CA 2E

Release Notes

Release 8.6.00

CA Technologies Product References

This document references the following CA Technologies products:

- CA 2E
- CA 2E Web Option
- CA 2E Toolkit

Contact CA Technologies

Contact CA Support

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

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

Providing Feedback About Product Documentation

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

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

Contents

Chapter 1: New Features 7

DVD Delivery 7
Documentation Changes 7
Multiple-instance Arrays and the ARR Context 8
*MOVE ARRAY Built-in Function 9
*MOVE ARRAY Parameters 10
*MOVE ARRAY Examples 10
*MOVE ARRAY Usage 13
How to Create a Deployable Web Service Using a Multiple-instance Array 16

Chapter 2: Enhancements to Existing Features 17

Enhanced Array Support 17
Enhanced Array Support Terms 18
Enhanced Array Support Restrictions 18
Performance Considerations for Multiple-Instance Array Parameters 19
Generated Source 20
Enhanced Array Support Usage 20
Updated Splash Screen 25
YCHKFUNPAR (Check Parameter Interfaces) Command 26
Web Option Enhancements 26
YSKLCHK Control Value 27
YWRKW2EVAL Command 27
Generate HTML from DSPMDLUSG and DSPMDLREF 27
Install Web Option 28
COBOL 74 Variant Support 30
LDO Libraries 31

Appendix A: Published Fixes 33

2E 33
CA 2E Toolkit 50
Change Management 51
CA 2E Translator 52
Web Option 52

Chapter 1: New Features

This section contains the following topics:

DVD Delivery (see page 7)
Documentation Changes (see page 7)
Multiple-instance Arrays and the ARR Context (see page 8)
*MOVE ARRAY Built-in Function (see page 9)

DVD Delivery

This product can be installed from directories on your CA Technologies product DVD.

Note: For more information, see the *Installation Guide* on the DVD.

Documentation Changes

Starting with this release, each guide that has updates or changes contains the new Documentation Changes section starting on page four, immediately before the Table of Contents. Each entry has a hyperlink and a quick description of the changes. This enables you to quickly scan and review the documentation changes for each book without having to reread the entire book.

The following guides have the Documentation Changes section for Release 8.6:

- Building Applications
- Command Reference Guide
- Installation Guide
- Toolkit Concepts Guide
- Web Option User Guide

Note: If you do not see the Documentation Changes section in a guide, there are no changes in that guide.

Multiple-instance Arrays and the ARR Context

Arrays are a standard component of the 2E model that you might have utilized in the past. An array is a structure, comprised of multiple fields (array subfields), that has a specified number of elements. Arrays are defined in the *Arrays file.

Prior to release 8.6, there were three ways to use an array:

- Through the use of CHGOBJ, CRTOBJ, DLTOBJ, and RTVOBJ functions built over an array to process data in the array, treating the array as if it were an access path, where each element of the array equates to a record.
- As a structure that can be used to pass parameters to a function.
- With the *CVTVAR built-in function, you can compose a single string from an array structure of multiple fields. You can also decompose a single string into its constituent fields.

  For more information on these functions, see the *Building Applications Guide* and the *Command Reference Guide.*

With the second and third uses, only a single *instance* of the array structure was referred to – the array was being used only as a structure definition, in other words.

With Release 8.6, we increased the ways you can use arrays:

- Treat an array as a multiple *instance* structure. In other words, as a *true* array with multiple elements within the Action Diagram. In this case use the new *MOVE ARRAY function – individual fields in a variety of contexts can be copied into array subfields in a specified instance of the array and vice-versa.
- Pass a multiple-instance array as a parameter to a function. By passing a parameter in this way, you can pass multiple instances of data in or out as a single parameter, in a single call to a function.

Important! As a CA 2E developer, you need to understand the architectural distinction between the two mechanisms to manipulate array data, despite the ability to use a common structural definition:

- Data can exist and be modified in an array by using database functions (Create Object – CRTOBJ, Delete Object –DLTOBJ, Change Object – CHGOBJ, and Retrieve Object – RTVOBJ) based over the *Arrays file. However, this array data cannot be accessed by the *MOVE ARRAY function.
- Conversely, data can exist and be modified in a multiple-instance array parameter (in the PAR context) and in the ARR context by using the *MOVE ARRAY function. However, that array data cannot be accessed by database functions (Create Object – CRTOBJ, Delete Object –DLTOBJ, Change Object – CHGOBJ, and Retrieve Object – RTVOBJ) based over the *Arrays file.

To make this new functionality possible, we created a new array-related context, ARR context. The ARR context is similar to the WRK context, but is used to define a multiple-instance array. Similar to fields in the WRK context, arrays in the ARR context are initialized during program initialization.The ARR context is available in the following circumstances:

- For use with the *MOVE ARRAY built-in function (see page 9).
- When passing a multiple-instance array to a function that has a multiple-instance array parameter. For more details, see Enhanced Array Support (see page 17).

*MOVE ARRAY Built-in Function

The *MOVE ARRAY built-in function allows you to move multiple instances of an array. To specify an array subfield, specify the Array Subfield name, Array Name and Array Index (Element Number).

Use *MOVE ARRAY in the following ways:

- Move the value of one array subfield into another array subfield, either in the same array or a different array.
- Move the value of an array subfield into a field in a non-array context, for example, WRK or LCL.
- Move the value of a field in a non-array context or a constant value or a valid condition into an array subfield.

Note: In all these cases, the special value **ALL* can be used in place of a field name. *ALL pertains to all fields in the specified array index.

For more information and examples, see the*MOVE ARRAY Examples (see page 10).

When using the *MOVE ARRAY function, the array must be a multiple-instance array, using the new ARR context, or a parameter context where the parameter is defined as a multiple-instance array parameter.

*MOVE ARRAY Parameters

The *MOVE ARRAY built-in function uses the following parameters.

Note: The first three parameters define the target field, and the last three parameters define the source field:

***Result**

The target field or the special value *ALL

***Array**

The array in which the target field exists (if it is an array subfield)

***Array index**

The index number which specifies the element of the array in which the target field exists (if it is an array subfield)

***Factor 2**

The source field or the special value *ALL

***Array**

The array in which the source field exists (if it is an array subfield)

***Array index**

The index number that specifies the element of the array where the source field exists (if it is an array subfield)

Within the Action Diagram, the syntax of each group of three fields is as follows (when *Array and *Array index are specified):

```
array-context.array(array-index-context.array-index).array-subfield
```

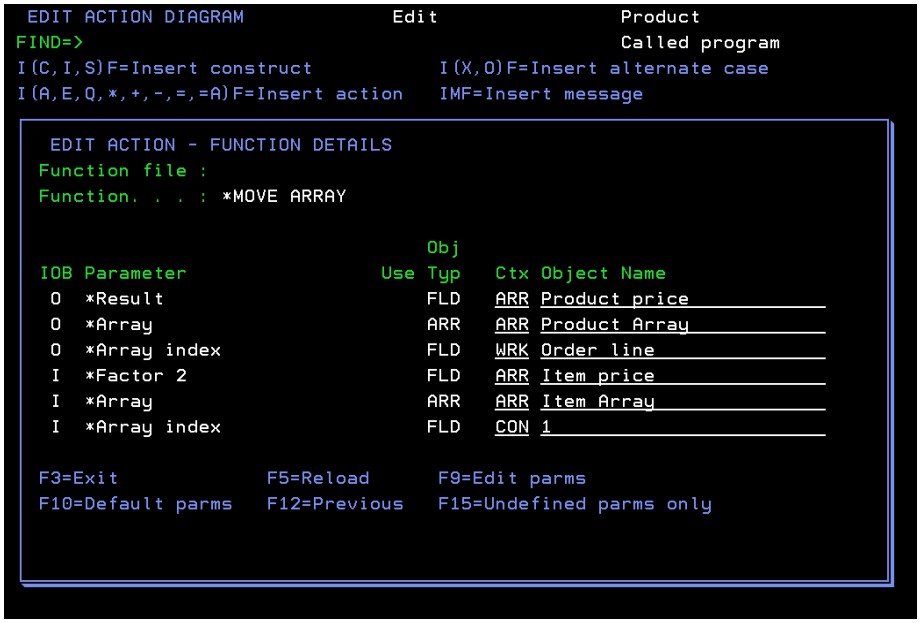
Unlike most function calls, some of the parameters, and their contexts, to the *MOVE ARRAY built-in function can be blank, as shown in the following examples.

*MOVE ARRAY Examples

Use *MOVE ARRAY in the following situations:

- Move an array subfield within a specified element of an array into another array subfield, either in the same array or a different array.

In this example, the Product price subfield in the element of the Product Array, which the current value of the Order line field specifies, is set to the value held in the Item price subfield in the first element of the Item Array:

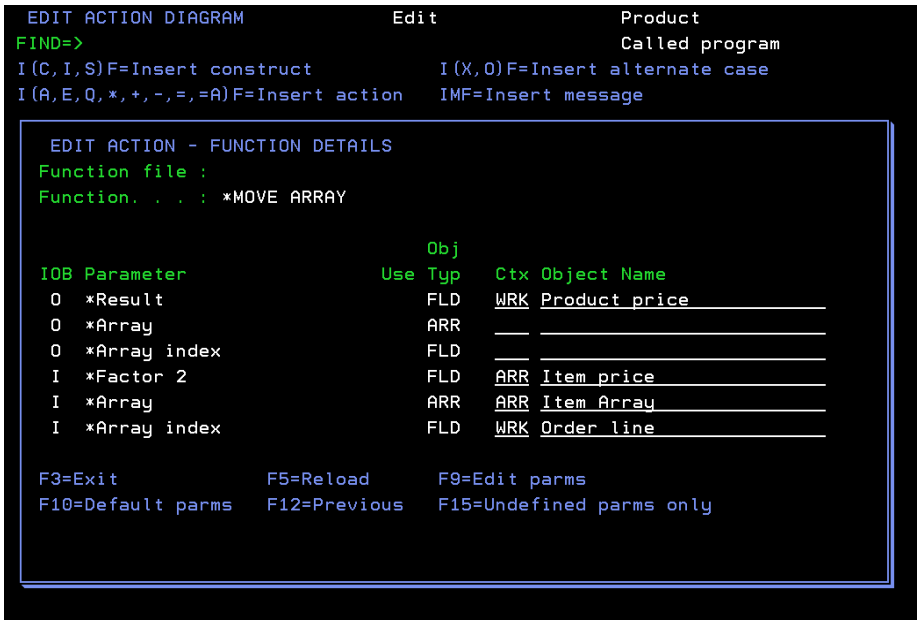


This Action Diagram statement displays as:

```
ARR.Product Array(WRK.Order line).Product price =
ARR.Item Array(CON.1).Item price
```

- Move an array subfield into a field in a non-array context, for example, a field in the WRK context.

  In this example, the Product price field in the WRK context is set to the value held in the Item price subfield in the element of the Item Array specified by the current value of the Order line field:

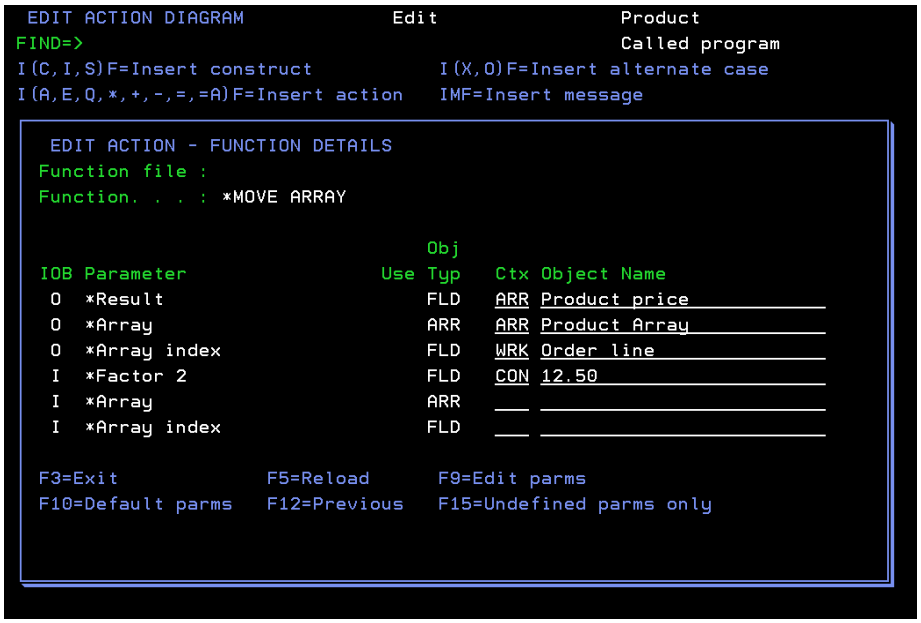


  This Action Diagram statement displays as follows:

```
WRK.Product price = ARR.Item Array(WRK.Order line).Item price
```

- Move a field or value (including conditions and constants) in a non-array context into an array subfield.

  In this example, the Product price subfield in the element of the Product Array specified by the current value of the Order line field is set to a value of 12.50:



  This Action Diagram statement displays as follows:

```
ARR.Product Array(WRK.Order line).Product price = CON.12.50
```

*MOVE ARRAY Usage

If you specify the ARR context, in addition to the code required for the *MOVE ARRAY function, CA 2E generates code to define the required array structures. Therefore, you do not need to define arrays in the ARR context. You can define these arrays automatically by using the *MOVE ARRAY function. When you generate a *MOVE ARRAY statement, CA 2E generates additional code that checks for array indexing errors.

Multiple instance arrays in 2E are 1-based, the first element in an array is element 1. You cannot specify a constant value (CON context) less than 1 or greater than the maximum number of elements in the array in the Action Diagram Editor. However, if you specify a runtime field value less than 1 or greater than the maximum number of elements in the array in the *Array index field, you receive an error.

If an array indexing error occurs, the PGM.*Return code field is set to a condition value of '*Array index error', which corresponds to the Y2U0068 message in the Y2USRMSG message file. This error can be monitored for as in the following example:

```
WRK.Product price = ARR.Item Array(WRK.Order line).Item price
.-CASE
¦-PGM.*Return code is *Array index error
¦ <-- *QUIT
'-ENDCASE
```

The *MOVE ARRAY function is only valid in the Execute External Function (EXCEXTFUN) function type.

If *Result or *Factor 2 is not an array subfield, the following restrictions apply to that field:

- The related *Array and *Array index fields must be blank.
- The context specified must be one that would be valid in a *MOVE statement.
  - The valid contexts for *Result are PGM, LCL, WRK and NLL (including any valid parameter context, if the function has an appropriate output parameter).
  - The valid contexts for *Factor 2 are PGM, JOB, LCL, WRK, CND and CON. This context includes any valid parameter context, if the function has an appropriate input parameter.

If *Result or *Factor 2 is an array subfield then the following restrictions apply to that field.

- The related *Array and *Array index fields cannot be blank.
  - The *Array index can be a positive integer constant with a value less than or equal to the number of elements in the array, or refer to a numeric variable with no decimal places.
  - The *Array index cannot itself be a subfield of a multiple-instance array.

- The context specified for *Result or *Factor 2, and the related *Array, must be one of the following contexts:

  ARR

  Valid for both *Result and *Factor 2.

  PAR

  Valid for *Result if the specified field exists as an Output, Both or Neither parameter field on a multiple-instance array parameter. Valid for *Factor 2 if the specified field exists as an Input, Both or Neither parameter field on a multiple-instance array parameter.

  PR*n*

  n is an integer 1–9.

  Same validity as PAR, but used where the function has duplicate parameters.

If *ALL is specified for either *Result or *Factor 2, the following restrictions apply:

- Certain context-specific restrictions apply when you specify here *ALL is specified:
  - If you specify *ALL for *Result, also specify *Factor 2 and vice-versa.
  - If you specify *ALL for *Result, you cannot specify CND and CON as the context for *Factor 2.
  - If you specify *ALL for *Factor 2, you cannot specify NLL as the context for *Result.
- Code is only generated to move a field if all the following conditions apply:
  - The field exists in both the *Factor 2 context and the *Result context
  - If the *Factor 2 context is a parameter context, the field must have a usage of O, B or N
  - If the *Result context is a parameter context, the field must have a usage of I, B or N
- Any target fields (or target array subfields) that exist in the *Result context, but do not exist in the *Factor 2 context, are not changed.

Note: The same field type validation rules apply to *MOVE ARRAY as to *MOVE, in terms of moving numeric fields to non-numeric fields.

From the main Action Diagram Editor screen, you can use the new subfile option I=M to insert and prompt the *MOVE ARRAY built-in function.

How to Create a Deployable Web Service Using a Multiple-instance Array

We created a a working scenario to explain how an experienced CA 2E Application Developer can use CA 2E web service support with enhanced array support to make an invocation and retrieve an order. This process includes the order header and multiple order detail lines.

For the full scenario, see the Appendix "How to Create a Deployable Web Service Using a Multiple-instance Array" in the *Building Applications Guide*.

Chapter 2: Enhancements to Existing Features

This section contains the following topics:

Enhanced Array Support (see page 17)
Updated Splash Screen (see page 25)
YCHKFUNPAR (Check Parameter Interfaces) Command (see page 26)
Web Option Enhancements (see page 26)
COBOL 74 Variant Support (see page 30)
LDO Libraries (see page 31)

Enhanced Array Support

Before Release 8.6, you could define a parameter to a function to be passed as a FLD, RCD, or KEY.

- **FLD–**Each field specified as a parameter on the parameter details display is passed as an individual parameter.
- **KEY–**A single parameter, where the length is derived from the keys of the specified access path or array, is passed. An externally defined data structure is used to define the parameter.
- **RCD–**A single parameter, where the length is derived from the specified access path or array format, is passed. The parameter contains all the fields which are individually specified as parameters using the parameter details display. An externally defined data structure is used to define the parameter.

Release 8.6 gives you the ability to pass certain parameters as an array.

By passing a parameter as an array, multiple instances of data can be passed in or out, in a single call to a function. For example, if a customer record structure is defined in the Customer array on the *Arrays file, that array can be used to define a parameter to an Execute External Function (EXCEXTFUN) or Execute User Program (EXCUSRPGM) being passed as RCD (ARRAY), any amount of customer records can be passed in that one parameter, in one single function call.

The Edit Function Parameters Panel and the Edit Function Parameter Details Panel were updated to accommodate this enhancement.

Enhanced Array Support Terms

To assist you with understanding this enhancement, we use two new descriptive terms throughout the CA 2E documentation:

Multiple-instance array parameter

Describes when a parameter is passed as an array (when the "Pass as Array" flag is set to 'Y'). The parameter contains multiple instances of data, where each instance contains all the fields which are individually specified as parameters using the parameter details display.

Single-instance array parameter

Describes when a parameter defined using an array is not passed as an array (when the *Pass as Array* flag is not available or is set to blank). The parameter contains all the fields which are individually specified as parameters using the parameter details display.

Enhanced Array Support Restrictions

This new functionality has a number of fundamental restrictions:

- Only functions of type Execute External Function (EXCEXTFUN) and Execute User Program (EXCUSRPGM) allow parameters to be passed as array.
- Parameters can only be passed as array when the parameter structure is defined using an array based over the *Arrays file.
- Parameters can only be passed as array when they are being passed as RCD or KEY.
- No fields can be dropped on a parameter being passed as an array.
- Does not allow a multiple-instance array parameter in a function call, in both ARR and PAR context, except when calling an EXCEXTFUN or EXCUSRPGM which has multiple-instance array parameter. Additionally, the call must be from the top-level action diagram of an EXCEXTFUN.
- The Submit job (SBMJOB) feature and Y2CALL command do not support function calls that contain multiple-instance array parameters.

When working with two functions, function A and function B, for example, you can model in the action diagram of function A a call to function B, where B has a parameter interface passed as an array. In this case these additional restrictions apply:

- Function A must be of type EXCEXTFUN, and function B must be of type EXCEXTFUN or EXCUSRPGM.
- The parameter context must be PAR or ARR and the array name must exactly match on the parameter definition of A and B.
- If a parameter is passed as an array on A it must be passed as an array on B.
- The parameter must be passed as RCD on both A and B, or KEY on both A and B.

- Though the usages of the subfields on a parameter passed as an array can be mixed, the usages must be compatible, such that the calling function can call the called function.
- In general, the 2E tool prevents the use of modeling scenarios that cannot be successfully generated.

Note: For more details, see the sections for Edit Function Parameters Panel (see page 21) and Edit Function Parameter Details Panel (see page 22), or see the chapter "Defining Function Parameters" in the *Building Applications* guide.

Performance Considerations for Multiple-Instance Array Parameters

When using multiple-instance array parameters (MIAPs), the following performance-related considerations apply:

General performance considerations when using MIAPs

Even though a MIAP can have many instances, only a single pointer is passed by the operating system to the program, as is the case with a non-MIAP parameter. Therefore, in terms purely of the parameter being passed as a *normal* parameter or as a MIAP, there is no additional performance *hit* to using MIAPs.

Performance considerations in programs that use MIAPs

Any *Neither* parameters that are passed to a function are explicitly initialized in the ZZINIT subroutine in that function. In the case of a MIAP parameter containing *Neither* subfields, this initialization occurs for every field, in every element in the MIAP.

Performance considerations in programs that call other programs that use MIAPs

When one program calls another and passes a structure parameter (RCD or KEY), CA 2E generates code in the calling program to initialize the intermediate structures used to pass the parameters to the called program, to load those structures from the variables specified in the Action Diagram and to unload those structures into the return variables. This code is generated whether the parameter is defined as a MIAP or as a *normal* parameter.

When the parameter is defined as a MIAPs

- If all the MIAP subfields are defined with the same usage (I, O, B or N), then the generator loads the entire array structure into and out of the intermediate structure using a single MOVE. By contrast, if the MIAP subfields have different usages, each field must be moved separately. Since the MOVE is repeated for every element of the MIAP, this can affect your performance. Additionally, the amount of code generated for a MIAP with varying-usage subfields can be significantly greater.

- As with fields passed as parameters in non-MIAPs, MIAP subfields with a usage of *Neither* are explicitly initialized prior to the call – this occurs even if all the subfields have a usage of *Neither*.
- As with fields passed as parameters in non-MIAPs, if any of the MIAP subfields are ISO-type fields (DT#, TM# or TS#), code is automatically generated to explicitly initialize them, whether or not they are passed with the same usage as all other subfields within the MIAP.

To ensure the best possible performance when using MIAPs, follow these guidelines as closely as you can

1. All subfields within a MIAP should be defined with the same usage (I, O or B).
2. MIAP subfields with a usage of Neither should be avoided.
3. ISO-type MIAP subfields should be avoided.

Generated Source

Using either multiple-instance array parameters or the related *MOVE ARRAY built-in function results in additional fields and structures (and the code to initialize and process them) to be generated in your source. These fields and structures can significantly increase the size of the source member. Therefore you can control the level of in-line source commenting through the YGENCMT model value, as follows:

***STD**

A single comment line is generated for each multiple-instance array definition, control structure definition and initialization

***ALL**

A comment line is generated for each multiple-instance array subfield definition, control structure subfield definition and initialization

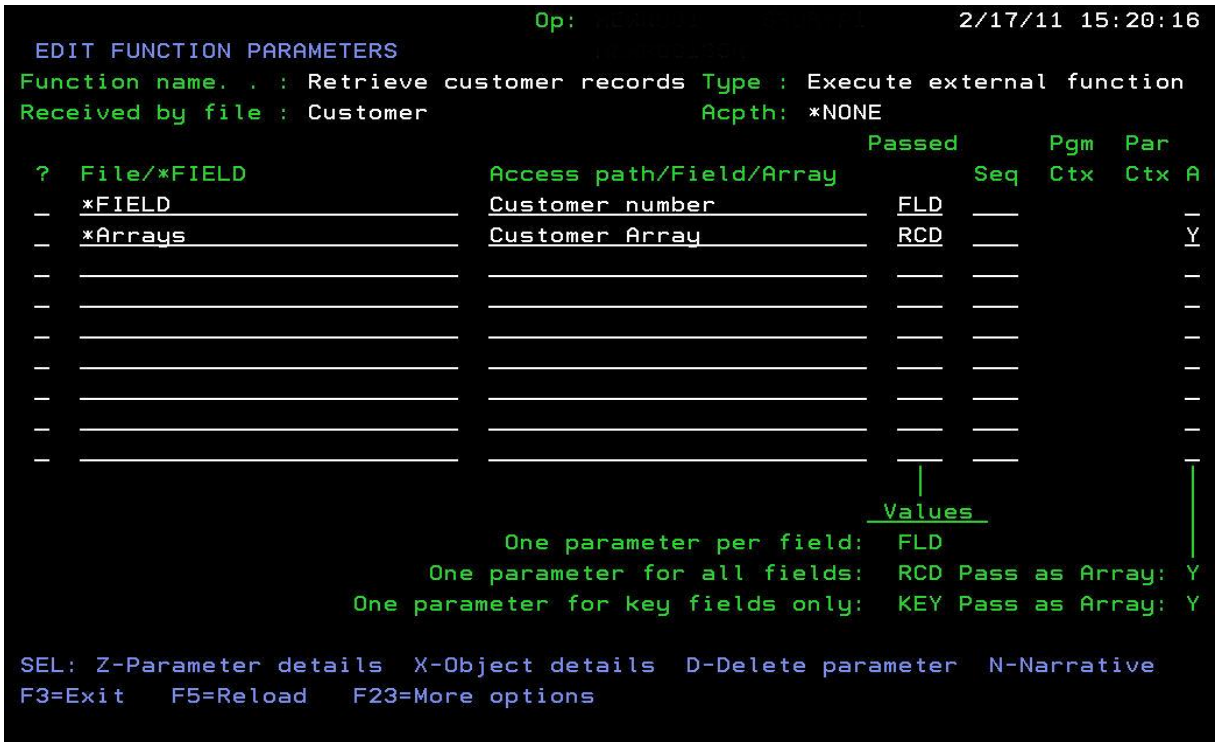
Note: This is the only difference between YGENCMT(*STD) and YGENCMT(*ALL); switching between these two values does not affect any other comment generation within your source code.

Enhanced Array Support Usage

The following sections show how you can use the Enhanced Array Support in CA 2E panels.

Edit Function Parameters Panel

The new functionality allows you to specify, in the Edit Function Parameters panel, that a parameter should be passed as an array, rather than passed as a single-instance structure.



This is done using the new ‘A’ (Pass as Array) field as follows:

- Enter Y when the parameter should be passed as array.
- Leave this field blank when the parameter is a single-instance structure.

The following situations apply when using the 'A' (Pass as Array) field:

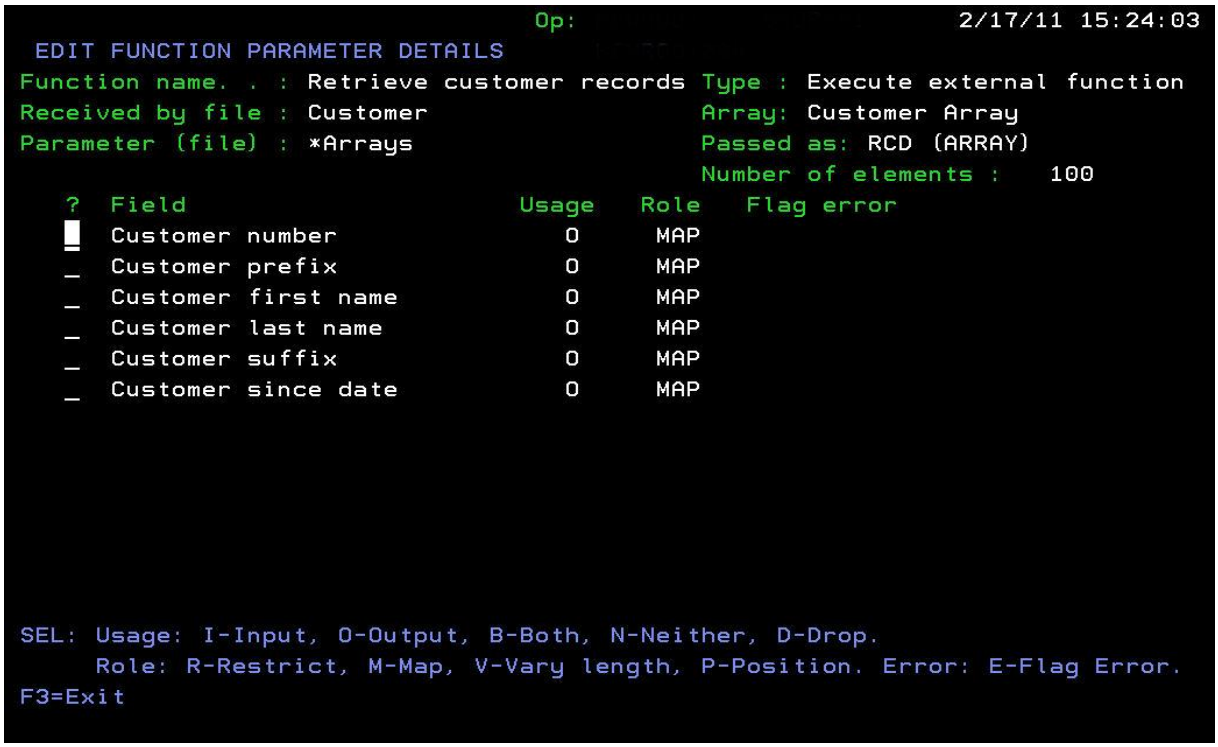
- If the function is not an EXCEXTFUN or an EXCUSRPGM, the field is not available on this panel.
- Specifying 'Y' is only valid when the parameter is an array based on the *Arrays file, and only when the parameter is passed as RCD or KEY.

Edit Function Parameter Details Panel

For EXCEXTFUN and EXCUSRPGM only, there are two new values for the *Passed as* field:

- RCD(ARRAY) = Parameter is defined using RCD structure and passed as array.
- KEY(ARRAY) = Parameter is defined using KEY structure and passed as array.

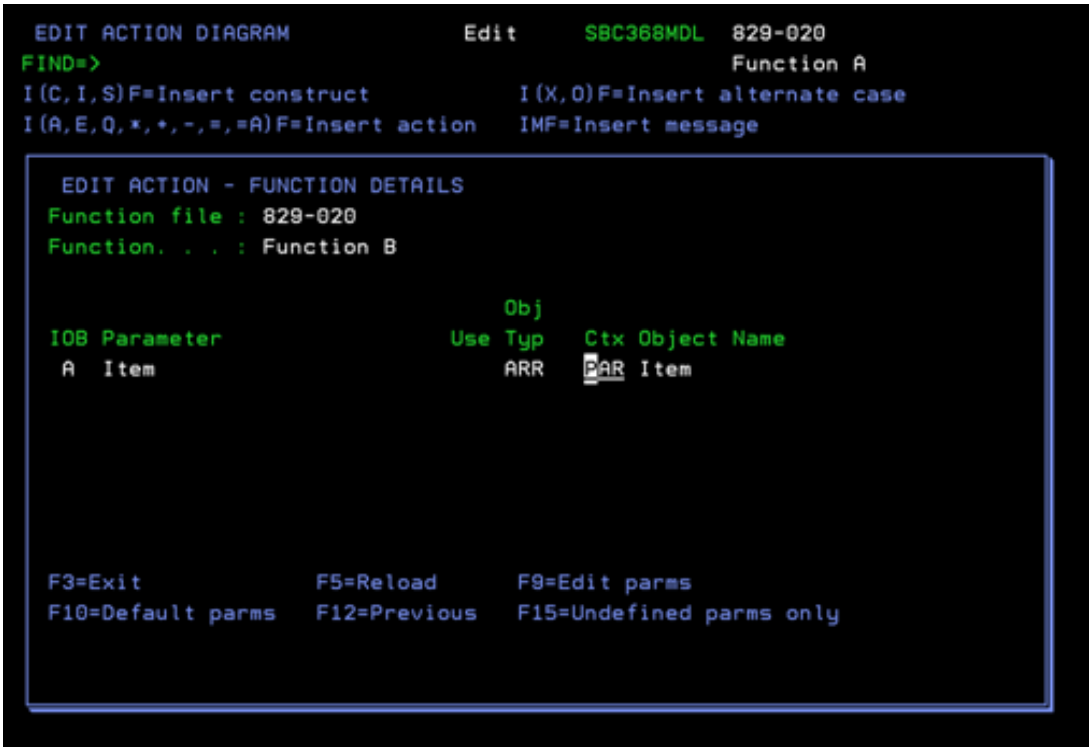
When *Passed as* has a value of RCD (ARRAY) or KEY (ARRAY), the new *Number of elements* field displays the number of elements defined for the array being passed. You can view and modify the array definition by visiting the *Arrays file. If you use option D to drop any fields, an error message displays.



Edit Action Diagram Panel

When a parameter is not passed as an array the behavior of the Edit Action Diagram panel remains the same as previous versions of CA 2E. However, where a parameter is being passed as an array, there is a new single subfile line that indicates an array being passed.

Note: If the called function's parameter interface is modified to toggle the parameter *Passed as Array* field from *Y* to blank, the behavior of the EDIT ACTION – FUNCTION DETAILS changes accordingly to match.



IOB

A, indicating an Array.

Obj Typ

ARR, indicating array.

Ctx

Only PAR or ARR is allowed.

Note: This is an input capable field. ARR is always valid as a choice when the called function's parameter is passed as an array. However, if you select PAR, but the definition of the array on the calling function is incompatible with the definition of the array on the called function, a warning is sent (after PAR is selected) and the change to PAR cannot be saved/completed.

Object Name

Item, indicating the name of the array.

Note: This field is output only.

The array's subfields and their usages are NOT shown on this panel. But you could examine Function B's parameter interface (and detailed usage) via F9=Edit Parms.

This screen shows what the function call will look like for passing an array called *Item*:

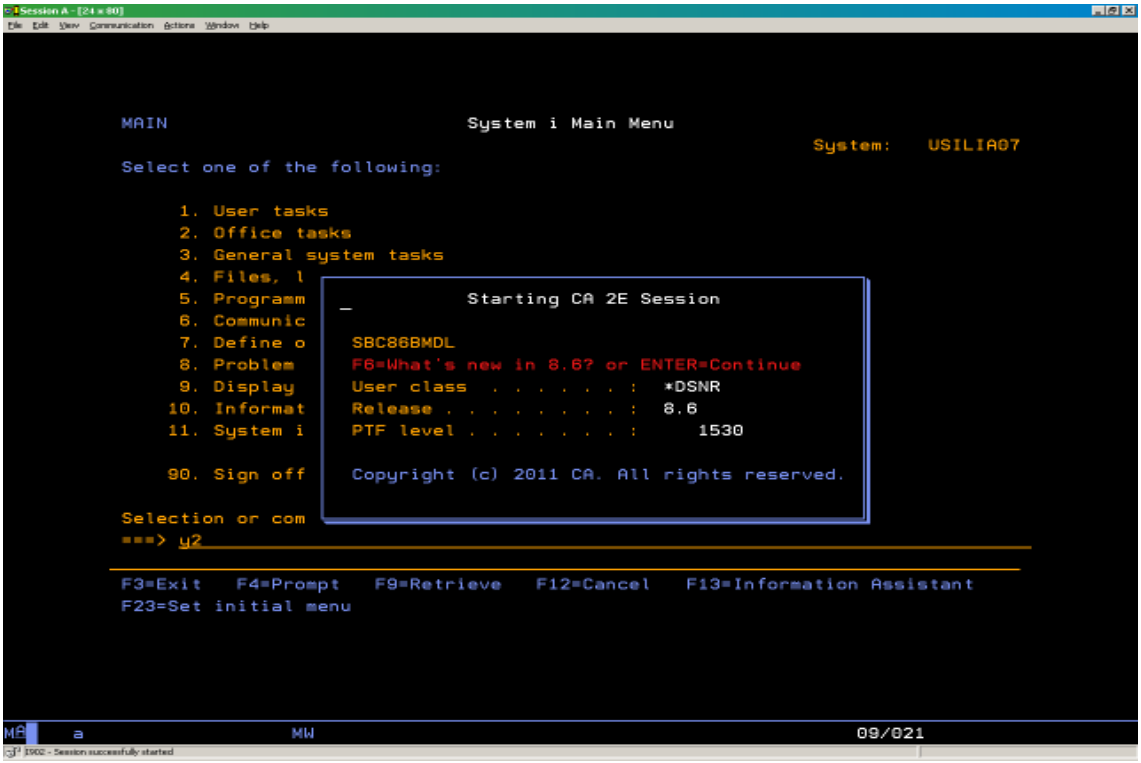
When a parameter is being passed as an array,the context defaults to ARR if F10=default parms is used. If the context is changed to PAR, the panel validates that the selected array is available in the PAR context and that all array subfields have compatible usages.

Note: MIAPs do not automatically get defaulted to PAR/PRn context, even when they are available. Your choices for populating MIAP parameters are:

- Manually specify ARR context which is always valid.
- Manually specify PAR/PRn contexts if available.
- Prompt for available contexts.
- Use F10 to default MIAPs to use ARR context.

Updated Splash Screen

We updated the CA 2E *splash screen* – the window that appears briefly when a 2E model is being loaded. When entering the model for editing, this screen, which used to briefly appear and disappear, now remains until you dismiss it. An additional line of text appears on the screen immediately following the model name:



You can press F6 to display the fixes included in this release, or you can press Enter to continue and load as normal.

By default, the message in this example appears. However, this new functionality is controlled by the YEDTMDLSYA data area, which exists in the Y2SY 2E product library. If you want to disable the functionality, change the YEDTMDLSYA data area to a value of *0* (zero), as in the following example:

```
CHGDTAARA DTAARA(Y2SY/YEDTMDLSYA) VALUE('0')
```

If you make this change processing will function as it did prior to Release 8.6, and the splash screen will again display only briefly when the model loads.

Note: To ensure that developers have the chance to view details of any new fixes to the 2E model environment, with each new release this data area will be reset to '1' during the installation.

YCHKFUNPAR (Check Parameter Interfaces) Command

Use the YCHKFUNPAR (Check Parameter Interfaces) command to identify any functions that have duplicate parameter fields when the *Duplicate parameters* function option is set to *N*. Prior to Version 8.6, it was possible to enter duplicate parameters even when the *Duplicate parameters* function option was set to *N*. This command allows you to analyze an entire model and identify functions that violate the *Duplicate Parameters* restriction and exception. Executing the command produces a joblog that contains error messages identifying problematic functions.

Important! After gathering the results from the YCHKFUNPAR command, it is your responsibility to rectify any problems by modifying, as necessary, the parameter interfaces.

The command searches for two specific parameter interface problems:

- Invalid Duplicate Parameter field:

  For a given function (where function option *Duplicate parameters* is set to N) each parameter field must be unique, regardless of usage. The only exception is that a field can appear once for Input and once for Output.

  Sends error message (Y2V0719) to joblog.

  Note: If a function has *Duplicate parameters* function option changed from Y to N, previously valid duplicate fields may now become invalid (however no warning or error will be sent). It is your responsibility to revisit the parameter interface and ensure that any invalid duplicate parameters are modified or removed accordingly. The YCHKFUNPAR can be used to identify such invalid duplicate parameters.

- Non-unique Parameter SEQ:

  When defining parameters on the EDIT FUNCTION PARAMETERS panel, SEQ does not need to be unique. This has no adverse impact when function option Duplicate Parameters = *N*. However, when function option Duplicate Parameters = *Y*, and two parameters are specified with the same sequence number, the same duplicate parameter context is used for both parameters. Therefore in the AD, a field can appear in a given duplicate parameters context more than once.

  Sends error message (Y2V0720) to joblog.

For more information, see the published fix for C22E 56 (see page 33), the chapter "Modifying Function Parameters" *Building Applications Guide*, and the chapter "Commands (YADDMDLLE - YCHKMDLOBJ)" in the *Command Reference Guide*.

Web Option Enhancements

We enhanced the following Web Option items.

YSKLCHK Control Value

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

Note: This was in response to the issue C2WEB 215 (see page 52).

For more information, see the *Web Option User Guide*.

YWRKW2EVAL Command

In previous releases of Web Option, the YWRKW2EVAL command did not allow users to easily discern which control values were used by the Web Option runtime. These are a combination of the records in the product library and in the environment data library, but the YWRKW2EVAL command only displayed one set at a time.

We enhanced the YWRKW2EVAL command so that, when a control value is changed, if it was retrieved from the environment data library, that value is updated. If it was retrieved from the product library, the new value is copied to the environment data library. While being displayed, control values from the product library are shown in green, and the control values from the environment data library are shown in white. We added a new delete option that allows a user to delete a control value from the environment data library (which would reinstate the control value from the product library).

Generate HTML from DSPMDLUSG and DSPMDLREF

We added the option *40=Generate HTML* to the Display Model Usages (DSPMDLUSG) and Display Model References (DSPMDLREF) panels. When using Impact Analysis, this new option allows you to generate a skeleton for identified functions.

Install Web Option

For Release 8.6, we improved the way you upgrade and install the Web Option. This first procedure is for all users. The secondary procedure you use to complete your installation varies depending if you are installing the Web Option for the first time, or if you are upgrading from a previous release of the Web Option.

Note: Upgrading directly from releases older than r8.5 is not supported. If you are running a version older than r8.5, you must first upgrade to r8.5 before upgrading to Release 8.6. For more information on upgrading to r8.5, see the CA 2E r8.5 documentation at http://ca.com/support.

To install the Web Option libraries

1. Sign on to the IBM i with a user profile that has Special Authority *SECADM or *SECOFR.

   Note: This user profile must also have *IOSYSCFG, *ALLOBJ, *SAVSYS, *JOBCTL, and *SPLCTL special authorities.

   Your library list should include the following libraries (or their renamed or merged equivalents), but the order is not important:

   - Y1SY
   - Y1SYVENG
   - QTEMP

2. Ensure no users are logged in to any previous release of Web Option you might have installed on your IBM i server.

   If you do have a previous release of Web Option, rename the libraries (for example, to Y2WEBSAV and Y2WEBVSAV). Change the YW2ELDORFA data area in the product library to point to the renamed LDO library. Ensure these renamed libraries are in your library list:

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

3. Restore the shipped Web Option libraries using the Restore Library command RSTLIB:

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

   Note: If necessary, you can rename these libraries when you restore them. Do this with either the RSTLIB parameter to the RSTLIB command, or rename the libraries after you restore them. However, if you do rename these libraries, you must change the YW2ELDORFA data area in the (renamed) Y2WEB library to point to the (renamed) Y2WEBVENG library. In the following steps, we assume that you did not rename these libraries, so the steps refer to Y2WEB and Y2WEBVENG.

4. Add the restored Web Option libraries to your library list:

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

The procedure you follow to complete your installation vary depending on if you are installing Web Option for the first time, or if you are upgrading from a previous version of Web Option. Go to the relevant section for your situation:

- Install Web Option for the First Time (see page 29)
- Upgrade Web Option from r8.5 (see page 30)

For more information, see the Installation section of the *Web Option User Guide*.

Install Web Option for the First Time

If you are upgrading a previous release of Web Option, skip this procedure and go to Upgrade Web Option from r8.5 (see page 30).

To install Web Option for the first time

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

   This creates all the required Web Option objects on your system, such as user profiles and job descriptions.

2. Create an initial Web Option environment data library.

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

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

   As this command processes, you are given the opportunity to change the Web Option environment control values and add a job schedule entry.

4. Change the YDBFCCS control value to the value of Code page from DSPSYSVAL QCHRID when you are prompted.

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

   Note: You must run this command interactively.

Upgrade Web Option from r8.5

If you are installing Web Option for the first time, skip this section and go back to Install Web Option for the First Time (see page 29).

Note: With Release 8.6, there is a a new command, YUPGW2EENV (Upgrade Web Option environment) that replaces the YINZW2E command. To upgrade, you will use this new YUPGW2EENV command.

To upgrade from a previous release of Web Option

- Specify the name of your existing Web Option environment data library, the name of the restored Release 8.6 Web Option product library, and the LDO library. If you changed the names of Y2WEB and Y2WEBVENG, remember to replace them accordingly in this example:

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

  Note: This command must be run interactively.

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

COBOL 74 Variant Support

We determined there is little to no use of the COBOL 74 variant. Starting with this release, we no longer support the COBOL 74 variant within the CA 2E COBOL generator. However, we continue to test and support the COBOL 85 variant.

LDO Libraries

The CA 2E non-English Y2SYVxxx LDO libraries are now shipped with fewer objects than in previous releases. The majority of the data files that we previously shipped in both the LDO libraries and the Y2SY product library are now only shipped in Y2SY. The foreign-language text that we previously retrieved from fields in those files is now retrieved from the related message identifiers in the Y2ALCMSG message file in the LDO library. This change enables us to ship smaller LDO libraries that do not require additional translation processing.

The YAPYTRNMDL and YAPYTRNSYS commands are still shipped for internal use. You do not need to used them to perform any end-user translation, and they should not be used. To generate your functions in a different language or to use your CA 2E model with a different display language (where available), add the relevant non-English LDO library to your library list above the English LDO library (Y2SYVENG).

Appendix A: Published Fixes

We corrected, modified, or fixed all of the issues described in this Appendix. Published fixes are also available from Published Solutions at http://ca.com/support.

Note: Fixes with an eight-digit number are internal to CA so are not posted to CA Support Online.

2E

We completed the following fixes, corrections, or updates for CA 2E:

C22E 41

RP4 functions with embedded EXCUSRSRC functions that define compile-time arrays were failing to compile due to the errors RNF8039 *Extra entries on last compile-time table or array record* and RNF8041 *Too many records provided to initialize array or table*. Also, the EXCUSRSRC compile-time array was being generated in the wrong section of the source. This problem occurred when the external function contained a long constant which would then cause the @CN array to also be generated.

C22E 56

There was an issue with the Duplicate Parameters functionality that allowed duplicate parameters to be added even when the Duplicate Parameters function option was set to *N*. You can no longer enter duplicate parameters (except where a field is defined once for input and once for output) when the Duplicate Parameters function option is set to *N*.

Notes:

- Although this fix correctly enforces the duplicate parameter and restriction when modifying parameter fields, you should be aware of the following scenario:
- If a function has the *Duplicate parameters* function option changed from *Y* to *N*, previously valid duplicate fields might become invalid; however, no warning or error is sent. It is your responsibility to revisit the parameter interface and ensure that any invalid duplicate parameters are modified or removed accordingly.

For more information on this process, see the *Building Applications Guide*.

The new command YCHKFUNPAR (see page 26) (Check parameter interfaces) can be used to analyze an entire model and identify functions that violate the duplicate parameter restriction and exception.

C22E 99

SFLRCDNBR could only hold a maximum value of 999, so paging down over 66 times resulted in the error MCH1210 Receiver value too small to hold result.We made modifications so that SFLRCDNBR now holds a maximum value of 9999. This increases the number of subfile records that can be stored in the Display All Functions, Display All Access Paths, and Edit Array Details panels.

C22E 172

With r8.5, we made a change that disallowed Locked Objects from being added to a Model Generation List. When doing Impact Analysis Usages (say for example you have changed a file and want to generate all associated objects) and you enter 14 against all the objects, the processing stops when a locked object is reached.

Processing changed in several programs to check whether an object which is being processed has a permanent lock. If so, an error is thrown and the relevant subfile selector is displayed in reverse-image. Entries on the list prior to the entry in error are processed and entries on the list subsequent to the entry in error are displayed with the subfile selector still displayed - as soon as the subfile selector for the entry in error is cleared and Enter is pressed, subsequent entries are processed.

C22E 239

Due to an IBM limitation, a PL/I run-unit has a set maximum of 16-megabytes of storage. At times, users would encounter storage limitation errors while editing large 2E Models. For example you might see this error:

```
Tried to go larger than storage limit for object PE$BASHEAP. STORAGE condition raised
at nnnn in YEXPUSGR1I. ONCODE 8085.
```

C22E 253

In the EDIT SCREEN FORMAT DETAILS panel, the Command key for SFLFOLD field could not contain 05 . 05 was not included within the list of allowed values regardless of the value stated for the *Reset LST Condition. You can now use 05; however, only if CF05 is not currently stated for the *Reset LST Condition.

C22E 310

In certain scenarios, when *SUBSTRING or *CONCAT functions take arguments in the CON context or in the PAR context in an internal function, where the external function calls the internal function passing a parameter argument in the CON context and when the string used was longer than five-characters, compilation failed with errors RNF5338/RNF7030 (RPG IV) and QRG5258/QRG7030 (RPG III).

C22E 351

At times, the generated DDS included the CHECK(AB) keyword and value even though the corresponding screen field was specified with *Allow zero/Allow blank* set to blank; ' ' in other words. When *Allow zero/Allow blank* was set to blank, the intended functionality was that zeroes/blanks are not allowed and *Allow zero/Allow blank* was set to *Y*, the intended functionality was that zeroes/blanks are allowed.

We modified the DDS Generator to overcome this reported problem for fields that reside within the RCD, DTL, 2ND, or 3RD screen formats. Fields that reside within the CTL format will continue to allow zeroes/blanks for usability, otherwise, an error (Reverse Image for example) occurs when you press ENTER in this scenario. Typically, the DDS Generator generates CHECK(AB) (CHECK keyword with the *Allow Blanks* value) to enable this functionality. However, at times validation logic is generated within the corresponding RPG/CBL program.

Notes:

- Fields that have specified *Modulus 10/11* or *Valid system name* continue to generate the CHECK(AB) keyword/value into the DDS regardless of the *Allow zero/Allow blank* value.
- Value mapping fields (for example, TM#, TS#, DT#, DTE, D8#, STS with *Translate cnd values*) continue to not generate the CHECK(AB) keyword/value into the DDS regardless of the *Allow zero/Allow blank* value.

C22E 358

An RPG SQL SQL0206 compile error was occurring when multiple internal functions existed that were using access paths over the same physical file, and when the access paths were defined with T-TABLE (Data access method). The RPG Generators now use the correct RPGPFX.

C22E 361

When generating an RPGIV EDTRCD function that used a 1000-character field, and model value YNLLUPD was set to *YES, the compile was failing with the following error:

```
 *RNF5272  30  Either a valid Operation Code or Conditioning-Indicator entry must be
specified.
```

This error no longer occurs with the modifications we made to the RPGIV Generator.

C22E 362

The submission of access path generation from a model list using option 14 was not submitting a job to the job list, although it did send a message indicating that it had done so. Option 14 now functions correctly for access paths.

C22E 363

When a Prototype function had several versions, the current version of the *Default prototype function* was not being used when a new function was being created. There is now included logic to check that the *Default prototype function* is current when a new function is being created.

C22E 365

When using the Language Dependent Objects (LDO) libraries for non-English languages, some or all of the Action Diagram User Points and text should be in that particular language. With r8.5, all of the Action Diagram text appeared in English for all LDO libraries. A change was made to retrieve the Action Diagram text from LDO messages rather than the file, but many of the necessary messages were missing.

C22E 366

While editing a model list, the position by name feature of the *Edit Model Object List* function is case sensitive. The ALTSEQ keyword was missing, so we replaced it.

C22E 367

In a *Change Controlled* Model, the *IMCOSY1 was called recursively* error was occurring. Due to added *Template functionality, the error was seen when attempting to access the action diagram of a Current *Production* version of a non-Checked Out function while specifying *Make this version current* set to *Y* within the Check Out Model Object display.

C22E 368

On a STS FLD, if a LST CND contained more than 100 VALs, the Action Diagram line CASE statement for a STS field referencing *ALL values only generates 100 VAL CNDs in the source for RPG IV. This fix increases the RPG IV limit to 500.

C22E 370

We made previous modifications to validate that the Input Parameter to the *RTVCND (Retrieve Condition) built-in function is a Status (STS) field. We made additional modifications to allow a Referenced (REF) Status (STS) field to be a valid Input Parameter to the *RTVCND built-in function.

C22E 371

When a user edited a function and, from the Action Diagram Editor, accessed the Display Usages (by taking option U against a called function), and then took option 14 (Submit to compile) against the editing function, the compile was not submitted and error message Y2V0099 was sent.

This fix prevents objects that have a permanent lock being added to a job list (Option 14) from Display Model Usages/References.

C22E 372

At times, RPGIV Functions were failing to compile with error RNF4005. These errors occurred when the Access Path Format Name was 6 characters (or less).

C22E 373

In YWRKMDLLST, option 2 (Edit Model Object List), followed by option 38 to run YCHKFUNACT on a PRTFIL, followed by option 10 to edit the PRTFIL's action diagram caused the error message CPF5032 when attempting to exit and save from the action diagram editor. This fix prevents this record locking error.

C22E 375

The RPG and RP4 generators did not generate a CLOSE statement for the display file in subroutine ZXEXPG for a function set as a window when the program had the *Closedown program* option set to *No*. This caused a second call to the program to crash as it tried to open a display file that is already open. This was a side effect from a fix to an unrelated issue published in r8.5.

C22E 376

Users of large models, for example models that require two or more user spaces, were experiencing *ERROR condition at nnn in YFLDPTRR1I* errors when they were performing *Find condition name* searches within the Action Diagram Services panel. This problem only occurred for large models that required more than one user space.

C22E 377

Generated RPG subfile functions (EDTFIL, EDTTRN, DSPFIL, DSPTRN, SELRCD) were experiencing RPG1255: *Session or device error occurred* errors. These runtime errors would occur when F4-Prompt was active for subfile functions that did not have any data, then the user positioned the cursor to the subfile portion of the display, and then pressed the F4 command key. We made modifications to the RPG Generators that guard against these runtime device errors.

C22E 378

Option 30 (Open Functions) against a function on the Edit Model List panel caused the error *Run-unit ended at 832 in YEDTMOLE1I*. This only occurred if option 10 had not been used since the model list was loaded or repositioned.

C22E 379

At runtime, a COBOL function using the built-in function *SUBSTRING was failing with the error MCH0603. This only occurred when the string length was greater than 50 characters.

C22E 381

If the value for YGENLMT was changed from the default (99999) then YGENSRC create/compile jobs MUST be processed in one single-threaded JOBQ.

Note: If more than one queue was used, then *Program...already exists in library*" and *CPF1240: Job Ended Abnormally* errors could occur.

There was a product limitation when YGENLMT model value was introduced. This fix removes this product limitation.

C22E 382

When searching for errors in an Action Diagram via the F7 key from AD services with Find Option=2, a error *Y2V0348: Parameter mismatch* (on a given AD line) was taking a higher precedence than the error *Y2V6300: Undefined parameter error* (on that same AD line).

C22E 383

Compile overrides for a display file were not being correctly applied. Though the override was appearing in the generated DDS, Z* CHRID(*JOBCCSID) for example, the override was not being submitted to the compilation job.

Note: There may be other combinations/sequences of such special characters appearing in the 2E object description that might have been causing similar problems.

The fix for issue C21E 50 (see page 50) resolves this problem.

C22E 384

In r8.5, the *EDIT FUNCTION PARAMETERS* screen changed its behavior when you placed a "?" in front of the array name, pressed Enter to list the arrays, and then pressed F3 to exit the EDIT ARRAY Panel. The array name was blanked out and message *Y2V0080* was sent. This fix restores the way an array name is passed back to the way it was before r8.5.

C22E 385

Subfile selection text and Action Bar (ABO) text was being displayed in English, even with a non-English LDO library in use. In r8.5, the LDO libraries were shipped with significantly fewer objects in them, with the relevant non-English text being programmatically retrieved at runtime. However, some non-English text was not being retrieved correctly.

C22E 386

When a function was copied from the *Templates file, any parameters for it were defined based on the access path used by the 'from' *Template function, rather than the access path over which the *to* function is based.

This problem, caused within the processing which duplicates the parameters for the *from* function to the *to* function, choose the access path to use based on the textual name of the access path used by the *from* function. This happened even if the *to* function is built over a different access path, for example an RSQ access path with a different key sequence.

C22E 387

A built-in Date function *DURATION with Duration Type *YEARS or *MONTHS was resulting in a compilation error QRP8046. This only happened when the generated code was RPG and the date fields used in the *DURATION function were type DTE.

C22E 388

In some cases, when an EXCEXTFUN called an EXCINTFUN, passing in a CON value longer than nine-characters and the EXCINTFUN used CVTVAR to convert the PAR string to NBR, the result NBR field would incorrectly contain zero.

C22E 389

In some cases the generated source for a SRVPGM (service program) was incomplete. A SRVPGM with more than 23 modules bound into it resulted in the compilation error CPF5D02: *No Module found for the specified value in the QSRVSRC member*. This problem was caused due to an insufficient field length being used to hold all the modules in the SRVPGM.

C22E 390

A Print File (PRTFIL) function based over an SQL access path, generated in RPG IV, with Record Selection processing appeared to loop. There were possibly other instances in RPGIV/SQL where SQL statements were erroneously generated before a loop or compound statement instead of after it. This problem occurred when the SQL statement to *Read next record* was erroneously generated before the DOU (DO until) loop. This fix ensures the SQL statement correctly generates within the DOU (DO until) loop.

C22E 391

When using codepage 273 (German) SQL ACP with SELECT/OMIT for DT# field gave a compile error SQL0104. This was caused by the generation of symbol '^' (the “circumflex” character at code point SD150000) instead of symbol '¬' (the “not” character at code point SM660000).

This fix addresses the problem by changing the SQL generator to use dynamic retrieval.

C22E 392

When a COBOL ILE external function was created and used a CRTOBJ over an array which has three or more key fields, and entries are added to the array in non-key-sequence, the array was filled incorrectly, with blank spaces and potentially out-of-sequence entries. This problem occurred due to a number of changes that were made to the 2E COBOL generator as a result of an APAR that IBM issued related to differences between the way that the COBOL compiler changed between OPM and ILE.

C22E 393

YEXCWSIPDD failed with the error message CPF4102 *File YLICTBL1 in library YLUSLIB with member YLICTBL1 not found*. This occurred when the 2E licensing library YLUSLIB does not exist on the machine. YEXCWSIPDD is an unlicensed command and is able to run on a machine that does not have a 2E licensing library.

C22E 394

We updated the *Building Applications Guide* and the *Command Reference Guide* to clarify that the YEXCWSIPDD command requires the target IFS to contain YCA/CAWS/UserData and YCA/CAWS/ProdData/YQSHLOG.

C22E 396

At times, the F4=Prompt was not working for blank Status fields (STS) within generated functions. We made modifications so that CHECK(AB) is always generated, within display files, for STS fields that have a check condition specified while not translating condition values. For example, Translate cnd values: ' '.

C22E 398

Promoting a 2E SQL Table failed during the Create Request and Compile step with error *VIM3787: Error copying source/objects for file index name to the request work library*. Also, if the Compile Request was set to *N*, then you received error *VIM1252: Object name in "from" library GenLibName does not exist*.

We changed the YRTVOBJR1I program to return the following four flags within the existing ASSIM (Assimilated file flag) field:

- The DBFGEN value for the ACP (*DDS or *SQL)
- The DBFACC value for the ACP (*DBFGEN or *TABLE)
- The maintenance status for the ACP
- The unique key flag for the ACP

Note: For PHY ACP's, ASSIM was returned as *YES or *NO. For LGL ACP's, the four-character ASSIM parameter contained separate flags.

We changed the YUSROBJR1I program to allow four new substitution variables to be used with the YEXCMDLLST command:

- &Y6/@Y6 - DBFGEN value (D or S)
- &Y7/@Y7 - DBFACC value (G or T)
- &Y8/@Y8 - MNTSTS value (I, D or R)
- &Y9/@Y9 - UNQKEY value (Y or N)

Several other programs were changed to accommodate the changed ASSIM parameter to the two programs mentioned above.

By making these values available, it is now possible to dynamically allow or disallow the promotion of SQL access path auxiliary objects, such as SQL indexes.

Note: In order for this fix to work, you must also apply MKS patch 287503 if using Implementer 10.1, or apply patch 211846, which is the upgrade patch from 10.1 to 10.2.

C22E 399

At times, a field deleted from a database relation did not lead to the removal of the same field if it was specified within any associated parameter details. For example, this incomplete deletion occurred when a Create Object function was created over the *Arrays file with a parameter interface which was defined using the, now modified, Physical Access Path. To correct this, we added processing to ensure complete deletion of such fields, so that associated parameter interface fields are also deleted.

C22E 400

From Session A, attempting to add an action diagram function call to a function that was newly created, by copying, in a concurrently running Session B resulted in the error *Run-unit ended at 1905 in YPARDTAR1I*.

This problem occurred due to an existing lock not being removed correctly.

C22E 405

When invoked in batch, YSTRTRGSVR was only starting one server, regardless of the value specified in the NBRSVR parameter. When YSTRTRGSVR was invoked interactively, the NBRSVR parameter was honored. This problem occurred when the logic used to start trigger server jobs was incomplete.

We modified the command source for YSTRTRGSVR to change the RANGE value for number of server jobs(NBRSVR) 99. We also updated the Help text to show *MAX is retained as 9 for backward compatibility purposes.

C22E 406

When an RPG IV EXCUSRSRC with an input parameter was used in a MOVEL, if the argument passed was a single comma, "," for example, the compilation failed with error RNF5340. We modified the generator program to include logic to check if the input parameter has ',' as part of an array + index combination. If the ',' in the input parameter is part of an array + index combination, the parenthesis are included. If it is just text, then it is passed as is and the parenthesis are not included.

C22E 407

*CVTVAR Run-Unit Ended error was occurring at statement 3784 within YMSGACTE2I. Within an action diagram when specifying parameters for a *CVTVAR function, at times, an error occurred due to a pointer not being set.

C22E 408

When a modified tab sequence was assigned for fields within the Device Design, source generation failed with errors CPD7520, CPD7542, CPD7544. This problem was caused by incorrect source being generated.

C22E 409

Impact Analysis was taking a very long time or, at times, ending with the error *YEXPREFR2I Run-unit ended at 95*. The YDSPMDLUSG and YDSPMDLREF commands were experiencing excessive I/O (s), or the YEXPREFR2I program was ending in error when looping (due to recursive calls).

C22E 411

Calling YUNSWS (Uninstall Web Service) from a command line, inside or outside the model, failed with error MCH4402 *Tried to use Return External instruction with invalid return point*.

C22E 412

The *SELECT MODULE* screen showed all versions of a module; DEV, PRD, ARC, for example. If a developer selected a non-current version, the Service Program would not compile as the Module did not exist. This screen only showed the current version of a Module by default. We made modifications to filter the non-current versions of a module and display only the current version of a module.

C22E 413

Execution of the YCHKMDL command while specifying ACTION(*UPDATE) was extremely slow when processing over large models. This was related to excessive I/O's.

C22E 417

CRTSQLRPG was failing with QRG5005 (CRTSQLRPGI failing with RNF5005), when RENAMES are generated while the YSQLVNM Model Value is set to *SQL. We made modifications to the RPG Generators so that the ZZINIT BEGSR source statement is now being sequenced correctly within the generated RPG source member.

C22E 418

Access Paths were failing to compile when Select/Omit criteria was specified for Timestamp (TS#) fields. We made modifications to the DDS and SQL generators so that timestamp values are no longer being truncated.

Examples:

DDS

Before: COMP(NE '0001-01-01-00.00.00.0000)

After: COMP(NE + '0001-01-01-00.00.00.000000')

SQL

Before: ¬='0001-01-01-00.00.00.0000

After: ¬='0001-01-01-00.00.00.000000'

C22E 419

At times, F4=Prompt was not working for STS fields within EDTRCD and PMTRCD functions while another field within the panel had encountered a keyboard error due to a 'Check condition'. This restriction would occur when another field on the panel was REQUIRED and the field had also specified a non-zero/non-blank *Check condition*.

We corrected this issue by modifying the DDS Generator to restrict the generation of the DSPATR(MDT) keyword and value.

C22E 422

If a subfile record contained a constant that appears on the last line of the subfile record, while no other fields appear on that line, any cursor-sensitive processing (such as built-in F4 prompting or user-defined prompting based on *Cursor field) was not working for any subfile records on a page except the first.

C22E 423

In a YGENSRC job that contained both service programs and programs, it was not possible to specify which should be created first. This resulted in programs being generated first, and binding in modules instead of service programs. Additionally, if a 2E model definition existed for a module but the source has not been generated and compiled, then it could not be added to the service program definition within 2E.

C22E 424

We updated the RPG Generators so that the index generated for a Constant is not truncated to 2 digits. For example, when converting the DT# date, the '@CN,001' value was being truncated to '@CN,00'.

C22E 425

Previously, when a file was used as an input parameter for a function, the Display Model References for a function correctly showed the file having a usage as *FUNPAR. However, Display Model Usages for the same file did not show any functions with usage as *FUNPAR.

Note: This fix rectifies a negative performance impact that exists in the test fix for C22E 425 for CA 2E r8.5.

C22E 426

In an RPG IV (ILE) EXCUSRSRC function containing an EVAL statement and a built in function such as %Scan with 2 input parameters, the usage #I of the second input parameter was not being correctly substituted. This was causing a RNF7030 error at compilation time. This was seen in a model using RPG naming with model value YHLLVNM as *RPG. To correct this issue, we modified the logic for obtaining the position of multiple occurrences of '#' in the source string.

C22E 427

Previously, when selecting an invalid option on *EDIT FUNCTION DETAILS* screen for a function of type Service Program, the selected invalid character might not display in the error message. We corrected this issue so that the selected invalid character always displays in the error message.

C22E 430

When you modified the parameter sequence numbers, the corresponding parameters were not being resequenced within the subfile when you pressed ENTER. Also, when you set the Duplicate Parameters Option to *Y* and modified the parameter sequence numbers, the corresponding PRn values for Pgm Ctx and Par Ctx were not being updated when you pressed ENTER.

When you defined parameters and usages while the Duplicate Parameters Option was set to *Y* and then changed to *N*, the *Parameter is duplicate* error message was not being displayed when the EDIT FUNCTION PARAMETER DETAILS panel was initially displayed. Previously, this *Parameter is duplicate* error message would only display when you modified a usage. Therefore, if you did not modify an existing usage, you would have been unaware that invalid usages occurred if you only displayed the EDIT PARAMETER DETAILS panel without further modifications.

Additionally, when you set the Duplicate Parameters Option to *N* and the *Parameter is duplicate* error occurred within the EDIT FUNCTION PARAMETER DETAILS panel, you were not allowed to freely modify the usages without first navigating to the Duplicate Parameters Option and changing the setting from *N* to *Y*.

To correct these issues, we improved parameter validation and program flow. When you set the Duplicate Parameters Option to *Y*, there are validations that enforce unique parameter sequence numbers.

C22E 432

The YCRTWS and YUNSWS commands require the user issuing command to have special authorities *ALLOBJ and *IOSYSCFG. This is due to the underlying IBM IWS scripts requiring those special authorities and is documented in the section "Web Services Limitations" in the Building Applications Guide.

Additionally, the YCRTWS and YUNSWS commands were missing from the Command Reference Guide.

C22E 433

When you create a function with an object type MOD, and the source name is greater than eight-characters and the MOD object is called from another function with an object type PGM, the generation of the PGM object source fails at the CALLB statement, incorporating the call to the MOD object. This error is a problem with the RPGIV CALLB statement generated when the MOD object is called from a PGM object.

We modified the YRPGCLLG1I program to overcome this problem and correctly generate the RPGIV CALLB.

C22E 434

When the YENDTRGSVR command was executed, the YTRGSVR jobs ended successfully. However there was no clear message that indicated the cause for the end of the trigger server job in the job log. To correct this, we added diagnostic information for the YENDTRGSVR command.

Note: Modified 2E trigger-related objects need to be updated within each trigger runtime library by executing the YDUPAPPOBJ command specifying DUPOPT(*TRG).

Important! You should not run YDUPAPPOBJ with Y1SY or any product libraries as the target.

For more information about the YDUPAPPOBJ command, see the *Command Reference guide* and the *Generating and Implementing Applications guide*.

C22E 435

Error Y2V3116 results when adding a COBOL commented MOVE statement in an Execute User Source with an unfinished or invalid parameter in the form USR-PARM.

We added a check for the commented statement before sending error Y2V3116.

C22E 441

The YDUPAPPOBJ command was not duplicating the objects YCMDPRCS1R, YGNRPRCS1R, YGNRPRCS2R, YMSGPRCS1R, and YYYYMSG when DUPOBJ was set to *TRG or *ALL. These objects are necessary to work with the Trigger Reference File(YWRKTRGREF)

We changed the Y1USROBJ file to include a new member Y1TRG. We also populated the member with the details of the objects YCMDPRCS1R, YGNRPRCS1R, YGNRPRCS2R, YMSGPRCS1R, and YYYYMSG. The program YDUPAPPR@C now includes logic for reading the member Y1TRG.

C22E 443

When YAPYMDLCHG was executed, the processing program granted authority to all the objects in the model library based on a reference file. This was resulting in authority of all the objects being changed to replicate the use authorities of the reference file.

We modified the command to correctly reinstate object authorities using an authority list.

C22E 444

Users were seeing compilation failure for source types RPGLE and CBLLE when comparing the constant *ZERO to a condition value *ZERO inside any Internal DBF, or in nested internal functions.

We modified the programs YACTCTXK4I and YACTCTXR4I to generate a 0 instead of ZEROS for DBF and USR functions for CBLLE source type. The programs now also generate a 0 instead of *ZEROS for DBF and USR functions for RPGLE source type.

C22E 445

In a new r8.5 model, created using the German LDO libraries, the name of the *Standard header/footer file appears in English, not German. This text came from message Y2F1200 in Y2ALCMSG. We updated the message so that the proper file name appears in German: *STANDARD KOPF-/FU~ZEILE

C22E 446

In an EDTRCD2/EDTRCD3/DSPRCD2/DSPRCD3 function, when an attempt is made to initialize the access path fields in 2ND/3RD context, using the context based initialization, it does not work correctly in all scenarios. Problems occur regardless if the record is new or existing. These problems occur due to an error in the algorithms used for naming and populating the fields on DTL/2ND/3RD screens.

Note: For more information, see the Knowledge Base article The working and limitations of EDTRCD2, EDTRCD3, DSPRCD2 and DSPRCD3 function types.

C22E 18495726

The DISPLAY ALLOWED VALUES panel, which is invoked when a user types *?* against the *HLL Type* in the EDIT FUNCTION DETAILS panel, was displaying some of the target HLL list descriptions incorrectly.

C22E 18550141

In the Japanese localized product, the subfile selector text ("Type options, press Enter.") appeared as upper-case English rather than in Japanese. This fix now retrieves the message from the Y2ALCMSG message file.

C22E 18589189

We added help text for the MACHINE parameter for the YUNSWS command.

C22E 18696513

The *Contains Search* functionality was not returning records correctly when searching in DBCS environments that contained Shift In and Shift Out characters.

C22E 19046944

When a user copied a function from an existing function, if a Default Template function existed, then messages in the joblog incorrectly stated that the Default Template function was copied. This fix ensures that, when a function is copied, the copy is successful and are not any messages in the joblog stating that the function was copied from a template function.

C22E 19284039

When calling the YEXCWSIPDD command, passing *GENLIB to the WSIPDDLIB parameter failed with the error: *Value '*GENLIB' for parameter TOFILE not a valid name*. We made modifications to correctly process the *GENLIB value by retrieving the name of the generation library.

C22E 19932075

Additional upgrade processing was required to include the addition of the YPRDRDRRFA data area into user models.

C22E 20350772

In some circumstances, the object text on the SERVICE PROGRAM MODULES screen was errantly derived from the *MODULE object rather than using the model object name. We corrected this issue.

C22E 20378178

Upgrade to Release 8.6 fails if YMDLLSTRFP is missing the standard member YMDLLSTRFP.

Note: One legitimate way to remove this member from the file is to do YWRKMDLLST and take option 4=Remove on the YMDLLSTRFP list.

We modified the processing to upgrade from model level 44 to 45. This adds member YMDLLSTRFP to YMDLLSTRFP if it has been removed. If not, the duplicate of YMDLLST10L will fail.

C22E 20509087

When doing an FF on a function call, the EDIT ACTION - FUNCTION DETAILS panel displayed message Y2V6000 "File '*Arrays' is in use, details cannot be accessed." Because the locked file cannot be expanded to verify the parameters on the called function, parameters on the function call were not listed. Since the parameters were not being accessed, the function call was synchronized incorrectly. Therefore, if the function was saved on exit, the parameters were being removed from the function call with unpredictable results.

Note: The problem only occurs when a model needs to be synchronized and YOPNACC is set to *YES.

We updated processing on the EDIT ACTION - FUNCTION DETAILS panel so that if a file cannot be expanded due to locking, and message Y2V6000 is sent, then the processing does NOT try to synchronize parameters

C22E 20556989

Usages of an array by the CVTVAR built-in function were not listed on YDSPMDLUSG. We corrected this and YDSPMDLUSG now shows the Usages.

C22E 20608293

If a function call had an existing constant (CON) argument, the context is over-typed with a field-related context (WRK, for example) the value is over-typed with ?, and the user subsequently presses F3 without making a selection from the DISPLAY FIELDS panel, you received an internal error and the run-unit ended.

We made improvements to catch all such errors when prompting all field-related contexts in this manner.

C22E 20619779

The function generation could fail for certain header/footers when the value populated to window title *WDWTTL was longer than 25-characters.

We improved the window title centering logic so that if the result would have been less than two, then it is changed to be column two.

C22E 20676951

Array deletion processing was failing to honor array usage in *CVTVAR function. Even when used by *CVTVAR, the processing was not preventing the array from being deleted.

C22E 20761586

CVTVAR with Factor 2 set to ELM context with an array that has > 1 element can give generation error in source Y2V0080 with RPG/RP4 generation. We made a modification to the generator to resolve this problem.

CA 2E Toolkit

We completed the following fixes, corrections, or updates for the CA 2E Toolkit:

C21E 49

If the spool file did not exist, the YCVTSPLF command was failing and showed error CPD0085. Now, if a spooled file does not exist, error CPF34C0 *Spooled file XXX not found* is sent.

Note: If you specify SPLNBR, and that spooled file does not exist, then error CPF3303 is sent.

C21E 50

The Compile preprocessor was not correctly processing T* lines that contain an apostrophe following a left parenthesis.

Note: There might have been other combinations/sequences of such special characters appearing in T* lines that were causing similar problems. This was causing problems in subsequent lines, Z* for example.

C21E 51

The YCRTOBJ command was not honoring compile preprocessor commands, for example Z* CRTLF MAINT(*IMMED). The compile preprocessor was being interrupted when the submitted command had a qualifying library.

C21E 53

The functionality for F16 was working correctly but the corresponding literal (F16=*RRN) was missing on the English version of the *Display File - Multi-record display* screen.

C21E 18595499

The 1E Compile Preprocessor was incorrectly processing DBCS strings that contained parenthesis.

C21E 19579093

When a 1E menu was encountered that did not allow the use of the F3 (Exit) or the F12 (Cancel) key (either because it was the top-level menu or because the option had explicitly been disallowed in the menu configuration), a *prehelp* error was sent to the screen. This locked the keyboard and required the user to press *Reset* to re-enable the keyboard.

We created the message *YMN0101: Function key not allowed*. When the user presses F3 or F12 on a Level 1 menu (highest level menu), this message appears and the screen does not lock.

C21E 20155193

The 1E Compile Preprocessor was comprised of separate programs. Due to the separate programs library list changes could cause a wrong version of the YBRTP2R program that was called. The YBRTP2R object is now compiled as a module, which is bound by copy into the YBRTPRC program.

C21E 20155236

We added Service Program Diagnostic Load Utility (SPDLU) processing to all 1E service programs. This functionality is intended to help CA Development and Support debug problems with Web Option.

Change Management

We completed the following fix for CA 2E Change Management:

C2CM 19

The *SRVPGM generator program includes the generated library of each module in the generated source,for example:

```
/*Z: MODULE(MYMDL/UUIJXFR MYMDL/UUIKXFR) +
```

Since CM moves objects being promoted into a temporary work library, this was not working with CM.

We changed the *SRVPGM generator program to not include the library name (only the module name is specified), so *LIBL will be used, which changes the example to:

```
/*Z: MODULE(UUIJXFR UUIKXFR) +
```

CA 2E Translator

C2TRAN 20648444

There were problems with the *Translate Panel Literals in Dictionary* option. Taking option *1 = Translate in Context* gave an empty display. Option *7 = Work with Formats* lists the formats but when select one, nothing is displayed.

We resolved the issue so that panel literals can now be viewed and translated with the Translate in Context option.

Web Option

We completed the following fixes, corrections, or updates for the CA 2E Web Option:

C2WEB 211

Web Option had performance problems related to multiple, repeated click issues. We made the necessary changes to correct this issue.

C2WEB 215

Checking each skeleton for changes as it is loaded consumes large quantities of the CPU. To correct this we added the YSKLCHK Web Option control value, which allows you to specify how frequently the skeleton date should be checked.

C2WEB 216

Potentially significant quantities of CPU were being consumed when checking the string length in the rtv_data_len() procedure. This was causing a performance hit at runtime. We discovered that this Procedure was unnecessarily being called in some situations.

Note: You might continue to experience performance issues with the rtv_data_len() procedure itself. However, this is less of an issue with the smaller memory allocation.

C2WEB 217

The YINZW2EENV command was failing if the Y2WEB and Y2WEBVENG libraries were combined. From Release 8.5 onwards, merging the Web Option product and LDO libraries must be performed after Web Option is initialized or upgraded.

C2WEB 218

If the Web Option product and LDO libraries Y2WEB and Y2WEBVENG were merged, the YGENMLS command (and possibly others) might have gone into a loop.

We changed the YW2ELDOR1C program, which sets/resets the library list prior to performing any processing, to avoid errors when it attempts to add a library to the prior library list.

C2WEB 219

When a screen element had the same element Customization Identifier as the previous element (that had any element customization), the second element's customization was not being used.

C2WEB 220

With Web Option, any customization applied to a Header/Footer in the 2E model is applied to all generated skeletons, regardless of what Header/Footer you are using. This is due to a limitation in the Customization process; it is not possible to associate a customization element with one specific Header/footer.

C2WEB 221

For a Web Option function with a skeleton and that has a status field with a condition where the file value is 0 (zero), that 0 value is not generated as a valid selection value in the drop down list at runtime.

Modifications have been made to correctly generate 0 as an option.

C2WEB 222

An embedded window function was not being identified and was not associated with the generated markup language skeleton at runtime. This was due to window functions being displayed where the screen cross-reference file contained records where SIDOFF was greater than zero and close to the maximum screen size for a 27 by 132 screen of 3564 characters. The screen identification processing caused a buffer length overrun error.

C2WEB 223

When the YGENMLS command was first used within a job, the MLS Syntax (MLSSYN) and Element Customization (ELMCST) data was cached in memory to improve performance on subsequent calls. However, if any changes were made to those data, they weren't picked up by YGENMLS unless the user signed off and then back on.

We changed the YGENMLS command to *flush* the memory cache for a job. This occurs before it performs any processing, which ensures that each call to YGENMLS uses up-to-date data.

C2WEB 224

This problem covers a number of separate issues that relate to the use of Element Customization by Web Option, specifically, the way in that the *DROP ELEMENT ElmCst was applied and the hierarchical levels of ElmCst which should be applied. Various problems with the Element Customization generator caused a number of seemingly unrelated problems: ElmCst not being applied, being applied incorrectly, being applied to the wrong element.

C2WEB 227

There were performance problems with Web Option when scaling to large numbers of concurrent users – greater than 1000), for example. For installations where there are a large number of users, the original limit of 30 server jobs was insufficient. Each job could get overloaded, which was reducing response time for users. This applied especially when one user job was allocated to a server job with a long-running query, which was locking up not only that user's job but also the jobs of all other users allocated to that same server.

Additionally, at runtime, intermittent problems with the CREATE_WINDOW procedure were occurring.

By increasing the number of server jobs, all users are now spread out more 'thinly', reducing the number of users who will be affected in such instances. We raised the maximum number of server jobs from 30 to 500, and added further processing which checks for locks before ending the Web Option auditor job and clearing the data queue.

C2WEB 228

From the Work with Screen Cross references screen (YWRKSCRXRF), when entering a 2 to Edit the Cross Reference record, and then changing the Skeleton Name on the Edit Screen Cross-reference panel, you would receive a message that the member was renamed. However, on return to the Work with Screen Cross references screen, the original name was displayed. Pressing F5 (refresh) did not update the Skeleton Name.

This occurred because to the details of a skeleton are cached for performance reasons and although the record on file had been changed, the data displayed on the main YWRKSCRXRF panel was being retrieved from the cache. Therefore, we changed processing to update the internal memory arrays from file on every screen reload.

C2WEB 231

The W2E_SELECT field was being set to an incorrect value when the displayed screen was a Window function. This then caused the cursor to not be placed within the first input capable field within the window. This problem was occurring due to the window offset not being taken into consideration when calculating the field related to W2E_SELECT.

C2WEB 232

The text within the YMLSXRFP file for record SYS0000014 was set as CONFIRM: For non-English environments, this was causing the Confirm Window for a Window Function to be displayed in JIT. In the YMLSXRFP file, we changed the matching record for SYS0000014 to include a sequence number (SEQ) of 3, so only if all 3 records are matched will the confirm prompt display.

C2WEB 233

Action Diagram code that was directly commented out, and which referenced the RCD.*SFLSEL field, was generating Options in the MLS Syntax for the #1SEL (Subfile Select) field. This was incorrect as Subfile selections that are commented out should not be showing up.

C2WEB 234

At times, a popup/embedded window was not being identified properly. This was related to the current size of the window when it nearly overlaid the underlying screen to include the top line, which contains the screen identifier. When a user pressed Enter on that window, the iSeries attempted to restore the underlying screen first. When the screen was being rewritten, only the visible portion of the screen was being written and the hidden portion of the screen, which contained the screen identifier, was being ignored.

C2WEB 235

When the YWRKSCRXRF (Work with Screen Cross-References) command was executed while a record with a sequence number greater than one was being edited, saving the changes and exiting the edit panel produced the error *CPF5009 – Duplicate record key in member YMLSXRF.*

C2WEB 236

At times, a screen was not identified properly when the screen was first displayed in JIT mode prior to a skeleton generation. The screen would continue to display in JIT mode after skeleton generation even if the Web Option server was restarted using YSTRW2ESVR RESTART(*YES). The screen was being identified correctly when the Web Option server was fully stopped and started.

C2WEB 237

Values present within the value list for a field in the display file DDS had their leading zeros stripped off when the corresponding drop down list was generated in HTML.

We made the necessary changes to ensure that the leading zeros are not removed from the condition values at the time of generating the HTML for the display file. Leading spaces are retained when the HTML is generated. We also ensured that the proper list values are loaded into the caching array, so they can be correctly retrieved during runtime. This change is reflected when YLNGOPT values is either *SINGLE or *MULTIPLE.

C2WEB 238

When a large number of Web Option server jobs were running concurrently and the YENDW2ESVR command was issued to end the jobs, the Web Option environment did not start up properly when the YSTRW2ESVR command was subsequently issued.

We updated the Web Option server to change the way it deletes the temporary objects is uses and added a new error message for display when the Web Option server is ended with FORCE(*YES). The number of seconds taken to end the Web Option server is now a function of the number of active server jobs and may take as long as a minute if many server jobs are active.

The final fix library, YC2WEB238, contains all of the necessary objects.

C2WEB 240

A properly running Web Option installation suddenly experiences runtime errors such as MCH0603 from the YALLOC module, and starts displaying JIT screens. This error was occurring due to an issue with the memory management routines used in Web Option, which leads to the runtime errors and displaying the JIT screens.

C2WEB 18387355

Running YCRTW2EENV with parameters W2ELIB and LDOLIB set to *W2ELIB and Data Option set to *COPY, caused the message *Web Option environment created successfully*. While there was supposed to be two different warning messages, there were no warning messages in the joblog, and YMLSSRC was not being created in the environment library.

We made changes to copy the W2ELIB copy of YMLSSRC into environment data library (LDOLIB copy has been removed from the product). Also, the messages are only cleared if the program ends without warnings.

C2WEB 18806512

YCRTW2EHTP was giving error W2E8502 when running in job with DBCS CCSID. The fix correctly opens and creates the HTTP configuration file using the single byte code page associated with the job's DBCS CCSID.

C2WEB 19639596

We changed the Web Option control values so that the environment library contains only those control values which have been changed from the default (shipped) value held in the product library. At runtime, environment-level control values override default control values. The YWRKW2EVAL display identifies environment-level control values and gives you the ability to delete them. If you change a control value, the default value remains in the product library and the changed value is copied to the environment library. When that environment-level control value is then deleted, the default value in the product library is used.

C2WEB 20145007

We added Service Program Diagnostic Load Utility (SPDLU) processing to all Web Option service programs. This functionality is intended to help CA Development and Support debug problems with Web Option.