaround, we created a new model value - YTRNAPI - Convert case API - *CHAR(8). This model value ships with value QDCXLATE so that the current functionality is unchanged. With this value the product behaves the same as it did with CA 2E r8.1 SP2.

If the model value contains any value other than QDCXLATE, then at generation time for DSPFIL and SELRCD functions, source will be generated that, at runtime, invokes a program whose name is equal to the value in the YTRNAPI model value. Also, the third and fourth parameters that were generated and required for a QDCXLATE call are not generated. For example, the non-QDCXLATE call will be made using only length and value of the data for a control field filter.

A sample alternative to QDCXLATE is available: YTRNAPIC. YTRNAPIC is a shell program that invokes the more recent IBM API QLGCNVCS. This API correctly handles case conversion. YTRNAPIC will ship in Y2SY and the source will be shipped in Y2SYSRC/QCLSRC.

IPv6 Compatibility

The 1E command YEXCFTP has been enhanced for IPv6 support. The INTNETADR parameter on the command can now accept IPv6 IP addresses.

We also enhanced Web Option so that it is IPv6 compatible.

CL ILE EXCUSRPGM Functions

The ability to define CL ILE EXCUSRPGM functions was added to CA 2E. A new HLL value is available for this. This allows you to create CL ILE Programs and Modules. Add these CL Modules to Service Programs in the same way as RPGIV Modules.

LDO Modification

In order to simplify the Product localization process and allow the LDO libraries to be better integrated into the 2E base product, we moved a number of files from the Y2SYVENG library (and other Y2SYVxxx libraries) into Y2SY. This change has no effect on current functionality or performance.

Chapter 2: Fixes to CA 2E r8.5

This chapter describes the fixes included in CA 2E r8.5.

Note: The fixes are listed with problem numbers, but where there is no problem number available, an issue number is listed.

These are published fixes and can be located on CA Support Online. Cases marked as internal are not listed on CA Support Online, so you must contact CA Support directly for more information. The following fixes and behavioral changes are included in this release:

This section contains the following topics:

2E (see page 67)
CA 2E Toolkit (see page 74)
Web Option (see page 75)

2E

C22E 28

When RPGIV code was generated, some of the file-level keywords were being concatenated in some instances, resulting in compile errors.

C22E 137

When an EXCEXTFUN calls RTVOBJ over QRY access path, and Function option is set to CLOSEDOWN *NO, and HLL is RPGIII, on calling the function a second time, get error message RPG1211 I/O operation was applied to closed file.

C22E 148

YCMPMDLOBJ Y2E0311 error when comparing function versions using option 34, then *Select to select a version from the Select Version panel.

C22E 159

We made modifications to the RPGIV Generator to support 10 character file names. Previously, these functions would fail to compile. Note that RPG functions still support up to 8 character file names.

C22E 172

From the Usage display of a locked function, option 14 'GEN batch' still works, but then the YGENSRC job fails with an error since the function is locked. Message Y2V0098'Locked object. Unable to perform request should be sent when option 14 taken since function is locked

C22E 270

Action Diagram Services would sometimes display objects of type FLD instead of type FIL when a '?' was entered for “Find function file name”. Objects of type FIL are now correctly displayed whenever a “?” is entered for the “Find function file name”.

C22E 289

Previously, when generating an RPG function with several Internal Functions where the total number of Long Constants was greater than 99, the function generation or compile would fail. RPG functions can now handle more than 99 Long Constants.

C22E 290

At 8.1 SP1 PTF and 8.1 SP2, when using some custom Header/Footer WINDOW Functions, the "Design exceeds device size limits" message is incorrectly displayed when editing the Device Design and the Function is unable to generate. Modifications have been made so that the "Design exceeds device size limits" message is no longer displayed when editing the Device Design for custom Header/Footer WINDOW Functions that have specified 'Override length' for System Fields (for example, *COMMAND KEY TEXT 1, *COMMAND KEY TEXT 2, etc.).

C22E 291

When a Compile Override was specified for a QRY access path, the override was being generated into the dummy PF, and caused a Compile fail with CPD0043.

C22E 292

When calling an EXCMSG, in the generated code message Y2U0021 is sent if the call ended in error - should be Y2U0032. Also in some instances the message data field was not being defined and caused a compile fail.

C22E 293

When Y2SNMGC is invoked with a blank message ID, it attempts to send YYY0101, but fails and results in 'Text not available for message'. Message YYY0101 is added to Y1USRMSG so that Y2SNMGC can now correctly send it.

C22E 294

When attempting to select a trigger function by using '?' in the 'Trigger function' subfile field on the EDIT TRIGGER FUNCTIONS panel, the 2E model was crashing with an MCH1202 error at statement 2557 in program YMSGDTAE1I.

C22E 295

YDLTJOBLST was not deleting job lists properly.

C22E 299

After entering some text in one of the Find fields and hitting Enter to go to the Action Diagram panel, if you then press F17 to go back to the Action Diagram Services screen then press F3, the Find text was being erased.

C22E 300

When Tab sequence set for fields in a Device Design, the tabbing did not work correctly according to the Tab sequence numbers.

C22E 301

When defining a RNG (Range) condition on a (NBR) number field when the values are negative, e.g. -9 to -1, getting error message Y2V0007 Range lower value exceeds upper value. This has been fixed and ranges with negative values can now be entered

C22E 302

COBOL functions were missing release information in the generated source that RPG functions contained. COBOL function headers now have the same information.

C22E 305

In certain circumstances, the following error would occur when creating a function from a Template function that called another Default Prototype Template function: UNDEFINEDFILE at 2518 in YDUPREFR1I. ONCODE 84.

C22E 306

Problems when a Data structure > 9999 is defined in generated RPGIV code the Z1DBRC DS is only being defined as four digits instead of five.

C22E 308

Cursor-positioning problems for records in folded mode, on the last page of a DSPFIL function. The cursor was not being positioned to the correct record on a return from called EDTRCD function. This issue has been corrected.

C22E 310

RNF5338/RNF7030 compile errors with *CONCAT using RPG ILE, and QRG5258/QRG7030 in RPG III.

C22E 312

When the screen title is removed from a function and it's generated and compiled, the UIM help text compile fails with error CPD5AEA.

C22E 313

Could not change 2E function PGM source member name to a value that contained the _ character.

C22E 314

When using YCMPMDLOBJ to compare objects between a copied model, if the PRTHDRS parameter is set to *YES, User Point headings do not appear in the report.

The fix caters for the fact that YCMPMDLOBJ PRTHDRS(*YES) maybe be operating over two functions with the same surrogate number, i.e. in two different models.

The fix appends the model name, as well as the function surrogate, to "USER:" and "CALC:", AD lines - to force the internal YCMPSRC to print out the "forced to be different" user point titles in the underlying YDOCMDLFUN reports.

C22E 315

When a RTVOBJ function was created over the *Template file, which uses the RTV access path, and has been specified to be the Default prototype function. Any new RTVOBJ functions created that used the PHY access path encountered the error Y2E0616 You cannot attach this function here limitation as if the new RTVOBJ function was being created using the RTV access path.

Modifications were made so that the new RTVOBJ functions, that use the PHY access path, are able to call a DLTOBJ function that was created over the same PHY access path.

C22E 316

PRTFIL AD processing (Print First Page - user point) was erroneously searching for available fields in the 1PG format of the PRTOBJ embedded before the "First Page" This was causing the PRTFIL generation to fail.

C22E 317

Compile error RNF0289 received when using RPGIV EXCUSRSRC that contains long constants. This error was caused by invalid array syntax being generated.

C22E 319

When a 'Subfile control' (CTL) field has been 'Hidden' and includes an 'Override length', if a 'Select operator' of CT (or ST) has been specified, code is generated to call QDCXLATE.

Modifications have been made to the RPG and CBL Generators for DSPFIL and SELRCD functions to use the 'Override length' when the code is being generated to call QDCXLATE in order to avoid a potential MCH3601 runtime error.

C22E 320

When executing the YCVTMDLMSG command from the Services Menu, (option 2) Convert model data menu, (option 1) Convert model messages to database file; if error(s) existed (i.e. parameter mismatch), the YCVTMDLMSG command would hang due to the MCH4404 error. The YCVTMDLMSg command no longer hangs and the Y2E0030 message is correctly sent when error(s) exist.

C22E 322

Using SUBSTR command with Constants sometimes resulted in Compile errors RPG III QRG5225/5258/7030; RPG IV RNF7030/5338/7260; COBOL 4 LBL1447; COBOL ILE 4 LNC2935.

C22E 324

After doing a Field search using the action diagram find services, if you pressed F11 immediately after, to do a Condition search, the Context field from the Field details was not being cleared, resulting in message Y2V0360 Too many find criteria specified. This was corrected.

C22E 326

Occasionally an error was encountered when doing an interactive generation of a 2E function which called an array function-this error was resolved.

C22E 327

Generation Error with the RPGIV function when a field name contains the # character.

C22E 328

COBOL functions were failing to generate due to MCH3601 ('Pointer not set for location referenced.') error. These generation errors would occur for functions generated over a file that included a 'Refers to' relation to a file that included a 'Qualified by' relation. We made modifications to the COBOL Generator to correct this error.

C22E 331

The key order in the YMDLLST10L logical file has been changed so that it includes the object surrogate, thus ensuring that objects which appear on a CM promotion request do so in the 'correct' order (function, display file, help) rather than the old order, which included object type (FUN, DSP, HLP), and therefore was not intuitive, since the display file would appear first.

Note: This change is only visible when using CM.

C22E 335

CA 2E uses generates QDCXLATE calls in 2E functions; for example, DSPFIL & SELRCD to support control field filters. This old IBM API QDCXLATE does not correctly handle case conversion for certain code pages; for example, CCSID 273 German, for upper/lower case Umlaut, so IBM recommends using QLGCNVCS when case is being converted. A new model value, YTRNAPI, makes it possible to use this API, which will resolve this problem. See the section Additional Enhancements and Updates for more details of this model value

C22E 340

If an EDTRCD had a call to a CRTOBJ (or DLTOBJ) in the Action Diagram, and the Create Record function option was N, then a Usage on the CRTOBJ would not show any usages for the EDTRCD function. A fix has been done so that these usages are now picked up. A new Reason code *DSLDBF is now shown for such usages.

See the section Additional Enhancements for more details.

C22E 341

RPGIV EXCUSRSRC functions that included substitution parameters were not being generated correctly when the Specification Type was a lowercase character. We made modifications to the RPGIV Generator to correctly generate substitution parameters when the Specification Type is a lowercase character; 'c' for example.

C22E 343

At V6R1 Create User Space (QUSCRTUS) API has a changed default for the optional Optimum space alignment parameter.This can cause 2E user space processing to fail with MCH5401 sent to QUSCUSAT.

This fix allows 2E to be compatible with V5R3, V5R4 and V6R1.

C22E 344

In a *RTVCND function if the input parameter is not of type STS the generated code fails with compilation error. This fix adds validations to the input parameter fields in "EDIT ACTION - FUNCTION DETAILS" screen so that no other field is specified other than the STS field as an input parameter.

C22E 345

Y2V0241 error would occur when changing 'Ctx' from "CND" to "LCL" while entering "?" for 'Object Name', press ENTER, then press F3=Exit from within the DISPLAY FIELDS screen without making a selection.

C22E 347

We made modifications to the DDS Generators for PRTFIL/PRTOBJ functions for IGC fields. When Keyboard Shift (i.e. K'bd shift) is "O", "A", "W", "E", or "J", an "O" is generated for the Data Type. When Keyboard Shift is blank (i.e. " "), a blank is generated for the Data Type. 'Hidden' internal version of IGC fields always generate a blank (i.e. " ") for the Data Type regardless of the Keyboard Shift value.

C22E 348

RPG functions that included the 'Confirm prompt' and 'Generate error routine' function options were failing with RPG1122 ('Roll-up key stopped work station with no indicator') errors when the ROLLUP key was pressed while the "CONFIRM: _ (Y/N)" prompt was being displayed.

C22E 350

RPGIV Functions that included a Timestamp (TS#) field were failing to compile due to a RNF3316 ('The item has already been defined on a Definition Specification; specification is ignored.') error. These errors would occur when the Action Diagram contained a *MOVE into the DB1 Contect of a Timestamp field.

C22E 356

When using the screen cursor field checking in a 2E CASE statement using '*Cursor field', an error is signaled when the screen field is non-alpha. Processing added to suppress data-type errors where the '*Cursor field' field (473) is used.

16812748 (internal)

When RPGIV code was generated, some of the file-level keywords were being concatenated in some instances, resulting in compile errors. This has been resolved.

17158687 (internal)

An intermittent problem with the subfile display on Edit Functions, where not all the functions over a file were displayed on returning to Edit Functions after editing a function, is resolved.

CA 2E Toolkit

C21E 38

The SPLNBR and RTNSPLNBR parameters on the YCVTSPLF and YRTVSPLFA commands have been updated to allow 6 digits.

C21E 39

YCVTSPLF output (type *PDF) was being truncated in some cases, E.g. when the spooled file width was 198. FONTSIZ parameter has been added to YCVTSPLF so that a smaller font, 6 for example, can be specified (for PDF only) so that the data is not truncated.

C21E 43

If a compilation override is specified for a function or file where a single parameter splits over multiple lines, the Toolkit compile preprocessor YBRTP2R may send error message YYY0112.

In fact, error message YYY0120 should be sent, which warns about unmatched parentheses in the compilation override.

The correct error message is now sent by YBRTP2R and it has been changed to allow for line-spanning compilation overrides.

C21E 46

If a commented-out Procedure specification is present within the first 50 lines of a source member (e.g. P*MyProc), then the 1E compile preprocessor incorrectly treats it as a call to an exit program. The same goes if they use e.g. P*========== as a divider between procedures or anything similar. The compile preprocessor was changed to perform more checking to determine whether a line which begins P* is, in fact, an exit program compile preprocessor directive, by using the following checks:

1. Is the total length of the string (following the P*) less than 21 characters long (the maximum length for a valid qualified name)?
2. If the string includes a '/', does it appear on or before position 11 in the string?
3. If the string does not include a '/', is the string less than 11 characters long?

Only if all the above conditions are met is the P* line treated as a compile preprocessor exit program directive. If any of the above conditions fails, warning message YYY0116 is sent to the job log, but the compilation continues.

16638850 (internal)

An internal limitation in the 1E compile preprocessor caused some compilations to fail. The limitation was due to a string being only 100 characters in length, which is not enough for some very long compilation overrides.

17305323 (internal)

The Toolkit compile preprocessor program YBRTP2R has been changed to allow for more flexibility when the user has specified an X* copy member including blanks.

The syntax for specifying X* copy members has been changed from:

```
[[library-name/]file-name,]member-name|*CMD
```

To:

```
[[library-name[ ]/[ ]]file-name[ ],[ ]]member-name|*CMD
```

Web Option

C2WEB 166

A fix was completed to ensure that, on DBCS systems where an extended DBCS CCSID such as 5026 is used, the equivalent SBCS CCSID is used to retrieve special characters.

C2WEB 194

The Values List processing in Web Option was not correctly processing the last record in the Y2VLLSL1 file if that record is the only VAL record for a LST condition - it should update the condition text to be the same as the LST text.

Note: This was only an issue for customers running Web Option with the YLNGOPT control value set to *MULTIPLE. Other customers would see no problems with displayed text on their web Option pages.

C2WEB 196

The YGENMLS command was not checking whether a 2E model function was locked before generating a skeleton for it. YGENMLS now explicitly checks for both permanent and temporary locks and will stop the skeleton from being generated if the function is locked. Additionally, the generation process has been changed so that if warning message W2G1019 ("Skeleton &1 in &2/&3 already exists...") is sent, the user can take a 'C' to cancel the skeleton generation.

C2WEB 197

This fix addresses 2 problems:

1. The Web Option screen identification processing was not working correctly at runtime; the screens were not being correctly identified (even though the screen identification data appeared to be correct in the YWRKSCRXRF program.
2. When manually identifying a screen, it was not possible to set the Screen Identification Field to *FIELD or *F to indicate that the screen was to be identified by the presence of a field at a particular offset.

C2WEB 198

When '(_v...' screen-value tags are replaced by data from the green screen, both leading and trailing blanks were being removed from the replacement value. Now only trailing blanks are removed from the replacement value when it is inserted in place of the tag.

C2WEB 202

The YCGIPRCS1R service program incorrectly referred to the outdated IBM service program QTCP/QTMHCGI rather than the newer version QHTTPSVR/QZHBCGI.

C2WEB 203

The YWRKW2EVAL command was fixed so that you can now change the YMLSFLR web option value to a valid folder name that exists within the '/' (root) directory.

C2WEB 207

Intermittent HTTP 500 Internal HTTP-server error occurred in certain circumstances. Log files contain: ZSRV_MSG0107: Premature end of script headers: File name is YROUTER.PGM

C2WEB 209

When a JIT Page was generated by the runtime and sent to the browser, the generated HTML source contained an unclosed PAN tag before the command keys section.

C2WEB 211

A bug was discovered in the YW2ESRVR module of the YW2ECTL program, where the Check_Session_Timeout() procedure was being called when a client job had any session status other than 'T' (Timed-out) or 'E' (Ended). This meant that sessions which had a status of 'S' (Pending Timeout) or 'E' (Pending End) would have timeouts sent for them. If there were many jobs in the system, this could quickly overwhelm the Web Option auditor, resulting in a loop, where it repeatedly put entries on the YW2ERQSQ data queue for these jobs. Processing was changed in YW2ESRVR so that the Check_Session_Timeout() procedure was only called for jobs where the session status had a value other than T, E, S, D or X - essentially meaning that only active jobs would be checked.

16426474 (internal)

The Web Option generator was generating duplicate values for some *SFLSEL options (e.g. 2 select options for 4=Delete).

16544873 (internal)

Where Web Option data was held in a physical file with a single keyed logical file (which is the only file accessed via the program), the physical file was changed to have the same key as the logical file, all program references to the logical file were changed to point to the physical file, and the logical file was deleted. The logical files that have been deleted are:

```
YMLSDRF01L (based on YMLSDRFRFP)
YMLSFMT00L (based on YMLSFMTRFP)
YMLSHDR00L (based on YMLSHDRRFP)
YMLSSYN00L (based on YMLSSYNRFP)
YMLSUDT0   (based on YMLSUDTP)
YW2EAUD0   (based on YW2EAUDP)
YW2EUSR00L (based on YW2EUSRRFP)
YW2EVAL00L (based on YW2EVALRFP)
YW2EVAL01L (based on YW2EVALRFP)
```

16638762 (internal)

The Web Option HTTP server configuration that is created through the YCRTW2EHTP command has been changed so that it no longer incorrectly includes the 'YROUTER.PGM' suffix within the <Directory> tag. Whilst the existence of this suffix has not affected the way in which Web Option works, it is technically incorrect.

17151961 (internal)

Processing was added to the YPMTMSGS1R service program to cache retrieved messages to improve performance on subsequent retrievals of the same messages.