ORACLE®

JD EDWARDS WORLD

# JD Edwards World Programmer's Guide

**Version A7.3 to A9.1**

**Revised - June 11, 2008**

JD Edwards World
Programmer's Guide



**Open Source Disclosure**

Oracle takes no responsibility for its use or distribution of any open source or shareware software or documentation and disclaims any and all liability or damages resulting from use of said software or documentation. The following open source software may be used in Oracle's PeopleSoft products and the following disclaimers are provided.

# Send Us Your Comments

JD Edwards World Release A9.1 Documentation, Revised - June 11, 2008

JD Edwards World welcomes your comments and suggestions on the quality and usefulness of this publication. Your input is an important part of the information used for revision.

- Did you find any errors?
- Is the information clearly presented?
- Do you need more information? If so, where?
- Are the examples correct? Do you need more examples?
- What features did you like most about this manual?

If you find any errors or have any other suggestions for improvement, please indicate the title and part number of the documentation and the chapter, section, and page number (if available). You can send comments to us by e-mail at:

jde_world_doc_ww@oracle.com

If you would like a reply, please give your name, address, telephone number, and electronic mail address (optional).

Contact a JD Edwards World representative by calling Oracle Global Support Center at 1-800-289-2999 for current information or if you have any questions regarding this document.

# Table of Contents

# Overview

This guide details technical information for the JD Edwards World A9.1 release. This guide is intended for programmers who integrate other software with JD Edwards World software, or who customize JD Edwards World software for particular needs. This guide does not provide an overall explanation or how programs work together within the JD Edwards World system, but it details specific areas of interest to programmers.

## Before You Begin

**Note:** If you are upgrading your system from the A8.1 release to the A9.1 release, use the *A81 to A91 Programmer's Guide*.

**Note:** The file changes indicated for A7.3 Cumulative Updates may or may not apply to you, depending on which Cumulative Update level you are upgrading from. Some of the file changes were first included in UQFs/PCCPYs. JD Edwards World recommends that you check your custom programs for any of the changed files to determine if you need to modify your code and/or recompile the program(s).

**Note:** JD Edwards World A9.1 is based on the JD Edwards World A8.1 software. When you see sections that discuss A8.1 changes, they apply to A9.1 as well.

## About this Guide

This guide includes database and system changes for the following products:

- Cross Industry systems
- Distribution and Manufacturing systems
- Human Resources and Payroll systems
- Technical Foundation
- Localization

Database changes include items such as added or changed fields, new, changed or obsolete files, and new, changed or obsolete logical files. For detailed database changes, see *Database Changes* in the product sections that apply to you.

System changes describe various enhancements, such as new programs and files. The descriptions provide helpful information you need be aware of as you integrate

or customize JD Edwards World software. For detailed system changes, see *System Changes* in the product sections that apply to you (such as *Cross Industry Changes and Enhancements*).

In addition, this guide includes information about:

- Performance considerations for programmers
- National language support
- Double-byte enablement

# General Considerations

If your programs use files that have database changes, review the following items and make the appropriate changes:

- If an added or changed field applies to an entry screen, modify the screen by adding or changing the field. Clear the fields in S001
- If the program can add records to the file, initialize the added or changed fields before you add records
- Check for internally defined data structures that describe file records. To accommodate field changes, add fields, and/or change the size of the data structure

## Added Fields

To determine the programs that the added fields affect, perform cross-references to the following items after you review the Added Fields tables in this guide:

- Physical Files. Consider the programs in update status only
- Logical Files. Consider the programs in update status only. Use the IBM command DSPDBR to display all of the logical files associated with an updated physical file
- File Aliases. Consider the programs in update status only
- I/O. Consider only the calls that write to an updated physical file (where @@OPER is set to WRITE)

Initialize each added field to *BLANKS (for alpha fields) or *ZEROS (for numeric fields). Some fields require you to perform additional tasks. These instructions appear in the appropriate chapter for system changes (such as *Cross Industry Changes and Enhancements*).

## Changed Fields

For left-justified alpha fields that have increased in size, additional spaces are filled with blanks. Right-justified alpha fields that have increased in size, do not require the blanks. For example, if a left-justified field increases its size from 20 to 40 alpha characters:

| Previously Stored Value | New Stored Value |
|---|---|
| '5555430 ' | '5555430 ' |

For functional I/O and other programs that use files, in which field sizes have changed, check data structures, format definitions, and work fields for related size definitions. Work fields usually have hard-coded definitions if they are defined the same as data items in a file or data structure. JD Edwards World recommends that you use a *LIKE/DEFN structure instead of hard-coding the field length.

For example, instead of:

```
MOVE WPPH1  $PH1  20    (Hard-coded length of 20)
```

Use:

```
Subroutine s999  -  Housekeeping
 *LIKE      DEFN  WPPH1 $PH1
```

**Note:** You may need to adjust hard-coded references to all fields that have increased in size.

## File Layout Issues

The following types of file changes also require file layouts to change:

- New fields
- Changed field sizes
- Deleted fields
- Reorganized or shifted fields
- New key fields

You may need to reorganize data structures that are over these files or their aliases (files that resemble them). In addition, you may need to adjust the beginning and ending values for added or changed fields. Review the Database Changes chapters in this guide to identify file changes, and then refer to the appropriate chapters for system changes (such as *Cross Industry Changes and Enhancements*) to determine file specifications for specific field organization.

# Technical Foundations – General Changes

This chapter provides database and system changes for the Technical Foundation systems.

## Database Changes

This section lists file level database changes for the Technical Foundation systems.

### Summary of Database Changes

Double-click the following icon to open the Technical Database Changes Summary.

### Changed Physical File Layouts

Double-click the following icon to open the Technical Changed Physical Files Detail.

### New Logical Files for Existing Physical Files

Double-click the following icon to open the Technical New Logical Files for Existing Physical Files.

### Changed Logical Files

Double-click the following icon to open the Technical Changed Logical Files for Existing Physical Files.

## New Physical Files in A8.1

| File | File Description | Prefix | Explanation |
|---|---|---|---|
| F9207 | Error Message Program to Call | FR | Replaces F9205 |
| F9210 | Data Field Specifications | FR | Replaces F9201 and F9206 |
| F96012 | Generic Function Key Master | XF | |

## Obsolete Files in A8.1

| File | File Description | Explanation |
|---|---|---|
| F9201 | Data Field Specifications | Replaced by F9210 |
| F9205 | Error Message Program ID | Replaced by F9207 |
| F9206 | Alternate User Defined Codes – Tag File | Replaced by F9210 |

# Copy Member Changes (non-Database)

| Rptg Code | Copy Member | Func-tion | Description | Action | Release/ Cume |
|---|---|---|---|---|---|
| 73 | C73012 | COPY | /COPY Multi-Intra State Taxes | Changed | A91 |
| 73 | C73012L | CPYL | /COPY Multi-Intra State Taxes | New | A91 |
| 82 | C82PARM | COPY | Query Parms for RPG Programs | New | A73CU09 |
| 91 | C9802L | CPYL | Data Dictionary Print | New | A73CU16 |
| 98 | C0000L | CPYL | Business Unit Security Check | New | A73CU14 |
| 98 | C0000L | CPYL | Business Unit Security Check | Changed | A91 |
| 98 | C0001L | CPYL | Edit Action Code | New | A73CU14 |
| 98 | C0001L | CPYL | Edit Action Code | Changed | A91 |
| 98 | C0001T | COPY | Edit Action Code - Including PC Import/Export | New | A91 |

| Rptg Code | Copy Member | Func-tion | Description | Action | Release/ Cume |
|---|---|---|---|---|---|
| 98 | C0001TL | CPYL | Edit Action Code - Including PC Import/Export | New | A91 |
| 98 | C0011L | CPYL | Center Descriptive Titles | New | A91 |
| 98 | C0012 | COPY | Right Justify Numeric Fields | Changed | A73CU09 |
| 98 | C0012L | CPYL | Right Justify Numeric Fields | New | A73CU14 |
| 98 | C0012L | CPYL | Right Justify Numeric Fields | Changed | A91 |
| 98 | C00151L | CPYL | Currency - Translate Video Fields to Data Base | New | A73CU14 |
| 98 | C00151L | CPYL | Currency - Translate Video Fields to Data Base | Changed | A91 |
| 98 | C00161L | CPYL | Format Numeric Fields for Output with Overrides | New | A73CU14 |
| 98 | C00161L | CPYL | Format Numeric Fields for Output with Overrides | Changed | A91 |
| 98 | C0016L | CPYL | Format Numeric Fields for Output with Overrides | New | A73CU16 |
| 98 | C0016L | CPYL | Format Numeric Fields for Output with Overrides | Changed | A91 |
| 98 | C0034 | COPY | Load Menu Screen Titles | Changed | A91 |
| 98 | C0040L | CPYL | Alphanumeric Left Justify-Compress Blanks to 1st Character | New | A73CU06 |
| 98 | C0040X | COPY | Alphanumeric Left Justify and Compress - Language Support | New | A73CU12 |
| 98 | C0040X | COPY | Alphanumeric Left Justify and Compress - Language Support | Changed | A73CU15 |
| 98 | C0040XL | CPYL | Alphanumeric Left Justify and Compress - Language Support | New | A91 |
| 98 | C0042L | CPYL | Right Adjust Alphanumeric Field | New | A91 |
| 98 | C00E1 | COPY | Setup Interactive Export | New | A91 |
| 98 | C00E1L | CPYL | Setup Interactive Export | New | A91 |
| 98 | C00E2 | COPY | Perform Interactive Export | New | A91 |
| 98 | C00E2L | CPYL | Perform Interactive Export | New | A91 |
| 98 | C00EXWL | CPYL | Window Position Determination | New | A73CU13 |
| 98 | C00EXWL | CPYL | Window Position Determination | Changed | A91 |

| Rptg Code | Copy Member | Func-tion | Description | Action | Release/ Cume |
|---|---|---|---|---|---|
| 98 | C00I1 | COPY | Import Data from CSV File | New | A91 |
| 98 | C00I1L | CPYL | Import Data from CSV File | New | A91 |
| 98 | C00I2 | COPY | Import Data for subfiles | New | A91 |
| 98 | C00I2L | CPYL | Import Data for subfiles | New | A91 |
| 98 | C00IEM | COPY | Import/Export Messages | New | A91 |
| 98 | C00IEML | CPYL | Import/Export Messages | New | A91 |
| 98 | C00IESTS | COPY | Import/Export Status - Interactive | New | A91 |
| 98 | C00IET | COPY | Import/Export Termination | New | A91 |
| 98 | C00IETL | CPYL | Import/Export Termination | New | A91 |
| 98 | C00IEXP | COPY | Export Data to CSV File - Interactive | New | A91 |
| 98 | C00IIMP | COPY | Import Data from CSV File - Interactive | New | A91 |
| 98 | C00IMPSF | COPY | Import Data for Subfile - Interactive | New | A91 |
| 98 | C00LNML | CPYL | Import/Export Long File and Path Names | New | A91 |
| 98 | C00RSCL | CPYL | Copy Module - Retrieve Soft Coding - Reports | New | A73CU15 |
| 98 | C00RSCL | CPYL | Copy Module - Retrieve Soft Coding - Reports | Changed | A91 |
| 98 | C00SCL | CPYL | Copy Module - Retrieve Soft Coding | New | A73CU14 |
| 98 | C00SCL | CPYL | Copy Module - Retrieve Soft Coding | Changed | A91 |
| 98 | C00SSEL | CPYL | Import/Export Substitution String Expansion | New | A91 |
| 98 | C74S347 | COPY | DREAM Writer - Dynamic Report Processing | New | A73CU16 |
| 98 | C74SFL | COPY | Format Fields left or right with blanks or no blan | New | A73CU16 |
| 98 | C81DRPT2 | COPY | DREAM Writer - Dynamic Report Processing | New | A73CU16 |

| Rptg Code | Copy Member | Func-tion | Description | Action | Release/ Cume |
|---|---|---|---|---|---|
| 98 | C81DRPTL | CPYL | DREAM Writer - Dynamic Report Processing | Changed | A91 |
| 98 | C9801L | CPYL | PRINT COVER PAGE | New | A73CU16 |
| 98 | C98031L | CPYL | Load Processing Options and Level Breaks | New | A73CU15 |
| 98 | C98031L | CPYL | Load Processing Options and Level Breaks | Changed | A91 |
| 98 | C9803L | CPYL | Load Processing Options | New | A73CU14 |
| 98 | C9803L | CPYL | Load Processing Options | Changed | A91 |
| 98 | C98208 | COPY | Acquire a Transaction eSignature | New | A73CU13 |
| 98 | C98208L | CPYL | Acquire a Transaction eSignature | New | A73CU13 |
| 98 | C98208L | CPYL | Acquire a Transaction eSignature | Changed | A91 |
| 98 | C98209 | COPY | Release a Transaction eSignature | New | A73CU13 |
| 98 | C98209L | CPYL | Release a Transaction eSignature | New | A73CU13 |
| 98 | C98209L | CPYL | Release a Transaction eSignature | Changed | A91 |
| 98 | C9822 | COPY | Double Byte Truncation Routine | Changed | A91 |
| 98 | C9822L | CPYL | Double Byte Truncation Routine | New | A73CU14 |
| 98 | C9822L | CPYL | Double Byte Truncation Routine | Changed | A91 |
| 98 | C997L | CPYL | Build Allowed Values Work Array | New | A73CU14 |
| 98 | C997L | CPYL | Build Allowed Values Work Array | Changed | A91 |

## Server Changes

Double-click the following icon to open the Technical Foundations New and Changed Server Programs.

# Technical Foundations – Enhancements

This chapter describes enhancements made within Technical Foundations that apply to all modules.

## A8.1 Changes

### Special Instructions for Fields

#### FROWDI - EnterpriseOne Data Item Name (F9210)

This field is the 'C alias name. It was removed from the F9204 file and placed into the F9210 file. The COBOL alias name remains in the F9204 file.

#### XFFLDN - Field Name for Generic Exit (F96012)

This field must contain #G01 through #G30. Uses:

- To determine the order in which the generic exits are displayed to the user
- To implement security through Function Key Security (P9611)

#### XFPFL1 - Zero Parameters 1 through 10 for Generic Exit (F96012)

These fields are optional. If needed, they should be filled left to right (that is, do not fill PFL2 if you have not filled PFL1). The fields can contain a variable name from the calling program (for example, VDAN8), or constants (*BLANK or *ZERO), or a literal value (for example, DL). You need to know the parameters the called program expects to specify these correctly. The calling program must be observable in order for the software to get the value from the variable you specify and supply it as a parameter.

If the calling program had observability removed, the system displays the following error message when users select a generic exit that was defined with parameters.

020S Error Opening Source File Template

Cause System Error

Resolution Contact your technical support person. For more information, view JDE.LOG and JDEDEBUG.LOG

## Conversion and Update Programs

### COBOL Alias Field Names

EnterpriseOne does not update the F9204 file with COBOL alias field names. An update program is available to create COBOL alias field names for the associated data dictionary items.

Run program P92041 to verify COBOL alias field names for each data dictionary item and create names for those that do not have names.

### Available Function Keys (F24) and Cursor Sensitive Help (F1) Window

The Generic Exit implementation requires you to call J96012 rather than P9601H. This change was made in the A8.1 source code, but if the customer has custom modifications, or if a programmer copies source into A8.1 from another environment, the conversion program must be run against the A8.1 source code. DREAM Writer drives this program. The form ID is P96012CVT. The conversion replaces calls to P9601H with calls to J96012.

## Copy Books

Recompile programs that use the following copybook.

**Caution:** Make sure that you do not make copybook changes that conflict with A8.1 changes.

### I00SC - Soft Coding Server

The IOOSCG field was redefined as a 30-byte field with 30 one-byte subfields named #G01 through #G30. These are the flags used for Generic Function Key Exit security. The ##RVAL field was further defined with two new subfields (##RV1 and ##RV2), which are used to support the long date (MM/DD/CCYY or DD/MM/CCYY) and century for cursor-sensitive help. ##RV1 contains the long date returned from P00CAL. ##RV2 contains the century returned from P00CAL.

### Function Key Help (P9601H)

Generic exit capability was added to the A8.1 release, allowing customers to add an exit to another program from an existing program without changing the code in the existing program. The new exits are defined in file F96012 from menu G90 (option 17) or G92 (option 9). The Generic Exit file is keyed on Country, Language, and Screen Name. Customers can define different exits for the same screen for users in different countries. Users can press F24 to display the generic exits and list any generic exits defined for this screen for the user's

country/language, followed by the available function keys and selections. The Users execute the generic exit by entering '4' next to the desired exit.

The Function Key Help program (P9601H) was replaced by J96012. SOOEX calls the J96012 program. To use the new function, scan your source code for P9601H and replace it with J96012. The program J96012CVT converts your source to J96012. To run the J96012CVT program, select Replace P9601H with J96012 from the A8 Conversion Utilities menu (BA6).

The generic exit function supports up to ten parameters in addition to a DREAM Writer for ID and Version. Each parameter can be up to 2,000 bytes long. Parameters must be character or zoned decimal, or, if packed, you must get the value from a packed variable in the calling program. You need to write a small "stub" program to convert the zoned decimal data to packed data and call the target program if one of the following situations is true:

- You call a program and pass in a packed decimal parameter where the calling program variable is zoned
- The calling program variable is packed and the called program variable is zoned
- You pass zeroes into a packed parameter

Programs J96GFK and P96GFK) are examples to convert the zoned decimal data to packed data and call the target program. In this case, the program specified in F96012 is J96GFK, not the name of the eventual target program. RPG does not allow the same RPG program to be called repeatedly at different levels in one job (recursively). Therefore, in J96GFK there is a way to call different versions of the same program. The pre-compiler commands P96GFKR and J96GFKR create the different versions and create multiple versions of P96GFK. J96GFK to determine which version to call at any one time. The sample code accepts ten parameters. CL does not accept a variable number of parameters (you must always call a CL program with the number of parameters it is coded to accept), so all generic exits that use the sample stub program (J96GFK) are defined to pass ten parameters, even though less parameters are actually needed for the eventual target program called.

The generic exits can be secured just like function key exits by using Function Key Security (P9612) on the security menu (G94, option 10). Specify the field name (#G01 through #G30) defined in the generic exit file.

When defining a generic exit program, you can default the DREAM Writer version to the version used by the calling program by typing &##VERS in the version column. For example, if the generic exit is being defined for V4211, and some users execute P4211 with a version of BRAZIL while others use version ZJDE0001, if you specify $##VERS for the called program, it will get passed BRAZIL or ZJDE0001 as the version, depending on which user executes it.

## Date Conversion (X0028)

X0028 was enhanced to make turn-of-century processing easier. It now accepts the following two additional parameters:

- The first new parameter (#EDAT2) is a ten-alpha edited output date including century.
- The second new parameter (#SIDT2) is an eight-alpha unedited input Gregorian date field including century.

If you do not know whether the user entered the date as MM/DD/YY or MM/DD/CCYY, scrub the date into an eight-byte field and put it into #SIDT2. X0028 determines whether a long or short date was entered. The new parameter list appears below:

```
C     *ENTRY      PLIST
C                 PARM  #SIDAT      6
C                 PARM  #EDAT       8
C                 PARM  #FFMT       7
C                 PARM  #TFMT       7
C                 PARM  #SEP        7
C                 PARM  #ERTST      1
C                 PARM  #CTRY       2
C                 PARM  #FJPN       1
C                 PARM  #TJPN       1
C                 PARM  #EDAT2      10
C                 PARM  #SIDT2      8
```

To use the new function, specify the optional $CTRY, #FJPN, and #TJPN parameters. Initialize #FJPN and #TJPN to blanks before calling X0028.

If you specify #EDAT2 without #SIDT2, X0028 uses the $CTRY field if not blank. If $CTRY is blank, and the input date is Gregorian, X0028 calculates the century. It looks at the default value for data item #CYR (currently set to 10). If the year is more than this value, it sets the century to 19- If the year is less than or equal to this value, it sets the century to 20. This logic remains unchanged. For example:

#SIDAT = 010108, $CTRY = blanks

Input format = *MDY

MM = 01, DD = 01, YY = 08

Default value for #CYR = 10, 08 is less than 10

X0028 returns 20 in $CTRY, and in year portion of $EDAT2 if supplied

#SIDAT = 123166, $CTRY = blanks

Input format = *MDY

MM = 12, DD = 31, YY = 66

Default value for #CYR = 10, 66 is greater than 10

X0028 returns 19 in $CTRY, and in year portion of $EDAT2 if supplied

If the input date is Julian, X0028 looks at the first byte of the date to determine the century. For example:

#SIDAT = 086234, $CTRY = blanks

Input format = *JUL

Century = 0, year = 86, Julian days = 234

0 + 19 = 20

X0028 returns 19 in $CTRY, and in year portion of $EDAT2 if supplied

#SIDAT = 186234, $CTRY = blanks

Input format = *JUL

Century = 1, year = 86, Julian days = 234

1 + 19 = 20

X0028 returns 20 in $CTRY, and in year portion of $EDAT2 if supplied

If $CTRY is provided, X0028 uses that for the century. X0028 returns the date in the new format in #SIDAT, and an edited date in #EDAT2.

If the programmer specifies #SIDT2 and the format is Gregorian, X0028 uses the century portion of the date specified in #SIDT2. It ignores anything specified in $CTRY, and returns the century from #SIDT2 in $CTRY. It puts the converted date into #SIDAT, #EDAT, #SIDT2, and #EDAT2.

The screen field used to enter the date must be scrubbed through C0012 before moving it into #SIDT2. If #SIDT2 is specified on the parameter list and it contains blanks or zeroes, X0028 returns with an error.

## Code Examples

### Example One

Passing long input date into X0028:

```
CSR   MOVE  L'12312001'      #SIDT2     8
CSR   MOVE  *BLANK           #EDAT2     10
CSR   CALL  'X0028'                           81
CSR   PARM                   #SIDAT
CSR   PARM                   #EDAT
CSR   PARM  '*MDY'           #FFMT
CSR   PARM  '*YMD'           #TFMT
CSR   PARM                   #SEP
```

```
CSR   PARM              #ERTST
CSR   PARM  '19'        #CTRY       2
CSR   PARM  *BLANK      #FJPN       1
CSR   PARM  *BLANK      #TJPN       1
CSR   PARM              #EDAT2      10
CSR   PARM              #SIDT2      8
```

The program returns:

```
#SIDAT = 011231
#EDAT = 01/12/31
$CTRY = 20
$EDAT2 = 2001/12/31 $SIDT2 = 20011231
```

### Example Two

Passing in short date and century, getting long date back:

```
CSR   MOVE  L'123101'   #SIDAT      6
CSR   MOVE  *BLANK      #EDAT2      10
CSR   CALL  'X0028'                       81
CSR   PARM              #SIDAT
CSR   PARM              #EDAT
CSR   PARM  '*MDY'      #FFMT
CSR   PARM  '*YMD'      #TFMT
CSR   PARM              #SEP
CSR   PARM              #ERTST
CSR   PARM  '19'        #CTRY       2
CSR   PARM  *BLANK      #FJPN       1
CSR   PARM  *BLANK      #TJPN       1
CSR   PARM              #EDAT2
```

The program returns:

```
#SIDAT = 011231
#EDAT = 01/12/31 $CTRY = 19
$EDAT2 = 1901/12/31
```

### Example Three

Passing in long/short date, getting long date back (user types 123101 into VDDATE):

```
CSR   MOVE  AVDDATE     @NM
CSR   EXSR  C0012
CSR   Z-ADD#NUMR        $DT         80
CSR   MOVE  $DT         #SIDT2      8
CSR   MOVE  *BLANK      #EDAT2      10
CSR   CALL  'X0028'                       81
CSR   PARM              #SIDAT
CSR   PARM              #EDAT
CSR   PARM  '*MDY'      #FFMT
CSR   PARM  '*YMD'      #TFMT
CSR   PARM              #SEP
CSR   PARM              #ERTST
CSR   PARM  '  '        #CTRY       2
CSR   PARM  *BLANK      #FJPN       1
CSR   PARM  *BLANK      #TJPN       1
CSR   PARM              #EDAT2
CSR   PARM              #SIDT2      8     (SIDT2 contains
'00123101')
```

The program returns:

```
#SIDAT = 011231
#EDAT = 01/12/31 $CTRY = 19
$EDAT2 = 1901/12/31 $SIDT2 = 19011231
```

### Example Four

Passing in Julian date, getting long date back:

```
CSR   MOVE  L'101365'         #SIDAT      6
CSR   MOVE  *BLANK            #EDAT2      10
CSR   CALL  'X0028'                             81
CSR   PARM                    #SIDAT
CSR   PARM                    #EDAT
CSR   PARM  '*JUL'            #FFMT
```

```
CSR   PARM  '*YMD'      #TFMT
CSR   PARM             #SEP
CSR   PARM             #ERTST
CSR   PARM  *BLANK     #CTRY       2
CSR   PARM  *BLANK     #FJPN       1
CSR   PARM  *BLANK     #TJPN       1
CSR   PARM             #EDAT2
```

The program returns:

```
#SIDAT = 011231
#EDAT = 01/12/31
$CTRY = 20
$EDAT2 = 2001/12/31
```

### Example Five

Passing Julian date into #SIDT2, getting long date back (database field contains 101365):

```
CSR   MOVE  LGLDGJ     #SIDAT      6
CSR   MOVE  *BLANK     #EDAT2      10
CSR   CALL  'X0028'                      81
CSR   PARM             #SIDAT
CSR   PARM             #EDAT
CSR   PARM  '*JUL'     #FFMT
CSR   PARM  '*MDY'     #TFMT
CSR   PARM             #SEP
CSR   PARM             #ERTST
CSR   PARM  *BLANK     #CTRY       2
CSR   PARM  *BLANK     #FJPN       1
CSR   PARM  *BLANK     #TJPN       1
CSR   PARM             #EDAT2
CSR   PARM             #SIDT2      8     (SIDT2 CONTAINS
'00101365')
```

The program returns:

```
#SIDAT = 123101 #EDAT = 12/31/01 $CTRY = 20
$EDAT2 = 12/31/2001 $SIDT2 = 12312001
```

## Calendar Window (P00CAL)

Few programs call POOCAL directly. Most programs call POOCAL through cursor-sensitive help (X96CCX). Currently, POOCAL returns the date as an eight-alpha edited value. The following displays the parameters:

```
CSR   *ENTRY      PLIST
CSR         PARM  PSFLDN      4
CSR         PARM  PSDATE      8
```

X96CCX returns the selected date left-justified in ##RVAL, and the programmer moves the first eight bytes into the date field in the program within subroutine SOOVL. For example:

```
CSR   ##FLDN      WHEQ  'VDDGJ '
CSR         MOVEL ##RVAL      VDDGJ
```

The new function in POOCAL returns 30 bytes of information instead of eight. The existing programs that call POOCAL directly will be changed for the new length. The "short" edited date will be in the first eight bytes as before. The "long" edited date will be in the last ten bytes of the parameter. The century will be in positions 11 and 12 of the parameter. The soft-coding copy member (IOOSC) was changed in A8.1 so you can access these subfields within ##RVAL directly.

```
I* Returned value
I  11  4 0 ##RVAL
I* Returned value - edited date incl. century line
MM/DD/YYYY
I  31  40 ##RV1
I* Returned value - century for date
I  21  22 ##RV2
```

The copy member for A7.3 or A7.1 cannot change, but you can access these portions of ##RVAL (or the longer parm from POOCAL) by using a MOVE instead of MOVEL to get the last ten bytes ("long" date), or two MOVEs to get the century as displayed below:

```
CSR   MOVE  ##RVAL      VDDATE      10
CSR   MOVE  L##RVAL     $TEMP       12
CSR   MOVE  $TEMP       $CTRY       2
```

Those programs calling POOCAL directly need to be changed as follows:

```
CSR   *ENTRY      PLIST
CSR               PARM         PSFLDN      4
CSR               PARM         PSDATE      30
CSR               MOVELPSDATE  VDDATE
```

# A9.1 Changes

## Electronic Signatures

This section provides information on the Electronic Signatures enhancement.

## Overview

Electronic signature is a regulatory requirement of the Food and Drug Agency (CFR21 Part 11). This regulation states that transaction history logs be maintained, and database transactions be authorized, and documented, at the time of the database transaction. An electronic signature is required for every database transaction that is encompassed by the regulation.

The Database Audit Manager (DBAM) incorporates user configurable database transaction logging, credential verification, and the mechanisms necessary to support documentation of transactions thereby assisting JD Edwards World customers with compliance.

In release A9.1, the electronic signature functionality was implemented into high priority programs based on customer input. You can use this document to implement this functionality in other JD Edwards World programs or custom programs.

## Definitions

Electronic signature is the process of verifying the credentials, or authenticity, of the user performing or authorizing the record add, change, or delete, at the time of the database transaction, including that authorization with the transaction in the transaction history log.

Signatures applied to single record updates are referred to as record level signatures. User verification is performed for each database transaction. However, to eliminate the continual prompting during transaction processes, such as subfile programs, it is permitted that one signature be applied to the transaction block. Signatures applied to multiple record updates are referred to as transaction level signatures.

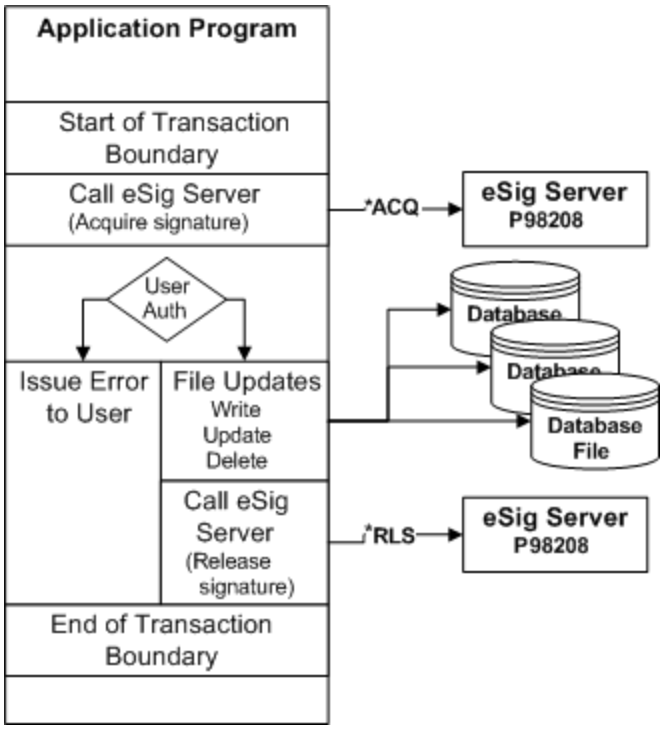
## Implementation

DBAM utilizes database trigger technology for recording transactions and implementing user credential authentication at the time of the database transaction.

## Two Levels of Signatures

### Record Level

Authorizations are implemented by database triggers configured *Before transactions are applied to the database. If user identity is not validated (authorization received), the trigger program cancels the database action. An I/O error is returned to the application program. The application program must respond to that error and convey it to the user. If authorization is received, the transaction is processed normally.

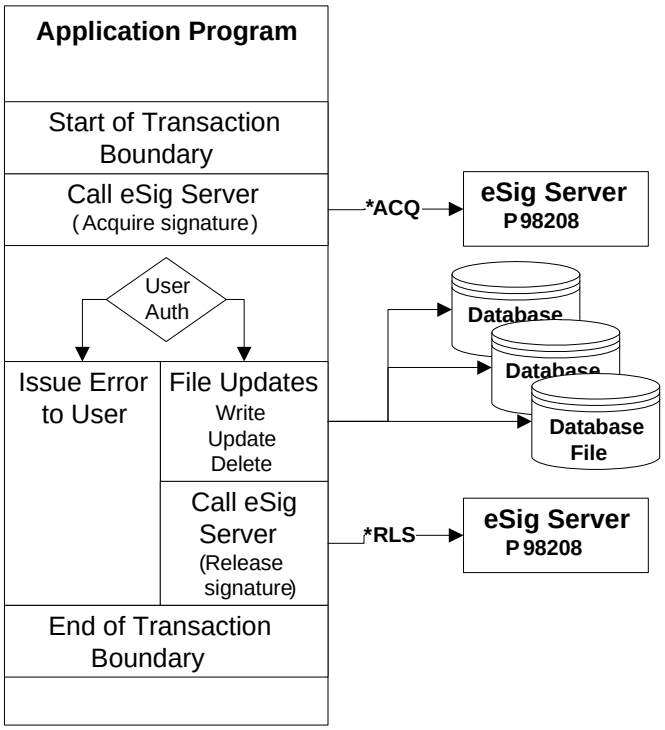



### Transaction Level

Transaction level authorizations are implemented by database triggers configured *After the transactions are applied to the database. Application programs determine the boundaries of the transaction, establish the authorization point, obtain the electronic signature, perform the database transactions, and then release the signature.

The application invokes and responds to the server accordingly. If user identity is not validated (authorization received), the program does not process the transactions and conveys the error to the user.

If authorization is received, the transactions are processed. The trigger programs apply the signature to the transactions as they are performed. Following completion of the transactions, the authentication server is called again to release the signature.




## Signature Servers

DBAM handles user authentication when configured at the record level. The DBAM trigger programs invoke the record level user authentication server.

For transaction processing applications, there are two transaction level authentications you can use:

### P98208

If your application falls under the umbrella of CFR21 regulations, use this DBAM transaction level authentication server. This server verifies that electronic signature is enabled on the primary transaction file before prompting for user authentication. If a DBAM audit is not configured on the primary database file, no prompting occurs.

### P00CKPWD

This server is external from DBAM. It is not dependent upon a DBAM configuration being setup before prompting occurs. It also has the following features:

- Backwards compatible to legacy implementations
- DREAM Writer ProcOpts control default behavior

- Optional parameters control behavior at runtime
- Can establish a DBAM compatible transaction level signature

### Entry Parameter Definitions

1. ReturnCode – User validation state (Required Parm)

   '1' = Users credentials verified

   '0' = Users credentials not verified

### Optional Parameters

2. Function

   *GET - validate and establish an eSignature

   *CLR - clear the current eSignature

3. UserId

   *Current - locks UserId prompt to current user (job)

   *AnyUser - opens UserId prompt for any UserId

   *ProcOpt - uses proc opt value from P00CkPwd/Zjde0001

4. eSig

   *Yes - establishes an eSignature upon validation

   *No - does not establish an eSignature

   *ProcOpt - uses proc opt value from P00CkPwd/Zjde0001

5. Pgm

   <name> - name of updating program to sign eSignature

6. eExp

   <text> - 40 char description to add to eSignature

See P00CKPWD program source for further documentation and usage.

## Application Program Code Samples

Two application program examples are listed in the next section. A single record update program impacted by record level signatures and a subfile transaction processor program that implements transaction level signatures. The single record update program, P4108, uses the same file server as the transaction processor P41080. Code segments vary from program to program even if you are handling the file I/O in your program. Programs vary in usage of subroutine S005 or S010 for updating database files.

### P4108 – Single Record Update

This program uses a file server for database I/O and checks the return code from the server to determine if the database action was successful. If the database action was not successful, it sets on an error indicator. Two more lines were added in this program, to set on additional error indicators and set up the error message. Later in the code, it checks if an error occurred and the screen and error are redisplayed to the user, otherwise, the display fields are cleared for the next transaction.

```
CSR               CALL 'XF4108 '
C*                ---- ---------
CSR               PARM        PS@@1
CSR               PARM        I4108
C*
CSR               SELEC
CSR   @@IOR       WHEQ 'ERR'
CSR               MOVE *ON          *IN99
CSR               SETON                          93  40
CSR               MOVE '1'          @MK,1
CSR               ENDSL
```

### P41080 – Transaction Processor

This is a subfile transaction processor. Since it falls under CFR21 regulations, it now includes the copy members for invoking the DBAM transaction authentication server. At the appropriate place in the program, before processing the subfile, the program calls the DBAM authentication server, and upon return, it checks the return code. If an error occurred, the program issues an error, exits the subroutine, and displays the error to the user. No transactions are processed. If the server successfully acquired a signature, the program begins processing the subfile transactions.

```
 * ------------------------------------------------------------
 * Check for and acquire electronic transaction eSignature...
 *
CSR               MOVEL'P41080'     ##PGM       P
CSR               MOVEL'F4108'      ##DBFN      P
CSR               MOVEL'*LIBL'      ##DBFL      P
CSR               MOVEL'*CHG'       ##DBFA      P
CSR               EXSR C98208
 *                ---- ------
```

```
CSR   *IN93       IFEQ '1'
CSR               SETON                         9340
CSR               MOVE '1'          @MK,1
CSR               GOTO END005
 *                ---- ------
CSR               ENDIF
 * ------------------------------------------------------------
```

Process the subfile transactions. After all transactions are processed, the authentication server is called again to release the signature.

```
 * ------------------------------------------------------------
 * Release the electronic transaction signature...
 *
C                 EXSR C98209
 *
 * ------------------------------------------------------------
```

The transaction process is now complete. The following is a Copy Module code to include the required subroutines for invoking the transaction level authentication server. These two modules must be included for transaction level signatures:

```
**************************************************************
 * Copy module to acquire a transaction eSignature.
 *
C/COPY JDECPY,C98208
 ******************************************************
 * Copy module to release a transaction eSignature.
 *
C/COPY JDECPY,C98209
 ******************************************************
```

## Example of Single Record Update

When the program updates the database file, it includes an error indicator on the I/O operation. After the file I/O, it checks for an error. If an error is received, it signals the condition to the user. Otherwise, it continues and resets for the next transaction.

```
C*************************************************************
C* SUBROUTINE S005  -   Scrub Input
C* ------------------------------
C*

CSR         S005        BEGSR
C*          ----        -----
C*
      Data validation code here
C* Update file. Monitor for I/O error.
C*
CSR                     SELEC
CSR         *IN21       WHEQ '1'
CSR                     WRITEIFILE              93
C*
CSR         *IN22       WHEQ '1'
CSR                     UPDATIFILE              93
C*
CSR         *IN23       WHEQ '1'
CSR                     DELETIFILE              93
CSR                     ENDSL
C*
C*Database I/O error. Maintain screen and issue error to user.
C*
CSR         *IN93       IFEQ '1'
CSR                     MOVE '1'          *INxx
CSR                     MOVE '1'          @MK,x
CSR                     ELSE
C*
C* Clear data fields for next transaction.
C*
CSR                     MOVE #FCLR        @@AID
CSR                     EXSR S001
C*                      ---- ----
a)    CSR                     ENDIF
C*-------------------------------------------------------------
```

```
CSR         END005      ENDSR
C*************************************************************
```

In the above example, the RPG error indicator and error message array position, depend upon your application requirements.

## Transaction Level Interfaces – Copy Modules

This section includes copies of the Copy Modules that must be included in transaction processor applications that are using the DBAM transaction level authentication server.

### RPG IV

C98209L - Copy module to invoke the transaction eSignature.

```
*************************************************************
 * SubRoutine C98208 - Check for and acquire an eSignature...
 * ----------------------------------------------------------

C     C98208`     Begsr
 *    ------      -----

C     *Like       Define      PsPgm       ##Pgm
C     *Like       Define      PsDbfn      ##Dbfn
C     *Like       Define      PsDbfl      ##Dbfl
C     *Like       Define      PsDbfa      ##Dbfa

C                 Call        'P98208'
 *                ----        ------
C                 Parm        '*ACQ'      PsFunc      4
C                 Parm                    PsRtn       1
C                 Parm        ##Pgm       PsPgm       10
C                 Parm        ##Dbfl      PSDbfl      10
C                 Parm        ##Dbfn      PSDbfn      10
C                 Parm        ##Dbfa      PSDbfa      4

 * Check return status from eSignature server...
```

```
C                   Select

 * Return code of *Zero means we have an authorization...

C     PsRtn         Wheneq      '0'
C                   Move        '1'         ##eSig      1

 * Return codes less than 4 indicate that eSignature is not
 * configured or not turned on...

C     PsRtn         Whenlt      '4'
C                   Move        '0'         ##eSig      1

 * Return codes greater than 3 are errors...

C     PsRtn         Whengt      '3'
C                   Move        '1'         *In93
C                   Endsl

C     E98208        Endsr
C*************************************************************
C98209L - Copy module to release the transaction eSignature.
*************************************************************
 * SubRoutine C98209 - Release transaction eSignature...
 * ---------------------------------------------------------

C     C98209        Begsr
 *    ------        -----

C     ##eSig        Ifeq        '1'
C                   Call        'P98208'
 *                  ----        ------
C                   Parm        '*END'      PsFunc      4
C                   Parm                    PsRtn       1

C                   Move        '0'         ##eSig
C                   Endif
```

```
C     E98209      Endsr
*****************************************************************
```

RPG III

C98208 - Copy module to invoke the transaction eSignature.

```
*****************************************************************
 * SubRoutine C98208 - Check for and acquire an eSignature...
 * -------------------------------------------------------------
 *
CSR   C98208      BEGSR
 *    ------      -----
 *
CSR   *LIKE       DEFN PSPGM        ##PGM
CSR   *LIKE       DEFN PSDBFN       ##DBFN
CSR   *LIKE       DEFN PSDBFL       ##DBFL
CSR   *LIKE       DEFN PSDBFA       ##DBFA
 *
CSR               CALL 'P98208'
 *                ----  ------
CSR               PARM '*ACQ'       PSFUNC            4
CSR               PARM              PSRTN             1
CSR               PARM ##PGM        PSPGM             10
CSR               PARM ##DBFL       PSDBFL            10
CSR               PARM ##DBFN       PSDBFN            10
CSR               PARM ##DBFA       PSDBFA            4
 *
 * Check return status from eSignature server...
 *
CSR               SELEC
 *
 * Return code of *Zero means we have an authorization...
 *
CSR   PSRTN       WHEQ        '0'
CSR               MOVE        '1'   ##ESIG            1
```

```
 *
 * Return codes less than 4 indicate that eSignature is not
 * configured or not turned on...
 *
CSR   PSRTN       WHLT        '4'
CSR               MOVE        '0'   ##ESIG
 *
 * Return codes greater than 3 are errors...
 *
CSR   PSRTN       WHGT        '3'
CSR               MOVE        '1'   *IN93
CSR               MOVE        '0'   ##ESIG
CSR               ENDSL
 *
CSR               ENDSR
*****************************************************************
C98209 - Copy module to release the transaction eSignature.
*****************************************************************
 * SubRoutine C98209 - Release transaction eSignature...
 * -------------------------------------------------------------
 *
CSR   C98209      BEGSR
 *    ------      -----
 *
CSR   ##ESIG      IFEQ        '1'
CSR               CALL 'P98208'
 *                ----  ------
CSR               PARM '*END'       PSFUNC            4
CSR               PARM              PSRTN             1
 *
CSR               MOVE        '0'   ##ESIG
CSR               ENDIF
 *
CSR               ENDSR
 *********************************************************
```

## Approvals

This section describes changes made to enable the approvals enhancement.

## New Physical Files and their Dependent Logical Files

| Rptg Code | Physical File | Dependent File | Description |
|---|---|---|---|
| 00A | F0030AW | | Approvals - Bank Transit Number Master File |
| 00A | F0030AW | F0030AWA | Approvals - Bank Transit Number Master File |
| 00A | F0030AW | F0030AWB | Approvals - Bank Transit Number Master File |
| 00A | F0030AW | F0030AWC | Approvals - Bank Transit Number Master File |
| 00A | F0030AW | F0030AWD | Approvals - Bank Transit Number Master File |
| 00A | F0030BW | | Approvals - Bank Transit Number Master File - Hist |
| 00A | F0030BW | F0030BWC | Approvals - Bank Transit Number Master File - Hist |
| 00A | F00A11 | | Approvals Transaction Header File |
| 00A | F00A11 | F00A11LA | Approvals Transaction Header File |
| 00A | F00A11 | F00A11LB | Approvals Transaction Header File |
| 00A | F00A11 | F00A11LC | Approvals Transaction Header File |
| 00A | F00A12 | | Approval Request File |
| 00A | F00A12 | F00A12LA | Approval Request File |
| 00A | F00A13 | | Assigned Approvers File |
| 00A | F00A13 | F00A13LA | Assigned Approvers File |
| 00A | F00A13 | F00A13LB | Assigned Approvers File |
| 00A | F00A14 | | Approver Substitute Cross Reference File |
| 00A | F00A14 | F00A14LA | Approver Substitute Cross Reference File |
| 00A | F00A14 | F00A14LB | Approver Substitute Cross Reference File |
| 00A | F00A17 | | Approval Rule Set File |
| 00A | F00A17 | F00A17LA | Approval Rule Set File |
| 00A | F00A17 | F00A17LB | Approval Rule Set File |
| 00A | F00A18 | | Approver Group File |
| 00A | F00A18 | F00A18LA | Approver Group File |

| Rptg Code | Physical File | Dependent File | Description |
|---|---|---|---|
| 00A | F00A19 | | Approval Route File |
| 00A | F00A19 | F00A19LA | Approval Route File |
| 00A | F00A19 | F00A19LB | Approval Route File |
| 00A | F00A20 | | Approval Schedule |
| 00A | F00A20 | F00A20LA | Approval Schedule |
| 00A | F00A21 | | Approvals Management Constants File |
| 00A | F00A21 | F00A21LA | Approvals Management Constants File |
| 00A | F00A22 | | Approvals Commitment Setup |
| 00A | F00A22 | F00A22LA | Approvals Commitment Setup |
| 00A | F00AC | | Approvals Transaction Detail - Change File |
| 00A | F00AC | F00ACLA | Approvals Transaction Detail - Change File |
| 00A | F00ACB | | Approvals Transaction Detail - Change File |
| 00A | F00ACB | F00ACBLA | Approvals Transaction Detail - Change File |
| 00A | F01014A | | Address Book - Diversity Status |
| 00A | F01014A | F01014AA | Address Book - Diversity Status |
| 00A | F01014A | F01014AC | Address Book - Diversity Status |
| 00A | F01014B | | Address Book - Diversity Status - History |
| 00A | F01014B | F01014BC | Address Book - Diversity Status - History |
| 00A | F01017A | | Approvals - Address Book Related Addresses |
| 00A | F01017A | F01017AA | Approvals - Address Book Related Addresses |
| 00A | F01017A | F01017AB | Approvals - Address Book Related Addresses |
| 00A | F01017A | F01017AC | Approvals - Address Book Related Addresses |
| 00A | F01017B | | Approvals - Address Book Related Addresses - History |
| 00A | F01017B | F01017BC | Approvals - Address Book Related Addresses - History |
| 00A | F01018A | | Approvals - Address Book Email / URL addresses |
| 00A | F01018A | F01018AA | Approvals - Address Book Email / URL addresses |
| 00A | F01018A | F01018AB | Approvals - Address Book Email / URL addresses |

| Rptg Code | Physical File | Dependent File | Description |
|---|---|---|---|
| 00A | F01018A | F01018AC | Approvals - Address Book Email / URL addresses |
| 00A | F01018B | | Approvals - Address Book Email / URL addresses - H |
| 00A | F01018B | F01018BC | Approvals - Address Book Email / URL addresses - H |
| 00A | F0101AW | | Approvals - Address Book Master |
| 00A | F0101AW | F0101AWA | Approvals - Address Book Master |
| 00A | F0101AW | F0101AWB | Approvals - Address Book Master |
| 00A | F0101AW | F0101AWC | Approvals - Address Book Master |
| 00A | F0101AW | F0101AWE | Approvals - Address Book Master |
| 00A | F0101AW | F0101AWJ | Approvals - Address Book Master |
| 00A | F0101AW | F0101AWR | Approvals - Address Book Master |
| 00A | F0101AW | F0101AWT | Approvals - Address Book Master |
| 00A | F0101AW | F0101AWU | Approvals - Address Book Master |
| 00A | F0101BW | | Approvals - Address Book Master - History |
| 00A | F0101BW | F0101BWC | Approvals - Address Book Master - History |
| 00A | F0111AW | | Approvals - Who's Who |
| 00A | F0111AW | F0111AWA | Approvals - Who's Who |
| 00A | F0111AW | F0111AWB | Approvals - Who's Who |
| 00A | F0111AW | F0111AWC | Approvals - Who's Who |
| 00A | F0111AW | F0111AWD | Approvals - Who's Who |
| 00A | F0111AW | F0111AWE | Approvals - Who's Who |
| 00A | F0111BW | | Approvals - Who's Who - History |
| 00A | F0111BW | F0111BWC | Approvals - Who's Who - History |
| 00A | F0115AW | | Approvals - Contact Phone Numbers |
| 00A | F0115AW | F0115AWA | Approvals - Contact Phone Numbers |
| 00A | F0115AW | F0115AWB | Approvals - Contact Phone Numbers |
| 00A | F0115AW | F0115AWC | Approvals - Contact Phone Numbers |
| 00A | F0115AW | F0115AWD | Approvals - Contact Phone Numbers |
| 00A | F0115BW | | Approvals - Contact Phone Numbers - History |
| 00A | F0115BW | F0115BWC | Approvals - Contact Phone Numbers - History |

| Rptg Code | Physical File | Dependent File | Description |
|---|---|---|---|
| 00A | F0116AW | | Approvals - Address By Date |
| 00A | F0116AW | F0116AWA | Approvals - Address By Date |
| 00A | F0116AW | F0116AWB | Approvals - Address By Date |
| 00A | F0116AW | F0116AWC | Approvals - Address By Date |
| 00A | F0116BW | | Approvals - Address By Date - History |
| 00A | F0116BW | F0116BWC | Approvals - Address By Date - History |
| 00A | F03015A | | Approvals - Customer Master Company/Business Unit |
| 00A | F03015A | F03015AA | Approvals - Customer Master Company/Business Unit |
| 00A | F03015A | F03015AC | Approvals - Customer Master Company/Business Unit |
| 00A | F03015B | | Approvals - Customer Master Company/Business Unit |
| 00A | F03015B | F03015BC | Approvals - Customer Master Company/Business Unit |
| 00A | F0301AW | | Approvals - Customer Master |
| 00A | F0301AW | F0301AWA | Approvals - Customer Master |
| 00A | F0301AW | F0301AWB | Approvals - Customer Master |
| 00A | F0301AW | F0301AWC | Approvals - Customer Master |
| 00A | F0301BW | | Approvals - Customer Master - History |
| 00A | F0301BW | F0301BWC | Approvals - Customer Master - History |
| 00A | F04015A | | Approvals - Supplier Master Company/Business Unit |
| 00A | F04015A | F04015AA | Approvals - Supplier Master Company/Business Unit |
| 00A | F04015A | F04015AC | Approvals - Supplier Master Company/Business Unit |
| 00A | F04015B | | Approvals - Supplier Master Company/Business Unit |
| 00A | F04015B | F04015BC | Approvals - Supplier Master Company/Business Unit |
| 00A | F0401AW | | Approvals - Supplier Master |
| 00A | F0401AW | F0401AWA | Approvals - Supplier Master |
| 00A | F0401AW | F0401AWB | Approvals - Supplier Master |
| 00A | F0401AW | F0401AWC | Approvals - Supplier Master |

| Rptg Code | Physical File | Dependent File | Description |
|---|---|---|---|
| 00A | F0401BW | | Approvals - Supplier Master - History |
| 00A | F0401BW | F0401BWC | Approvals - Supplier Master - History |

## Servers

| Rptg Code | Member ID | Description | Function |
|---|---|---|---|
| 00A | X00A11 | Transaction Server | RPGL |
| 00A | X00A111 | Transaction Workbench - Pop-Up Processing | RPGL |
| 00A | X00A112 | Transaction Submit | RPGL |
| 00A | X00A113 | Transaction Processor | RPGL |
| 00A | X00A115 | Approvals Transaction To SOA Submit | RPGL |
| 00A | X00A12 | Approval Request Server | RPGL |
| 00A | X00A121 | Approval Engine | RPGL |
| 00A | X00A13 | Assigned Approvers Server | RPGL |
| 00A | X00A14 | Approver Substitution Transaction Server | RPGL |
| 00A | X00A15 | Permanent Approver Substitution | RPGL |
| 00A | X00A17 | Approval Rule Set Transaction Server | RPGL |
| 00A | X00A18 | Approver Groups Transaction Server | RPGL |
| 00A | X00A19 | Approval Route Transaction Server | RPGL |
| 00A | X00A20 | Approval Schedule Transaction Server | RPGL |
| 00A | X00A21 | Approval Constants Transaction Server | RPGL |
| 00A | X00A21M | Approvals Constants Mode Server | RPGL |
| 00A | X00A22 | Approval Commitment Setup Transaction Server | RPGL |
| 00A | XF0030AW | Input/Output Server - Approvals Management (F0030) | RPGL |
| 00A | XF00A11 | Input/Output Server - F00A11 | RPGL |
| 00A | XF00A12 | Input/Output Server - F00A12 | RPGL |
| 00A | XF00A13 | Input/Output Server - F00A13 | RPGL |
| 00A | XF00A14 | Input/Output Server - F00A14 | RPGL |

| Rptg Code | Member ID | Description | Function |
|---|---|---|---|
| 00A | XF00A17 | Input/Output Server - F00A17 | RPGL |
| 00A | XF00A18 | Input/Output Server - F00A18 | RPGL |
| 00A | XF00A19 | Input/Output Server - F00A19 | RPGL |
| 00A | XF00A20 | Input/Output Server - F00A20 | RPGL |
| 00A | XF00A21 | Input/Output Server - F00A21 | RPGL |
| 00A | XF00A22 | Input/Output Server - F00A22 | RPGL |
| 00A | XF00AC | Input/Output Server - F00AC | RPGL |
| 00A | XF01014AW | Input/Output Server - Approvals Management (F01014) | RPGL |
| 00A | XF01017AW | Input/Output Server - Approvals Management (F01017) | RPGL |
| 00A | XF01018AW | Input/Output Server - Approvals Management (F01018) | RPGL |
| 00A | XF0101AW | Input/Output Server - Approvals Management (F0101) | RPGL |
| 00A | XF0111AW | Input/Output Server - Approvals Management (F0111) | RPGL |
| 00A | XF0115AW | Input/Output Server - Approvals Management (F0115) | RPGL |
| 00A | XF0116AW | Input/Output Server - Approvals Management (F0116) | RPGL |
| 00A | XF03015AW | Input/Output Server - Approvals Management (F03015) | RPGL |
| 00A | XF0301AW | Input/Output Server - Approvals Management (F0301) | RPGL |
| 00A | XF04015AW | Input/Output Server - Approvals Management (F04015) | RPGL |
| 00A | XF0401AW | Input/Output Server - Approvals Management (F0401) | RPGL |

# Cross Industry Systems

This section provides database and system changes for the Cross Industry systems.

## Database Changes

This chapter lists file level database changes for the Cross Industry systems.

### Summary of Database Changes

Double-click the following icon to open the Cross Industry Database Changes Summary.

### Changed Physical File Layouts

Double-click the following icon to open the Cross Industry Changed Physical Files Detail.

### New Logical Files for Existing Physical Files

Double-click the following icon to open the Cross Industry New Logical Files for Existing Physical Files.

### Changed Logical Files

Double-click the following icon to open the Cross Industry Changed Logical Files.

# Copy Member Changes (non-Database)

| Rptg Code | Copy Member | Func-tion | Description | Action | Release/ Cume |
|---|---|---|---|---|---|
| 00 | C0006 | COPY | File Server Key Lists - XF0006 | New | A91 |
| 00 | C00101 | COPY | Retrieve Currency Conversion Rate | Changed | A73CU09 |
| 00 | C00101 | COPY | Retrieve Currency Conversion Rate | Changed | A73CU10 |
| 00 | C00101 | COPY | Retrieve Currency Conversion Rate | Changed | A73CU11 |
| 00 | C00101L | CPYL | Retrieve Currency Conversion Rate | New | A73CU14 |
| 00 | C00101L | CPYL | Retrieve Currency Conversion Rate | Changed | A91 |
| 00 | C00102 | COPY | /COPY - Currency Conversion | Changed | A73CU09 |
| 00 | C00102 | COPY | /COPY - Currency Conversion | Changed | A73CU10 |
| 00 | C00102 | COPY | /COPY - Currency Conversion | Changed | A73CU12 |
| 00 | C00102L | CPYL | /COPY - Currency Conversion | New | A73CU14 |
| 00 | C00102L | CPYL | /COPY - Currency Conversion | Changed | A91 |
| 00 | C00103 | COPY | Retrieve Currency Conversion Method and Rate | Changed | A73CU12 |
| 00 | C00103 | COPY | Retrieve Currency Conversion Method and Rate | Changed | A73CU14 |
| 00 | C00103L | CPYL | Retrieve Currency Conversion Method and Rate | New | A73CU14 |
| 00 | C00103L | CPYL | Retrieve Currency Conversion Method and Rate | Changed | A91 |
| 00 | C00CUSL | CPYL | /COPY for functional server change user space | New | A73CU14 |
| 00 | C00CUSL | CPYL | /COPY for functional server change user space | Changed | A91 |
| 00 | C00RIXL | CPYL | /COPY for application retrieve errors from error index | New | A73CU14 |
| 00 | C00RIXL | CPYL | /COPY for application retrieve errors from error index | Changed | A91 |
| 00 | C00RUSL | CPYL | /COPY for application retrieve errors from error index | New | A73CU14 |

| Rptg Code | Copy Member | Func-tion | Description | Action | Release/ Cume |
|---|---|---|---|---|---|
| 00 | C00RUSL | CPYL | /COPY for application retrieve errors from error index | Changed | A91 |
| 01 | C0112 | COPY | File Server Key Lists - XF0112 | New | A91 |
| 01 | C0126 | COPY | File Server Key Lists - XF0126 | New | A91 |
| 03 | C73021 | COPY | /COPY Load Vertex Sales Tax Parameters - A/R | Changed | A73CU14 |
| 04 | C00LCTL | COPY | /COPY Country Server Control Structure | Changed | A91 |
| 04 | C00LCTLL | CPYL | /COPY Country Server Control Structure | New | A73CU14 |
| 04 | C00LCTLL | CPYL | /COPY Country Server Control Structure | Changed | A91 |
| 04 | C0411 | COPY | F0411 Category Code Algorithm | New | A91 |
| 04 | C73010 | COPY | /COPY Geocode Edit | Changed | A73CU07 |
| 04 | C73010L | CPYL | /COPY Geocode Edit | New | A73CU14 |
| 04 | C73010L | CPYL | /COPY Geocode Edit | Changed | A91 |
| 04 | C73013L | CPYL | /COPY Perform Tax Calculations - Vertex Mode | New | A91 |
| 04 | C7301L | CPYL | /COPY Determine if Vertex is active | New | A73CU14 |
| 04 | C7301L | CPYL | /COPY Determine if Vertex is active | Changed | A91 |
| 04 | C73023 | COPY | /COPY Load Vertex Sales Tax Parameters - A/P | Changed | A73CU11 |
| 04 | C73023 | COPY | /COPY Load Vertex Sales Tax Parameters - A/P | Changed | A73CU12 |
| 04 | C73023 | COPY | /COPY Load Vertex Sales Tax Parameters - A/P | Changed | A73CU13 |
| 04 | C73023 | COPY | /COPY Load Vertex Sales Tax Parameters - A/P | Changed | A91 |
| 04 | C73GEO | COPY | /COPY Retrieve GeoCode | Changed | A91 |
| 04 | C73GEOL | CPYL | /COPY Retrieve GeoCode - ILE | New | A91 |
| 09 | C0901 | COPY | Retrieve Account Master | Changed | A91 |
| 09 | C0903C | COPY | Determine Per No & Fiscal Year for 14 or 52 Per Accounting | Changed | A73CU09 |

| Rptg Code | Copy Member | Func-tion | Description | Action | Release/ Cume |
|---|---|---|---|---|---|
| 09 | C0906 | COPY | Edit Posting Edit Code from Account Master | Changed | A91 |
| 48S | C48096 | COPY | /COPY For Functional Server - Retrieve Billing Markup | Changed | A91 |
| 53 | C5302 | COPY | Compute the Best Amount | New | A73CU09 |

# Server Changes

Double-click the following icon to open the Cross Industry New and Changed Server Programs.

# Cross Industry Changes and Enhancements

This chapter describes changes made to Cross Industry systems.

## A8.1 Changes

## BCRC - Base Company Currency Code

This field was added to F0018, F0411, F0414, and F0911.

### Affected Programs

Any program that writes to files that contain the BCRC field (F0018, F0411, F0414, and F0911).

### Implementation Steps for F0911

- If currency is off, BCRC is blank
- If currency is on and if Ledger Type is CA, load BCRC with the transaction currency (the same value that is in CRCD)
- If currency is on and if Ledger Type is anything other than CA or AA, check on whether the Ledger Type is currency-specific (in this case, a currency code appears in the Special Handling Code field in the user-defined code table 09/LT), then:
    - If Ledger Type Currency is not blank, load BCRC with this currency, or

- If Ledger Type Currency is blank, load BCRC with the currency of the company on the record (the G/L account company)

**Note:** If you have used XS0013 (as most programs do) to determine the display decimals for the amount you are writing to the database, you can easily determine the proper values for the two conditions above.

Exception: Programs that create F0911 documents and that cross companies with different base currencies (those that result in multi-currency, intercompany settlements when posted). In this case, BCRC must be loaded in the AA ledger with the currency of the company that originated the transaction rather than the company on the record.

- For voucher distribution (Batch Type V), it will be the voucher company
- For invoice distribution (Batch Type I), it will be the invoice company
- For stand-alone journal entries (Batch Type G), it will be the company on the first line of the journal entry

| Multi-Currency Constant | Ledger Type | Currency-Specific Ledger Type | Multi-Currency-Intercompany Transaction | BCRC Value |
|---|---|---|---|---|
| Off | Any | N/A | N/A | Blank |
| On | CA | N/A | N/A | Transaction Currency |
| On | AA | N/A | No | Company Currency |
| On | AA | N/A | Yes | Originating Currency |
| On | Not CA or AA | Not Blank | N/A | Ledger Currency |
| On | Not CA or AA | Blank | N/A | Company Currency |

### Implementation Steps for All Other Files

- If currency is off, BCRC is blank
- If currency is on, load BCRC with the currency of the company on the record

| Currency | BCRC |
|---|---|
| Off | Blank |
| On | Company Currency |

## CRRM - Currency Mode - Foreign or Domestic Entry

The CRRM field holds the value of the original mode of entry for currency transactions. The field warns users when they attempt to change an entry in a mode other than the original mode of entry. Rounding errors can occur if an entry is

changed in a mode other than the one in which it was entered originally. The CRRM field is loaded only during an Add action. A Change action updates this field with the mode in which the change took place.

### Allowed Values

| Values | Explanation |
|---|---|
| F | When the foreign side of a multi-currency entry is made. Only applies when the multi-currency flag in the General Constants File (F0009) is set on. |
| D | When a domestic entry is made or when the domestic side of a multi-currency entry is made. Only applies when the multi-currency flag in the General Constants File (F0009) is set on. |
| blank | When the multi-currency flag in the General Constants File (F0009) is set off, the value in the CRRM field is always blank. |

## C0906 - Posting Edit Code from Account Master

Four new parameters were added to the C0906 copy module to validate Fixed Asset ID and Type Code.

### Affected Programs

The program is affected if the call to C0906 uses the new validations. You do not need to add these new parameters unless you want valid code to be checked for Fixed Asset ID and Type Code.

### New Call to C0906 for Validating Fixed Asset Posting Edit Code

```
MOVE GMFPEC #FPEC
MOVE GLASID #ASTID
EXSR C0906
```

### C0906 Parameter List

| Type | Name | Len | Value | Description |
|---|---|---|---|---|
| Input | #FPEC | 1 | | Fixed Asset Posting Edit Code |
| Input | #ASTID | 25 | | Fixed Asset ID |
| Output | #ERROR | 1 | F | Invalid Use of Fixed Asset Posting Edit Code |

### New Call to C0906 for Subledger Specified for Account

```
MOVE GLSBLT #SBLT
MOVE GMSTPC #STPC
EXSR C0906
```

### C0906 Parameter List

| Type | Name | Len | Value | Description |
|---|---|---|---|---|
| Input | #SBLT | 1 | | Subledger Type |
| Input | #STPC | 1 | | Type Code |
| Output | #ERROR | 1 | T | Disagreement between Type Code and Subledger Type |

## EFTJ - Date Effective (Deleted)

This field was moved from F04573 to F04572. Populate the field with the same value it had when it was in F04573. Change the file prefix from KI (F04573) to KK (F04572).

## HCO - Company Origination (HUB)

Originating Company (GLHCO) was added to the Account Ledger file (F0911) in the A8.1 release. This field indicates the origin company where the transaction originated. It must be populated.

### Affected Programs

Any program that writes records in the Account Ledger file (F0911).

### Rules for Populating Origination Company in the Account Ledger File

1. Journal Entries - G/L Company in the first line of the document.
2. Vouchers - Voucher Company.
3. Invoices - Invoice Company.
4. A/P Payments - Bank Account Company.
5. Cash Receipts - Bank Account Company.

6. A/R Adjustments - Adjustment Account Company.

7. A/R Spreads - First Spread Account Company created in the A/R pre-post.

## ICU - Batch Number

Populate F0414 ICU from F0413 ICU

## ICUT - Batch Type

Populate F0414 ICUT from F0413 ICUT.4

## NLS – National Language Support

In compliance with the IBM National Language Support program, certain characters in the JD Edwards World software were discontinued. The following user input characters were changed in programs:

| Before | Changed to |
|---|---|
| ## | && |
| # | & |
| @ | * |
| $ | Programmer's discretion |

The fields that used the variant characters in the JD Edwards World software were:

| Variant Characters | Fields |
|---|---|
| ## | DCT (Document Type) |
| # | ANI (Account Number - Input) |
| # | USER (User ID) |

### Examples

'##' for Document Type in Journal Entry, '#' for invalid accounts in Journal Entry, and '#' for User in General Journal Review.

## PKCO, PDCT, PO - Purchase Order Information

Purchase order information is now stored in the F04573- The paid voucher information populates the PKCO (Purchase Order Document Company), PDCT (Purchase Order Document Type), and PO (Purchase Order Number) fields. This information is also kept at the check level for contract processing.

## PSEL - Payments Print Selection Number

This field was expanded to eight characters, and it was enabled as a next number. This next number for Print Selection is system code 04, index number 09 (refer to the data dictionary in A9.1).

Key lists, data structure subfields, and work fields that reference this field should be expanded to eight characters.

## RC7 - File Line Identifier

This field was added to further identify the Check Control Number (CKC) within the Payment Group Control Number (HDC). This field must be set to '1' with each new check that is written and/or incremented for each new line within the check.

> **Note:** The keyed physical file for payment details (F04573) changed key lists. If your applications were accustomed to using this file for keyed access, you can use F04573LB. The LB logical has the key of the old F04573 keyed physical from past releases.

### Examples

| Payment Group Control Number (HDC) | Check Control Number (CKC) | File Line Identifier (RC7) |
|---|---|---|
| 1 | 1 | 1 |
| 1 | 1 | 2 |
| 1 | 2 | 1 |
| 1 | 2 | 2 |
| 1 | 2 | 3 |
| 2 | 1 | 1 |
| 2 | 1 | 2 |

## PH1 - Phone Number Field

The PHI field was expanded to 40 Characters to handle Internet addresses.

### Affected Programs

Any program that writes a phone number to a report or screen.

### Placement of Code for Reports

The code is instance-specific for each report program, but the bold test represents the underlying theme. The UDC table for Phone Type (01/PH) assigns code 'I' to Internet Address.

```
C*    If Internet address, skip edit and MASK.
C*
CSR   'I'   IFNE WPPHTP
CSR   *IN81 IFNE *ON
C*
CSR         MOVE *BLANK       #SINBR
CSR         MOVELWPAR1        #SINBR
CSR         MOVE T@AR1        #DTYP
CSR         MOVE W@AR1        #EWRD
CSR         MOVE E@AR1        #EC
CSR         MOVE F@AR1        #DSPD
CSR         MOVE G@AR1        #DATD
CSR         MOVE J@AR1        #ALR
CSR         MOVE ' '          #ECOR
CSR         MOVE ' '          #DCOR
CSR         EXSR C00161
C*          ---- ------
CSR   #ALR  IFEQ 'L'
CSR         MOVEL#SINBR       $AR1
CSR         ELSE
CSR         MOVEL#SINBR $AR1
CSR         ENDIF
C*
C*    Move to data structure #AH1 - Phone Number 1
C*
CSR         MOVE *BLANK       #SINBR
CSR         MOVELWPPH1        #SINBR
CSR         MOVE T@AR1        #DTYP
CSR         MOVE W@AR1        #EWRD
CSR         MOVE E@AR1        #EC
CSR         MOVE F@AR1        #DSPD
CSR         MOVE G@AR1        #DATD
```

```
CSR          MOVE J@AR1         #ALR
CSR          MOVEJ ' '          #ECOR
CSR          MOVEJ ' '          #DCOR
CSR          EXSR C00161
C*           ---- ------
CSR   #ALR   IFEQ 'L'
CSR          MOVEL#SINBR        $PH1
CSR          ELSE
CSR          MOVE #SINBR        $PH1
CSR          ENDIF
C*
CSR          ELSE
CSR          MOVELWPPH1         #AH1
CSR          ENDIF
```

## Placement of Code for Programs

The code is instance-specific for each program, but the bold test represents the underlying theme.

The boxed area in the following code, represents logic for output to a display field of only 20 characters in length. If the output is longer than 20 characters, do not display it because the user cannot maintain it. This helps eliminate the problem of truncation for change actions as well.

```
CSR          MOVE  *OFF         $2LNG
CSR   $2LNG  DOWNE*ON
CSR          MOVE  *BLANK       VDAR1
CSR          MOVE  *BLANK       VDPH1
CSR          MOVE  *BLANK       VDPHTP
CSR          MOVEL'WPKY00'      @@KLST
CSR          MOVEL'READ'        @@OPER
CSR          MOVE  'N'          @@LOCK
CSR          Z-ADD 2            @@KNUM
CSR          CALL  XF0115
CSR          PARM               PS@@1
CSR          PARM               I0115
CSR   @@IOR  COMP  'NE'                    82
CSR   @@IOR  COMP  'RL'                    99
```

```
CSR         CHECKWPPH1,21                79
CSR   *IN79 IFEQ  *OFF
CSR         MOVE  *ON         $2LNG 1
CSR         ENDIF
CSR         ENDDO

CSR   *IN82 IFEQ  '0'
CSR         Z-ADDWPRCK7       SHRCK7
C*  If Internet address, skip edit and MASK.
CSR   'I'   IFEQ  WPPHTP
CSR         MOVELWPPH1  VDPH1
CSR         ELSE
CSR   WPAR1 IFNE  *BLANKS
CSR         MOVE  *BLANK      #SINBR
CSR         MOVELWPAR1        #SINBR
CSR         MOVE  T@AR1       #DTYP
CSR         MOVE  W@AR1       #EWRD
CSR         MOVE  E@AR1       #EC
CSR         MOVE  F@AR1       #DSPD
CSR         MOVE  G@AR1       #DATD
CSR         MOVE  J@AR1       #ALR
CSR         MOVE  ' '         #ECOR
CSR         MOVE  ' '         #DCOR
CSR         EXSR  C00161
CSR   #ALR  IFEQ  'L'
CSR         MOVEL#SINBR       VDAR1
CSR         ELSE
CSR         MOVE  #SINBR            VDAR1
CSR         ENDIF
CSR         ENDIF
C*  Move to output – Phone Number
CSR   WPPH1 IFNE  *BLANKS
CSR         MOVE  *BLANK      #SINBR
CSR         MOVELWPPH1        #SINBR
CSR         MOVE  T@PH1       #DTYP
CSR         MOVE  W@PH1       #EWRD
CSR         MOVE  E@PH1       #EC
```

```
CSR          MOVE  F@PH1        #DSPD
CSR          MOVE  G@PH1        #DATD
CSR          MOVE  J@PH1        #ALR
CSR          MOVE  ' '          #ECOR
CSR          MOVE  ' '          #DCOR
CSR          EXSR  C00161
CSR   #ALR   IFEQ  'L'
CSR          MOVEL#SINBR        VDPH1
CSR          ELSE
CSR          MOVE  #SINBR             VDPH1
CSR          ENDIF
CSR          ENDIF
CSR          ENDIF
```

## TNST - Transit Number in the A/P Ledger

The Transit Number (RPTNST) must be populated for all programs that write records to the Accounts Payable Ledger file (F0411). If you use the Accounts Payable Voucher Functional Server (XT0411Z1) to write these records, disregard this section. The A/P Functional Server automatically populates the field correctly.

The Payee's Transit Number (RPTNST) was added to the A/P Ledger (F0411). Before writing new vouchers to the database, your program needs to access the Bank Account Information File (F0030) for the Payee Address Number on the voucher. The record that you retrieve for the payee must have a bank account type of 'V. If a bank account record is found, move the Transit Number (AYTNST) from the bank account to the Transit Number (RPTNST) on the voucher. If a bank account record does not exist, initialize the field to blanks, then continue. Example:

```
F*    File specifications
FF0030      IF    E     K     DISK
I*
 I*   Server parameter data structures.
 I*
 I/COPY JDECPY,I00PS@@
```

Prior to writing the F0411 records:

```
C*
 C*   Retrieve Transit Number from F0030
 C*
 CSR                    MOVELRPPYE        AYAN8
```

```
 CSR  RPBKTP    IFEQ *BLANKS
 CSR                    MOVEL'V '         AYBKTP
 CSR                    ELSE
 CSR                    MOVELRPBKTP       AYBKTP
 CSR                    ENDIF
 CSR                    MOVELRPCRCD       DSCRCD
 CSR                    MOVELRPDGJ        DSDTEF
 CSR                    MOVEL'04'         DSSYS
 C*
 CSR                    MOVE *BLANKS      PS@@
0CSR                    MOVELDSAY         KY@@
 CSR                    CALL 'XS0030'                98
 C*                     ---- --------

CSR                     PARM              PS@@
 CSR                    PARM              I0030
 CSR        RT@@        IFNE 'N'
 CSR                    MOVE *BLANKS      RPTNST
 CSR                    MOVELAYTNST       RPTNST
 CSR                    ENDIF
```

## C0411 - F0411 Category Code Algorithm

The C0411 copy module was created to map data to the F0411 voucher category codes.

C0411 maps data based on the user-defined code table specified in the data dictionary for a particular voucher category code. If the user-defined code table is related to an Address Book user-defined code table, the information for a particular Address Book number is inherited by the F0411 voucher category codes.

### Affected Programs

Any write to the F0411 requires C0411.

### Placement of C0411

Place the execution to C0411 directly before the write to the F0411.

```
C*
```

```
C*    Perform Category retrieval for F0411.
C*
CSR   EXSR C0411
C*
C*    Update/Write Voucher record
C*
CSR   WRITEI0411E
C*
```

Insert the I9800E module at the top of the program.

```
I*****************************************
I*   Copy member for server – X9800E
I/COPY JDECPY,I9800E
I*****************************************
```

Insert the C0411 copy module at the bottom of the program

```
C*****************************************
C* Copy Common subroutine – F0411 Category Code Algorithm
C/COPY JDECPY,C0411
C*****************************************
```

## Changes to I/O Server Functionality

The versioning feature of I/O Servers has proven to be ineffective. For Financial systems, the RPG that referred to the @@FMT field from the I/O Servers was removed. Therefore, you can remove any code that refers to the @@FMT field. Example:

Before:

```
CSR   MOVEL'A71'        @@FMT
CSR   MOVELWWAN8        ABAN8
CSR   MOVEL'CHAIN'      @@OPER
CSR   MOVEL'N'          @@LOCK
CSR   CALL ‘XF0101’
```

After:

```
CSR    MOVELWWAN8          ABAN8
CSR    MOVEL'CHAIN'        @@OPER
CSR    MOVEL'N'            @@LOCK
CSR    CALL ‘XF0101’
```

## Copy Module Naming Conventions

The names of many of the copy modules have changed in A9.1. The copy modules were changed, they no longer use the version suffix. Example:

Before:

```
I/COPY       JDECPY,I010171
```

After:

```
I/COPY       JDECPY,I0101
```

## Change Module Listing

| Obsolete Member | New Member | Description |
|---|---|---|
| I000661, I000671 | I0006 | Cost Center Master |
| I010171 | I0101 | Address Book Information |
| I012661 | ** | ** obsolete** |
| I011171 | I0111 | Who's Who Information |
| I011261 | I0112 | Flash Message Schedule |
| I011361 | ** | ** obsolete** |
| I011571 | I0115 | Phone Number Information |
| I011671 | I0116 | Mailing Address Information |
| I011771 | I0117 | Postal Code Transactions |
| I030171 | I0301 | Customer Master Information |
| I040171 | I0401 | Supplier Master Information |

| Obsolete Member | New Member | Description |
|---|---|---|
| I031171 | I0311 | A/R Ledger |
| I090161, I090171 | I0901 | Account Master |
| I090261 | I0902 | Account Balances |
| I091161, I091171 | I0911 | General Ledger (G/L) |
| I4809671 | I48096 | Cost Plus Markup Table |

### Exceptions

The copy modules I010161 will remain named the same. I010161 will only be used in connection with calls to XS0101LA. C00161 (Format Numeric Fields) is an example of a copy module that might be selected by mistake.

### Copy Module Source Code Conversion

Program P00CHGCOPY, which is a conversion program that is driven by DREAM Writer, changes the names of the copy modules in the Financial systems from A6.2 or A7.1 releases to A8.1 release. To run the P00CHGCOPY program, select Copy Module Rename - Financials from the A8 Conversion Utilities menu (BA6). The names of these copy modules previously contained suffixes that represented corresponding software versions. Now suffixes are no longer used. P00CHGCOPY renames the copy modules in the source code without the suffixes as follows:

| | | | |
|---|---|---|---|
| I0006 | I0113 | I0126 | I0901 |
| I0012 | I0115 | I0301 | I0902 |
| I0101 | I0116 | I0311 | I0911 |
| I0111 | I0117 | I0401 | I1011 |
| I0112 | | | |

If another job locks source code that must be changed, P00CHGCOPY prints an error report from program P00LOCKED.

### Payment Terms - Common Routines

The due date calculation module was amended to call X03021 (Compute Due Date for Advanced Payment Terms) in situations that use an advanced payment term. Three required parameters were added to this module to support enhanced term calculations.

X0302 was amended so that it determines whether a payment term that is passed is a simple one (one of the current payment terms) or an advanced one (one of the new payment terms). If the payment term is an advanced one, X03021 is called within the

existing X0302 module. Enhanced Payment terms (X03021) can calculate due date based on the invoice date, the G/L date, or the Service/Tax date. The three new parameters must be passed from the calling program to X0302, which passes the appropriate date to X03021.

The X0302 server currently has one optional and four mandatory parameters. However, the three new parameters were inserted before the current optional parameter, which results in one optional and seven mandatory parameters.

### Affected Programs

Any program that calls X0302 to calculate due date.

### X0302 Implementation Steps

Current call statement to X0302:

```
CSR   CALL 'X0302'
CSR   PARM        PSDDJ       6     due date(mandatory)
CSR   PARM *ZEROS PSDDDJ      6     discount dat(mandatory)
CSR   PARM RPPTC  PSPTC       3     payment term(mandatory)
CSR   PARM *BLANK PSERRM      4     error(mandatory)
CSR   PARM        PS0014            F0014 record (optional)
```

New call statement to X0302:

```
C*
C*    Calculate Due Date
C*
CSR   MOVE RPPTC  PSPTC
CSR   MOVE RPDIVJ RPDDJ
CSR   MOVE RPDIVJ PSDIVJ
CSR   MOVELRPDGJ  PSDGJ
CSR   MOVELRPDSVJ PSDSVJ
CSR   CALL 'X0302'
CSR   PARM        PSDDJ       6     due date (mandatory)
CSR   PARM*ZEROS  PSDDDJ      6     discount date(mandatory)
CSR   PARM        PSPTC       3     payment term(mandatory)
CSR   PARM*BLANKS PSERRM      4     (mandatory)
CSR   PARM        PSDIVJ      6     Invoice date(mandatory)
CSR   PARM        PSDGJ       6     G/L date(mandatory)
```

```
CSR   PARM        PSDSVJ      6     Service date(mandatory)
CSR   PARM        PS0014            F0014 record (optional)
```

Parameters PSDDJ, PSDIVJ, PSDGJ, and PSDSVJ must have a date value passed as input.

```
CSR   *ENTRY      PLIST
CSR   PARM        PSDDJ       6     net due date
CSR   PARM        PSDDDJ      6     disc due date
CSR   PARM        PSPTC             payment term
CSR   PARM        PSERR       4     error msg
CSR   PARM        PSDIVJ      6     Invoice date
CSR   PARM        PSDGJ       6     G/l date
CSR   PARM        PSDSVJ      6     Tax/Serv date
CSR   PARM        PS0014            F0014 record
```

## Retrieve Address Number (X0101) - Common Routines

X0101 was modified for performance to retrieve field information and descriptions, which eliminates successive calls to retrieve the descriptions.

### Current Call to X0101

```
CSR   MOVELVDTAX        PS#N8       P
CSR   CALL X0101
CSR   PARM ' '          PSSYM       1
CSR   PARM '3'          PSOMOD      1
CSR   PARM '3'          PSIMOD      1
CSR   PARM              PS#N8       21
CSR   PARM              PSSHT       80
CSR   PARM              PSLAB       20
CSR   PARM              PSTAX       20
CSR   PARM *BLANK       PSER        4
```

### New Call to X0101

```
CSR   MOVELVDTAX        PS#N8       P
CSR   CALL X0101
CSR   PARM ' '          PSSYM       1
CSR   PARM '3'          PSOMOD      1
CSR   PARM '3'          PSIMOD      1
CSR   PARM              PS#N8       21
CSR   PARM              PSSHT       80
CSR   PARM              PSLAB       20
CSR   PARM              PSTAX       20
CSR   PARM *BLANK       PSER        4
CSR   PARM I0101        PS0101      4 (OPTIONAL)
```

## E-mail Message Server (X00PPAT1)

### ZZ#ND# Send the Distribution to the Internet

This field informs the PPAT system that it is done with the Internet message, and that it is ready to be sent. For example, a message going to one AN8 number may be composed of one or more data dictionary glossary items. This field informs the PPAT system when no more items need to be accumulated, and that the message is ready to be sent.

In A8.1 release, changes were made to X00PPAT1 for Internet/PPAT functionality. As a result, any program that calls X00PPAT1 needs to load the ZZ#ND# field. This field exists already in the I00PPAT1 copy module, which is required by X00PPAT1. The only change required is to populate the ZZ#ND# field with a value of '1' before the call to X00PPAT1. The following example displays how to make this change.

**Note:** If your program uses ZZ#ND# already, you do not need to change anything in the call to X00PPAT.

Example:

Current call to X00PPAT1:

```
CSR          MOVE *ZEROS       ZZ#SRK
CSR          MOVE 'JDE4317'    ZZ#DD#
CSR          MOVELVTX020       DSDATA
CSR          MOVEL$PPTVS       PSVERS      10
CSR   PSVERS IFEQ *BLANKS
CSR          MOVEL'ZJDE0002'   PSVERS
CSR          ENDIF
```

```
CSR          MOVEL*BLANKS      PSERRM      4
CSR          CALL 'X00PPAT1'
CSR          PARM              DTASTR
CSR          PARM              DSDATA
CSR          PARM              PSVERS
CSR          PARM              PSERRM
```

New call to X00PPAT1:

```
CSR          MOVE *ZEROS       ZZ#SRK
CSR          MOVE 'JDE4317'    ZZ#DD#
CSR          MOVE '1'          ZZ#ND#
CSR          MOVELVTX020       DSDATA
CSR          MOVEL$PPTVS       PSVERS      10
CSR   PSVERS IFEQ *BLANKS
CSR          MOVEL'ZJDE0002'   PSVERS
CSR          ENDIF
CSR          MOVEL*BLANKS      PSERRM      4
CSR          CALL 'X00PPAT1'
CSR          PARM              DTASTR
CSR          PARM              DSDATA
CSR          PARM              PSVERS
CSR          PARM              PSERRM
```

## Architecture Changes

The Contract Billing Transaction Identifier program (X48120) was renamed to X52120 because this program is part of the Contract Billing product. The following two programs were affected by this change:

- Workfile Generation program (P48120)
- Workfile Transaction Re-Extension program (P481202)

## Changes to I/O Server (X48096) Functionality

The Cost Plus Markup Table Server (X48096) retrieves billable information from the Cost Plus Markup Table (F48096). This server is designed to recognize existing hierarchical search methodology to access the correct markup table.

## Cost Plus Markup Table Server (X48096)

The server follows the conventions of the Cost Plus Markup Table rules. The primary search sequence is as follows:

| Level | Search Criteria |
|---|---|
| 1 | Work Order |
| 2 | Work Order Class (UDC 00/W7) |
| 3 | Contract Number |
| 4 | Parent Contract Number |
| 5 | Customer Number (MCAN8O, WAAN8, or G4AN8) |
| 6 | Job/Cost Center |
| 7 | Job Class (RP11) |
| 8 | Not Used |
| 9 | System Default |

In each level, the record is checked for Effective Date. If a payroll transaction is being checked, the following sub-search is handled.

The first time through, the lookup is performed by using Job Class, Job Step, Pay Type, and Employee Number.

| Search Criteria | 1 | 2 | Search 3 | Search 4 | Level 5 | Level 6 | 7 | 8 | Data Item Name |
|---|---|---|---|---|---|---|---|---|---|
| Job Type | X | X | X | X | - | - | - | - | JBCD |
| Job Step | X | X | - | - | X | X | - | - | JBST |
| Pay Type | X | - | X | - | X | - | X | - | PDBA |
| Employee | X | X | X | X | X | X | X | X | AN8 |
| Home CC | - | - | - | - | - | - | - | - | HMCU |
| Cost Pool | - | - | - | - | - | - | - | - | RP12 |

The second time through, the sub-search is performed by using Job Class, Job Step, Pay Type, and either Home Cost Center or Cost Pool. AN8, HMCU, and RP12 are mutually exclusive, so the sub-search looks for a record with Home Cost Center or Cost Pool, or it looks for a record in which all three are blank.

| Search Criteria | Search Level | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| Job Type | X | X | X | X | X | X | X | X | X | X | X | - |

| | Search Level | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Job Step | X | X | X | X | X | X | - | - | - | - | - | - |
| Pay Type | X | X | X | - | - | - | X | X | X | - | - | - |
| Employee | - | - | - | - | - | - | - | - | - | - | - | - |
| Home CC | X | - | - | X | - | - | X | - | - | X | - | - |
| Cost Pool | - | X | - | - | X | - | - | X | - | - | X | - |

| Search Criteria | Search Level | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 |
| Job Type | _ | _ | _ | _ | _ | _ | _ | _ | _ | _ | _ | _ |
| Job Step | X | X | X | X | X | X | - | - | - | - | - | - |
| Pay Type | X | X | X | - | - | - | X | X | X | - | - | - |
| Employee | - | - | - | - | - | - | - | - | - | - | - | - |
| Home CC | X | - | - | X | - | - | X | - | - | X | - | - |
| Cost Pool | - | X | - | - | X | - | - | X | - | - | X | - |

Equipment is processed as follows:

| Search Criteria | Search Level | | | | | | | | | | Data Item Name |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | |
| Rate Group | - | X | X | - | X | X | - | - | - | - | (ACLO) |
| Equip # | x | - | - | - | - | - | - | - | - | - | (NUMB) |
| Rate Code | - | X | - | X | X | - | X | - | - | - | (ERC) |
| Employee | _ | _ | _ | _ | _ | _ | _ | _ | _ | _ | (AN8) |
| Home CC | _ | _ | _ | _ | _ | _ | _ | x | - | - | (HMCU) |
| Cost Pool | - | X | X | X | - | - | - | - | X | - | (RP12) |

All other transactions are processed as follows:

| Search Criteria | Search Level | | | | | | | | | Data Item Name |
|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | |
| Employee | X | X | X | X | X | - | - | - | - | (AN8) |
| Home CC | X | - | X | - | - | X | - | - | - | (HMCU) |

| | Search Level | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Cost Pool | - | X | - | X | - | - | X | - | - | (RP12) |
| Job Step | X | X | - | - | - | - | - | X | - | (JBST) |

## X48096 Server Implementation

Many parameters need to be passed to the server to ensure correct implementation. In addition, several calculations need to occur to adjust the billing rate to make it exactly as the Cost Plus Markup Table directs it. For this purpose, two copy modules (I48096 and C48096) were created to perform the calculations. To use the copy modules, include these modules as normal copy modules. Populate the following fields with all information known, and leave the others fields blank.

| Field | Description | Value |
|---|---|---|
| STMENT | Time Entry Flag | Set to 1 if flag comes from time entry |
| #CGTYP | Generation Type | Set to 1 for Invoice Rates, 2 for Independent Revenue Rates, and 3 for component markup (users external to Contract Billing should use 1) |
| #CDCT | Document Type | The type of document being examined (for example, T2, T4, and so forth) |
| #CTBDT | Work Date | This is in Julian format |
| #CAN8O | Not Used | Do not fill |
| #CMCU | Business Unit | Database format |
| #CRP11 | Job Class | Do not fill – defaulted from Business Unit |
| #CJBCD | Job Code | Fill from JBCD |
| #CJBST | Job Step | Fill from JBST |
| #CPDBA | Pay Type | Fill from PDBA |
| #CAN8 | Employee Number | |
| #CHMCU | Home Business Unit | |
| #CRP12 | Cost Pool | Do not fill – defaulted from Business Unit |
| #CACL0 | Rate Group | Do not fill – defaulted from Equipment Master |
| #CNUMB | Equipment Number | |
| #CERC | Equipment Rate Code | |
| #COBJ | Object Account | User to derive category codes |
| #CSUB | Subsidiary Account | User to derive category codes |

| Field | Description | Value |
|---|---|---|
| #CSBL | Subledger | User to derive category codes |
| #CSBLT | Subledger Type | User to derive category codes |
| #CCRCD | Currency Code | The currency code must match exactly for a rate to be derived |

When these fields are filled, place the cost amount into variable $#AA (29 9) in actual amount. The program returns $#AA exactly as entered and returns the value $RATE with the billing rate to be used. $RATE is set to 0 if any errors are encountered. Example Code:

Include modules:

```
F*
F*    Copy Composite Member for Common Subroutine – C0030
F*
F/COPY  JDECPY,C48096
F*
I/COPY  JDECPY,I48096
CSR   MOVE ‘1’          $TMENT      Time Entry On
CSR   MOVE ‘T4’         $CBDCT      Doc Type
CSR   MOVE ‘1’          $CBTYP      Gen Type
CSR   MOVE YTDWK        #CBDWK      Work Date
CSR   MOVE YTMCU        #CBMCU      Bus. Unit
CSR   MOVE SFJBCD       #CBJCD      Job Code
CSR   MOVE SFJBST       #CBJST      Job Step
CSR   Z-ADDYTPDBA       #CBPDB      Pay Type
CSR   MOVE YTAN8        #CBEMP      Emp #
CSR   MOVE YAHMCU       #CBHMC      Home Bus Unt
CSR   MOVE YTGOBJ       #CBOBJ      Recharge Obj
CSR   MOVE YTGSUB       #CBSUB      Recharge Sub
CSR   MOVE YTSBL        #CBSBL      Subledger
CSR   MOVE YTSBLT       #CBSBT      Subledger Ty
CSR   Z-ADD*ZEROS       #CBNMB      Equipment
CSR   Z-ADD*ZEROS       $#AA        No Cost markup
```

Call X48096 Service Billing Cost Plus Markup Table Functional Server.

```
CSR   EXSR        C48096
```

If no billing rate was retrieved from the call to X48096 Functional Server, then issue an error message.

```
CSR   $RATE       IFEQ  *ZEROS
CSR               MOVE '1'          @MK,26
CSR               SETON                   5393
CSR               ELSE
CSR   $RATE       MULT  #@PBRT      YTPBRT
CSR               ENDIF                          YKBCMA = '2'
```

Add C0030 copy module for subroutine:

```
C*
C*    Copy Common Subroutine - Cost/Markup Table Interface
C*
C/COPY JDECPY,C48096
C*
C*    Copy Common Subroutine - Cost/Markup Table Interface
C*
C/COPY   JDECPY,C0030
```

## A9.1 Changes

## Change UDC for State Codes to New File

### Objective

Because some countries have the same State Codes used in other countries, a new State-Province/Country file was created by converting the existing UDC 00/S - State Codes to a new file F0075, keyed by State and Country code.

This new file could potentially affect programs that currently use the Data Dictionary items: ADDS, IRIF, IRPN, ROO, SHST, SLST, and BUFF because all use the UDC 00/S as its Data Edit Rule.

You need to identify any custom code where the UDC 00/S is used and replace your code with a call to XF0075. Identify the program type and follow the general instructions below for the program type. It may be helpful to pre-open the F0075 file if it is heavily used.

**Note:** These are general guidelines and you may have to modify the suggested code change to fit your particular code.

### Interactive -- Flat Screen Programs

Input specs – If program is using X0005 to edit DD State codes.

```
I/COPY JDECPY,I00XFSRV                                    ADD
I*
I*    Data Structure for - State/Country file        ADD
I*
I/COPY JDECPY,I0075
```

*S004* - Look for the DD item ADDS, ROO, IRIF, IRPN, SHST, SLST, or BUFF being edited in X0005 and REMOVE:

```
C*R                 CLEARI0005U                          REMOVE
C*R                 MOVE *BLANK    $ERTST  1
C*R                 MOVELS@ADDS    #USY
C*R                 MOVE R@ADDS    #URT
C*R                 MOVE ALADDS    #UKY
C*R                 CALL 'X0005 '               81
C*                  ---- --------
C*R                 PARM           I0005U
C*R                 MOVE #UERR     *IN81
 *
```

If the screen has a State description field after the State Code, move it into a VC000x field. If there is no Country Code on the screen, Z-ADD 1 to @@KNUM, so the server uses only the state code to chain to F0075:

```
C                   MOVELVDADDS    SCADDS      ADD
C                   MOVELVDCTR     SCCTR
C                   MOVEL'SCKY00'  @@KLST
C                   Z-ADD2         @@KNUM
C                   MOVEL'CHAIN'   @@OPER
C                   MOVEL'Y'       @@LOCK
C                   CALL 'XF0075'
```

```
 *                    ---- --------
C                     PARM           PS@@1
C                     PARM           I0075
 *
C           @@IOR     COMP 'NF'                     81
CSR         *IN81     IFEQ '0'
CSR                   CLEAR@UA
CSR                   MOVEASCDL01    @UA
CSR                   Z-ADD21        #OUTLG
CSR                   EXSR C9822
C*                    ---- -----
CSR                   MOVEA@UB       VC0001
CSR                   ENDIF
```

If the screen has a Country field on the screen, the P0075W window returns the country value that you pick, back in the ##RDSC field (rather than the state description). You may move the State and the Country values selected from P0075W into both screen fields. If you have no Country field on your screen, work only with the ##RVAL and ignore the ##RDSC.

```
CSR                           MOVEL##RVAL    VDCTY1
C*
CSR         ##FLDN    WHEQ 'VDADDS  '
CSR                           MOVEL##RVAL    VDADDS
CSR                           MOVEL##RDSC    VDCTR         ADD
C*
CSR         ##FLDN    WHEQ 'VDCOUN  '
CSR                           CLEAR@UA
```

*S005* – EDIT the State DD code & REMOVE X0005 UDC edit

```
C*
C*    Edit from User Defined Codes - State        REMOVE
C*
C*R         R@ADDS    IFNE *BLANK
C*R                   CLEARI0005U
C*R                   MOVELS@ADDS    #USY
C*R                   MOVE R@ADDS    #URT
C*R                   MOVE ALADDS    #UKY
C*R                   CALL 'X0005'                  81
```

```
C*                  ---- -------
C*R                 PARM           I0005U
C*R         #UERR   IFEQ '1'
C*R                 MOVE '1'       @MK,09
C*R                 SETON                      5593
C*R                 ENDIF
C*R                 ENDIF
 *
```

Add the following code. If there is no Country code on the screen, move *BLANKS into SCCTR as it is a valid default Country code.

```
C                   MOVELVDADDS    SCADDS     ADD
C                   MOVELVDCTR     SCCTR
C                   MOVEL'SCKY00'  @@KLST
C                   Z-ADD2         @@KNUM
C                   MOVEL'CHAIN'   @@OPER
C                   MOVEL'Y'       @@LOCK
C                   CALL 'XF0075'
 *                  ---- --------
C                   PARM           PS@@1
C                   PARM           I0075
 *
C           @@IOR   COMP 'NF'                    98
C           *IN98   IFEQ '1'
```

Check the DSPF code for the program to see what indicators to use if an error occurs for the State/Country code. Highlight both the State and Country code fields (if Country code exists). Add the new DD Error here and in S999.

```
C                   MOVE '1'       @MK,32                ADD
C                   SETON                      555793
C                   ENDIF
```

*S999* – Add the new Error

```
CSR                 MOVE '562K'    EMK,32    Inv St/Ctr  ADD
```

### Interactive -- Subfile Programs

*Input Specs*

Same as Flat screen programs if X0005 is being used to edit DD State codes.

*S00VL*

Same as Flat screen programs. Check to see that ##RVAL is being moved into the SFxxxx state field and the VDxxxx state field when applicable.

*S004*

Same as Flat screen programs. If the DD state field is not being edited by X0005, then no changes are needed. When X0005 retrieves a state description, remove and edit with the XF0075 file server, using SFxxxx fields instead of VDxxxx fields.

*S005*

Same as Flat screen programs. Remove X0005 code when used to edit the State DD fields and add XF0075 file server, using SFxxxx fields instead of VDxxxx fields. Check the DSPF source to determine which state and country fields to highlight when an error exists. Add new error code.

*S999*

Same as Flat screen programs. Add new error code 562K.

### Batch -- Report Programs

*Input Specs*

Same as Flat screen programs if X0005 is being used to edit DD state codes.

Search for X0005 editing DD state codes and remove. Add the XF0075 file server:

```
C*                                                          REMOVE
C*    Edit from descriptive titles - Property Tax State
C*
CSR         $$ADDS    IFNE *BLANK
CSR                   MOVEL$$ADDS    FAADDS
C*R         R@ADDS    IFNE *BLANK
C*R                   CLEARI0005U
C*R                   MOVE ' '       $ERTST
C*R                   MOVELS@ADDS    #USY
C*R                   MOVE R@ADDS    #URT
C*R                   MOVE FAADDS    #UKY
C*R                   CALL 'X0005 '              81
C*                    ---- --------
C*R                   PARM           I0005U
```

```
C*R                    MOVE #UERR     *IN81
```

If a Report program is using X0005, it will usually be to edit the State/Country code and print an error if not valid or to retrieve the state description (SCDL01).

To retrieve the description, use the existing code that loads the description (probably from VTXxxx) after you add the XF0075 code below. Note whether or not the report program has a Country code and load SCCTR accordingly.

```
CSR                     MOVELFAADDS    SCADDS      ADD
CSR                    MOVEL'SCKY00'  @@KLST
CSR                     Z-ADD1         @@KNUM
CSR                     MOVEL'CHAIN'   @@OPER
CSR                     MOVEL'Y'       @@LOCK
CSR                     CALL 'XF0075'
 *                     ---- --------
CSR                     PARM           PS@@1
CSR                     PARM           I0075
 *
CSR           @@IOR     COMP 'NF'                    81
```

## Action Code Security

### Objective

The Action Code Security and Search Type Security processes currently work independently.

To have both Action Code Security and Search Type Security work in conjunction with each other, a new file (F0103) was created and a new program (X0103) written to validate user security with both Action Code and Search Type. This program allows you to set up combined search type and action code security by user ID, similar to the action code security program (P00031). This can be done by an individual user, user groups, or for *PUBLIC. For example: *PUBLIC may be given access to Inquire on Customers but not allowed to Add, Change, or Delete customers. Individual users may then be granted the ability to Add and/or Change and/or Delete a customer (or any other search type, according to your needs). You may also restrict individual users from inquiring on certain search types. It is worth noting that Inquiry is now included as one of the actions you may secure with this new feature.

The new program is set up to accept three Input Parms (User ID, Search Type, and Action Code) and one Output Parm (Error Flag).

The scope affects all programs that currently use Search Type Security. The old search type security can be recognized by searching custom programs for CCCRYC. Replace all AT1 Security Checks that use the UDC tables 94/x (where x is a Search Type such as C, V, E, and so on.) with a call to the new X0103 program. This program searches the new file F0103 with the User ID, Search Type, and Action Code to determine if the Action requested by this User for this Search Type is Valid/Invalid. If the error flag value returned to your program is equal to '1', it means the requested action is invalid for the current user and search type.

Program P0103 was added to maintain the F0103 file. This program is accessible from G94, option 6 (Name Search Type). The front-end program (P000901) is still used to define if you use search type security, but the UDC tables 94/C, 94/V, and so on, were replaced by F0103. Function key F5 from P000901 takes you to P0103, where you can set up the new security.

## Company/Business Unit Defaults

### Objective

To provide a method to maintain and access default information at a lower level than the defaults currently found in the F0301/F0401/F0101 files, allowing you to set up Customer Master defaults for Sales Orders and Invoices at a Company and/or Business Unit level. The same new functionality applies to Supplier Master defaults for Purchase Orders and Vouchers.

**Note:** From this point forward, there is going to be reference only to the Customer Master, Customer Co/BU defaults, and so on. The same information and procedures are available for A/P, just substitute 04 for 03, Supplier for Customer, voucher for invoice, PO for SO, and so on. The Supplier Master Co/BU Default program is P01154 and the Purchasing Instructions are P43063.

### Overview

JD Edwards World received requests for the ability to maintain and access default information at a lower level than what they had at the Customer level, when entering invoices and SO.

A new file was created, F03015, keyed by AN8, CO, MCU. You may update this file via P01153, which is accessible from the Customer Master P01053 via PO for auto display or FK. There are additional screens for the Billing Instruction Information, P42063/P420631 accessible from P01153 via PO or FK. You may set up this additional default/control information by AN8/CO/MCU, AN8/CO or AN8/MCU. This file and any input is optional.

The criteria for the fields made available for override from the F0301/F0401/F0101:

- Fields not updated by any other programs (example: Amount Invoiced YTD, Number of Reminders Sent, Credit Message)

- Need to fit at more than just a Customer Master level (example: Amount Currency works with the Amount fields, which are only maintained at the Customer Master level, Dun and Bradstreet Number is assigned per customer)

The Category Codes AC01-10 and Special/Factor Payee (originates from the F0101 file) were included.

### XF03015/XF04015

A regular I/O Server was created for the F03015/F04015 file, XF03015/XF04015. If you know the key and the record you are going to work with, you may use this server. It is necessary that you use this server if you plan to Add, Change, or Delete an XF03015/XF04015 record.

### XS03015/XS04015

This is a new server used when you need to retrieve the correct override information for your invoice or SO. This server just retrieves the record for you.

Input Parm:

```
PS@@
```

Output Parm:

```
PS@@
PS0301 (the F0301 record)
```

The XS03015 file server uses data structure DSC3D, defined in copy module I03015X. This data structure has the specific input parms necessary to locate the Company/Business Unit record you want to retrieve, as well as return the F0101 field values.

```
IDSC3D       DS                              100
I*
I*    INPUT Parms
I*      Mode  (Future Use)
I                                        1   1 #C3MDE
I*      Address
I                                        2   90#C3AN8
I*      Company
I                                       10  14 #C3CO
I*
```

```
I*      Business Unit
I                                        15  26 #C3MCU
I*
I*    OUTPUT Parms
I*      Special/Factor Payee (from f0101)
I*
I                                       27  340#C3A85
I*
I*     Category Codes (from f0101)
I*
I                                       35  37 #C3A01
I                                       38  40 #C3A02
I                                       41  43 #C3A03
I                                       44  46 #C3A04
I                                       47  49 #C3A05
I                                       50  52 #C3A06
I                                       53  55 #C3A07
I                                       56  58 #C3A08
I                                       59  61 #C3A09
I                                       62  64 #C3A10
I*
I*  Status Flag  (blank = Defaults found, 1 = No defaults foun
I*
I                                       65  65 #C3STS
I*
I*      Error Flag   (blank = OK, 1 = no F0301 found/returned)
I*
I                                       66  66 #C3ERR
```

Identify any custom programs that retrieve information from F0301/F0401 or A/B fields AC01-10 or AN85. To minimize the amount of work you have to do after calling this server, the F0301/F0401 record returns with any overrides located already overlaid on top of the corresponding F0301/F0401 fields. This change allows you to substitute the current calls to F0301 in your program to the XS03015 server and get the entire F0301 record returned with any overrides already in place. When you use the A5 fields later in the program, they are correct. The exception is the Address Book fields, AC01-10 and An85. These come back in the return parm data structure and you have to manually move these over the ABAC01-10 fields or directly into RPAC01-10 fields, and so on.

Most of the data structure is intuitive. The following is an explanation of other input/output fields:

#*C3MDE* = Blank for now. Reserved for future use.

#*C3STS* = This is a '1' if no overrides were located and we are just returning the F0301. The main purpose of this is to know whether you should use the F0101 overrides. If none were found, all fields are blank and you do not want to overlay.

#*C3ERR* = This is the error flag. '1' indicates that no Customer Ledger F0301 was found for the address.

Within this server, we first retrieve the F0301 record, then any override records in the following order:

1. Search for address, company, and business unit requested.
2. Search for address and business unit.
3. Search for address and company.

If no F03015 record is found, return the F0301.

Pass in the Company and/or the Business Unit, it may be necessary to relocate the F0301 retrieval in your programs and possibly retrieve it more than once if subfile records have different companies/business units associated with them.

This server was implemented in a number of programs, with the A/R Functional Server being one. The following code is the code from that program, showing how this new server retrieves the Overrides for a particular invoice. After retrieving AB information, the system overlays with any F0101 overrides found.

```
C*
C*  Retrieve Customer Master. Moved to this location so we can
C*  get default info for F0301 as well as select AB defaults
C*  if using company/bus unit defaults before we assign them.
C*  The I0301 image is returned and if override defaults were
C*  located in F03015, they have already been moved into the
C*  F0301 record so no extra proc is necessary. However, if an
C*  override record was found, we need to move those AB fields
C*  from the returned data structure into corresponding AB
C*  fields.
C*
CSR                     CLEARI0301
CSR          $$AN8      IFEQ *BLANKS
CSR                     CLEARPS@@
CSR                     CLEARDSC3D
CSR                     MOVE RPAN8      #C3AN8
```

```
CSR                   MOVE RPCO      #C3CO
CSR                   MOVE RPMCU     #C3MCU
CSR                   MOVELDSC3D     PS@@
CSR                   CALL 'XS03015'              87
C*                    ---- ---------
CSR                   PARM           PS@@
CSR                   PARM           I0301
CSR                   MOVELPS@@      DSC3D
CSR         *IN87     IFEQ *ON
CSR         #C3ERR    OREQ '1'
CSR                 MOVEL'3490'    $$AN8          error messag
CSR                   SETOF                     8799
CSR                   ENDIF
CSR                   ENDIF
C*
C*    Save amount currency
C*
CSR                   MOVELA5CRCA    $SCRCA
C*
C*    Check if suspend A/R is set
C*
CSR         A5HDAR    IFEQ 'Y'
CSR         A5HDAR    OREQ '1'
CSR                 MOVEL'1837'    $$AN8         error message
CSR                   ENDIF
C*
C*    If overrides were located, replace AB data w/override AB
C*    data, otherwise, leave AB values as they are from F0101.
C*
CSR         #C3STS    IFNE '1'
CSR         $$AN8     ANDEQ*BLANKS
CSR                   MOVE #C3A85    ABAN85
CSR                   MOVE #C3A01    ABAC01
CSR                   MOVE #C3A02    ABAC02
CSR                   MOVE #C3A03    ABAC03
CSR                   MOVE #C3A04    ABAC04
CSR                   MOVE #C3A05    ABAC05
```

```
CSR                     MOVE #C3A06    ABAC06
CSR                     MOVE #C3A07    ABAC07
CSR                     MOVE #C3A08    ABAC08
CSR                     MOVE #C3A09    ABAC09
CSR                     MOVE #C3A10    ABAC10
CSR                     ENDIF
```

When trying to determine if your program needs to implement this server and retrieve CO/BU overrides, consider the following:

- If your program retrieves and uses fields from the F0301/F0401, check to see if there are any of those A5 and A6 fields also in the F03015/F04015. Several programs retrieved the F0301/F0401 to determine if there was a Hold Payment or Hold Invoice flag set. Those hold fields are not maintained at the CO/BU level, therefore, retrieving just the Customer Master is sufficient. There is no need to search for overrides.
- If your program does not retrieve the F0301/F0401 but does use the ABAC01-10 or ABAN85, does it need to look for the overrides for those fields.
- If your program deals with invoices/vouchers and sales orders/purchase orders, check to see if there is a Company and/or Business Unit available to help you retrieve an override. For example, the Customer Ledger Inquiry defaults the Currency Code into the header of the screen from the F0301 for the Customer you are inquiring on. There is no Business Unit in the header. There is a Company. Look for an override for the AN8/CO only. There are a number of Lease programs that have a Tenant in the header and a Currency Code but do NOT have either a company or a Business Unit in the heading. Nothing to do but stick with the main default from F0301 for this program.

## Related Addresses

### Objective

To increase the number of related addresses you may associate with your address book numbers.

### Overview

JD Edwards World received requests to have more than six related addresses associated with the Address Book number. Currently, there are six related address book numbers available in the Address Book file F0101, AN81-AN86. Technically, only five of these are related addresses because AN85 was hard-coded with a specific meaning and use as the Special/Factor Payee.

With this enhancement, a new Related Address file was created, F01017, keyed by Address Number (AN8) and Related Address Code (RAC – 1 character).

Each related address is a separate record in the F01017 file. You can update these values through P01017, which is accessible by Function Key from P01051 and is located on Menu G0111.

There are three main fields that reference these related address numbers. Send Invoice To (SITO), Send Statement To (STTO), and the new Related Address (RLAB) used in Purchasing and Sales. The values previously available for use were the following:

P = Parent

1-6 = AN81 – AN86

C or blank = the customer itself

N = No Print

These values are all still available for use along with all the other numbers, letters, and characters.

In the past, if you left one of the related address fields blank when entering or changing Address Book information, the Address Book itself would populate that AN81-AN86 field. The new file does not follow that convention. There are a few rules that apply to the new Related Address entry screen and file:

- You may not enter a record where the Related Address is the same as the Address Number you are entering the record for
- You may not enter records that have the following 'hard-coded' usage:

  P, C, blank, N, 5

The XS server handles retrieving these values from other sources, returning them to the calling program so there is no need to enter them to the F01017 file.

A search window was created for Related Addresses (P01RAW). This window appears when you press F1 on the SITO or STTO input fields. Any group that allows input to the SITO, STTO, RLAB, or any other field that previously worked with related address values such as we are discussing here needs to implement this F1 override on all programs that allow them to enter values for RLAB. See program P01053 for an example of this in S00EX, search on STTO or SITO.

### XF01017

A regular I/O Server was created for the F01017 file, XF01017. If you know the key and the record you are going to work with, you may use this server. It is necessary that you use this server if you plan to Add, Change, or Delete an F01017 record; however, it is not expected to have changes or additions to occur anywhere but in P01017.

### XS01017

This is the new server created, which needs to be used when you need to retrieve a related address. This server just retrieves the related address and does not return a record image (as there are no other fields on the record you really need to see).

Input Parm:

```
PS@@
```

Output Parm:

```
PS@@
```

The data structure is defined in I01017X, called DSRAD. This data structure has the specific input parms necessary to locate the related address you are trying to retrieve.

```
IDSRAD       DS                            100
I*
I*    INPUT Parms
I*      Mode  (Future Use)
I                                       1   1 #RAMDE
I*      Address
I                                       2   90#RAAN8
I*      Related Address Code
I                                      10  10 #RARAC
I*    OUTPUT Parms
I*      Related Address
I*
I                                      11  180#RAA8R
I*
I*      Status Flag  (blank = record found, 1 = No rec fnd)
I*        AN8 is returned in A8R field if no record found.
I*
I                                      19  19 #RASTS
```

Identify your custom programs that work with Related Addresses. Pass in the Address Number and the Related Address Code. The server locates the correct Related Address and passes it back to you in #RAAN8R.

The #RASTS status flag is set to '1' if the record you are trying to locate is not found. The Address Number (#RAAN8) returns itself back in the #RAAN8R field as well and in most programs, you may be able to just use that and not do any further logic.

The following is an example of how this server can be used in your programs. The first part is a Before view of some code and the After view of that same section of code (P035001). You may be able to remove a significant portion of code from your

programs since the server chains to the F0101 file for the AN85 or the parent if requested.

Before:

```
C*
C*    Load "Remit To" address.
C*    If "Remit To Who" is blank, use the company, else use
C*    specific alternate for the address or the address given
C*    in the processing options.
C*
CSR         $REMIT    IFEQ *ON
C*
CSR         $ALT      IFNE *BLANK
CSR                   MOVE RPAN8     ABAN8
CSR                   MOVEL'ABKY01'  @@KLST
CSR                   MOVEL'CHAIN'   @@OPER
CSR                   Z-ADD1         @@KNUM
CSR                   MOVEL'N'       @@LOCK
CSR                   CALL 'XF0101'
C*                    ---- --------
CSR                   PARM           PS@@1
CSR                   PARM           I0101
CSR         @@IOR     COMP 'NF'                     81
CSR         *IN81     IFEQ *OFF
C*
C*  If overrides were located, replace AB data w/override ab
C*  data, otherwise, leave AB values as they are from F0101.
C*
CSR         #C3ERR    IFEQ *BLANKS
CSR         #C3STS    ANDEQ*BLANKS
CSR                   MOVE #C3A85    ABAN85
CSR                   MOVE #C3A01    ABAC01
CSR                   MOVE #C3A02    ABAC02
CSR                   MOVE #C3A03    ABAC03
CSR                   MOVE #C3A04    ABAC04
CSR                   MOVE #C3A05    ABAC05
CSR                   MOVE #C3A06    ABAC06
```

```
CSR                     MOVE #C3A07     ABAC07
CSR                     MOVE #C3A08     ABAC08
CSR                     MOVE #C3A09     ABAC09
CSR                     MOVE #C3A10     ABAC10
CSR                     ENDIF
C*
CSR                     SELEC
CSR           $ALT      WHEQ 'A'
CSR                     MOVE ABAN81     $RWHO
CSR           $ALT      WHEQ 'B'
CSR                     MOVE ABAN82     $RWHO
CSR           $ALT      WHEQ 'C'
CSR                     MOVE ABAN83     $RWHO
CSR           $ALT      WHEQ 'D'
CSR                     MOVE ABAN84     $RWHO
CSR           $ALT      WHEQ 'E'
CSR                     MOVE ABAN85     $RWHO
CSR           $ALT      WHEQ 'F'
CSR                     MOVE ABAN86     $RWHO
CSR                     ENDSL
CSR                     ENDIF
CSR                     ENDIF
C*
CSR           $RMWHO    IFEQ *BLANKS
CSR           $AN8CO    IFNE *ZEROS
CSR                     MOVE $AN8CO     $RWHO
CSR                     ELSE
CSR                     MOVE $CO8       $RWHO
CSR                     ENDIF
CSR                     ENDIF
C*
CSR                     MOVE $RWHO      ABAN8
CSR                     MOVEL'ABKY01'   @@KLST
CSR                     MOVEL'CHAIN'    @@OPER
CSR                     Z-ADD1          @@KNUM
```

After:

```
C*
C*    Load "Remit To" address.
C*    If "Remit To Who" is blank, use the company, else use
C*    specific alternate for the address or the address given
C*    in the processing options.
C*
CSR         $REMIT    IFEQ *ON
C*
CSR                   SELEC
C*
C*   Use company if both POs are blank/not entered.
C*
CSR         $RMWHO    WHEQ *BLANK
CSR         $RWHO     ANDEQ*ZEROS
CSR         $AN8CO    IFNE *ZEROS
CSR                   MOVE $AN8CO    $RWHO
CSR                   ELSE
CSR                   MOVE $CO8      $RWHO
CSR                   ENDIF
C*
C*   Retrieve Related Address if requested.
C*
CSR         $RMWHO    WHNE *BLANKS
CSR                   CLEARPS@@
CSR                   CLEARDSRAD
CSR                   MOVE RPAN8     #RAAN8
CSR                   MOVE $RMWHO    #RARAC
CSR                   MOVELDSRAD     PS@@
CSR                   CALL 'XS01017'              87
C*                    ---- ---------
CSR                   PARM           PS@@
CSR                   MOVELPS@@      DSRAD
CSR                   MOVE #RAA8R    $RWHO
C*
CSR                   OTHER
C*** No action.$RWHO already has the specific address from PO.
```

```
CSR                 ENDSL
C*
CSR                 MOVE $RWHO      ABAN8
CSR                 MOVEL'ABKY01'   @@KLST
CSR                 MOVEL'CHAIN'    @@OPER
CSR                 Z-ADD1          @@KNUM
```

## Multiple Vendor Bank Account

### Objective

To add more flexibility and functionality to our currency Bank Account processes.

### Overview

JD Edwards World received requests asking to be able to maintain and use more than one A/P and A/R Bank Account for payment and receipts processing (V and D Bank Types in F0030). Prior to this change, you were only permitted to have one V and/or one D account for each Address Book record in the F0030.

The functionality was expanded by allowing users to define alternate Bank Types to be used in place of V and D accounts as well as to assign a Currency Code to the Bank Type record to further separate like Bank Types (for example, you may have a 'V' for USD and a 'V' for BEF in the F0030 file for the same Address Book).

In addition, Effective Date and an Expiration Date were added to the file so you could have multiple Vs in the file for one Address Book record with all other information the same, if one of the Vs was expired and one was still effective.

This allows clients to better handle changes in their business environment but it makes it difficult for us to know just which Bank Type record to retrieve for their payments and auto debits.

The previous key to Bank Account retrieval was AN8/BKTP. Now it is AN8/BKTP/CRCD/DTEF and there is no guarantee that the key submitted will be found as we are not requiring the users to enter bank account records for ALL Bank Types and Currencies.

Identify your custom programs that work with the F0030 file.

### XF0030

A regular I/O Server was created for the F0030 file, XF0030. If you know the key and the record you are going to work with, you may use this server. It is necessary that you use this server if you plan to Add, Change, or Delete an F0030 record (with the exception of updating the Next Number in the G records, which is covered in the X0430 server).

### X0430

There is a server that works with the G type records also used in payments/auto debits. This G type record holds the Bank Account information of 'OUR' bank and this record does not have the same new fields. It is identified by Bank Type (BKTP=G) and Account Number (AID). The server that retrieves and updates the Next Payment Number for these records is X0430 and it has not changed.

### XS0030

This is the new server created, which needs to be used when you need to retrieve the correct Bank Account (previously V or D Bank Type records). This server just retrieves the record for you.

Input Parm:

```
PS@@
```

Output Parm:

```
DS0030  (the F0030 record)
```

Data structure defined as DSAY has the specific input parms necessary to locate the bank account record you are trying to retrieve.

```
D DSAY            DS
D  DSSYS                   1      2
D  DSAN8                   3     10  0
D  DSBKTP                 11     12
D  DSCRCD                 13     15
D  DSDTEF                 16     21  0
```

The DSSYS should be either '03' or '04' for A/R or A/P. This is used to determine if the default Bank Type is going to be 'D' or 'V' respectively. The remaining fields are self-explanatory. You may provide as much or as little of that information as you want and this server retrieves the bank account record that comes as close to your request as possible, ending with the basic AN8/V or AN8/D as the default that is always used. An error occurs only if there is no default V or D in the F0030 for your address.

The following is the order to look for the records within the server:

1. Search for bank type and currency they request.

2. Search for bank type and blank currency.

3. Search for default bank type (V or D) and currency.

4. Search for default bank type (V or D) and blank currency.

JD Edwards World carries out a Set Greater Than and Read Prior to take into consideration the effective date you send in.

Once you have retrieved the Bank Account record, it may be necessary to retrieve the same record later in a process. You can keep the Unique Key of the retrieved record (UKID) so you can do a direct read to that particular record later in your process. For example, in payments and auto debits, it is a multi-program process that can occur over time.

Store the value of UKID as a work variable, to guarantee the retrieval of the same record later in the process. In the event that someone enters another V record for the currency or enters another V with an effective date closer to the date, you are currently processing between the time of the first retrieve the V and when processing is finished this same information.

This server was implemented in a number of programs with the A/P Functional Server being one. The following is the code from that program, showing how this new server was used to retrieve the Bank Transit Account for a particular voucher. The DSAY data structure was defined in the Specifications displayed earlier. The reason for a SELEC statement for the BKTP assignment, is that some of the programs also allow the user to override the Bank Type in the Processing Options so you may have multiple sources of the Bank Type to use.

```
C*    Edit - Transit Number
C*      (after bank type, g/l date and currency)
C*
CSR                 MOVELRPPYE     DSAN8
CSR                 SELEC
CSR         RPBKTP  WHNE *BLANKS
CSR                 MOVELRPBKTP    DSBKTP
CSR                 OTHER
CSR                 MOVEL'V '      DSBKTP
CSR                 ENDSL
CSR                 MOVELRPCRCD    DSCRCD
CSR                 MOVELRPDGJ     DSDTEF
CSR                 MOVEL'04'      DSSYS
C*
CSR                 MOVE *BLANKS   PS@@
CSR                 MOVELDSAY      KY@@
CSR                 CALL 'XS0030'              98
C*                  ---- --------
CSR                 PARM           PS@@
```

```
CSR                 PARM           I0030
CSR         RT@@    IFNE 'N'
CSR                 MOVE AYTNST    RPTNST
CSR                 ELSE
CSR                 MOVE *BLANKS   RPTNST
CSR                 ENDIF
```

## Name Search

### Objective

For years, P01NS and P01200 were supported and maintained. These programs are virtually identical in function and purpose. To reduce redundancy, P01NS was made obsolete and replaced with the P01200 program.

Identify your custom programs that call the P01NS program and change that call to now call P01200.

The following is an example of the switch, as you will see in P00151:

Before

```
CSR                 MOVE *BLANKS   PSVAL   8
CSR                 MOVE *BLANKS   PSALPH
CSR                 MOVE '#N8 '    PSDTE   4
CSR                 CALL 'P01NS   '
C*                  ---- ------
CSR                 PARM           PSALPH
CSR                 PARM           PSVAL
CSR                 PARM           PSDTE
C*
CSR         PSVAL   IFNE *BLANK
CSR                 MOVELPSVAL     VD#N8
CSR                 MOVELPSVAL     @Q2,03
CSR                 ENDIF
```

After

```
CSR                 MOVE *BLANKS   PSVAL   8
CSR                 MOVE *BLANKS   PSALPH
```

```
CSR                   MOVE '#N8 '    PSDTE   4
CSR                   CALL 'P01200  '
C*                    ---- --------
CSR                   PARM 'P01200'  PSPID  10
CSR                   PARM 'ZJDE0001'PSVERS 10
CSR                   PARM *BLANKS   PSAT1   3
CSR                   PARM           PSALPH
CSR                   PARM           PSVAL
CSR                   PARM           PSDTE
C*
CSR         PSVAL     IFNE *BLANK
CSR                   MOVELPSVAL     VD#N8
CSR                   MOVELPSVAL     @Q2,03
CSR                   ENDIF
```

**Note:** You have to add the three parms that were part of the P01200 before the first three parms of P01NS. The PSAT1 Search Type parm was an optional fourth parm on P01NS and if used, you need to switch the position of that parm.

## General A/B, A/R, and A/P File Changes

Identify your custom programs where you Add/Change/Delete invoices (F0311) and vouchers (F0411) to load CNCD (both A/R and A/P), CRC (A/P), PA8 (A/P), BKTP (A/R and A/P), and VR01 (A/R and A/P) from the F0101/F0401/F0301 records.

Address Server copy mod I01100 used by X01100 was modified to handle new key fields as parms (add #1IDLN and #1ADTP). Recompile all programs using the X01100 server to handle additional parms, modifying a few to send in the IDLN (see P0111 for example).

For Structure Type (OSTP) equal to blanks, which is the A/R and A/P Parent/Child structure, modify custom programs that add/change the MAPA8 parent number in F0150 to also update the F0101 Parent field, ABPA8.

The key for Address Book File F0116 was changed from AN8, EFTB to AN8, IDLN, ADTP, and EFTB. Also, joins using F0116; F0101JB, JC, JD, and JE. Review all programs that access F0116 or these logicals for necessary changes to the key.

Evaluate the F0101 XAB Inactive Flag and the F01016 XAB Inactive Flag in any programs that currently evaluate the Hold Invoices, Hold Payment, and/or Subledger Inactive Code to prevent further processing if either of those Inactive Flags is '1'.

When adding new F0101, F0301, and F0401, add update to TORG - Originator.

Review all programs that do comparisons on Credit Limit and ensure that a zero credit limit means no credit allowed versus thinking of unlimited credit.

Review all programs that use Who's Who Type Code (TYC) and modify to handle the increased size (from 1 to 3).

Review all programs that use Bank Type Code (BKTP) and modify to handle the increased size (from 1 to 2).

If time stamp fields have been added to files that your program updates, be sure to update these fields before writing a record to the file.

For any alpha field that has changed size (TYC, BKTP), use a MOVEL instead of a MOVE.

If your program uses the field BKTP or TYC, because the fields changed size, so did the UDC values. Make changes within your program to reflect this UDC field size change for 00/BT (BKTP) and 01/W0 (TYC).

For any program that references data item RYIN, you need to change the UDC value it uses from 00/PY to 00/RY. This is because previously both data items RYIN and PYIN used the same UDC codes of 00/PY. The values were separated into two separate UDCs and RYIN now has a new UDC of 00/RY.

Any program that updates (add, change, delete) A/B files must implement file servers instead of doing direct updates. This is important to use Approvals logic.

F0101 – XF0101

F0301 – XF0301

F0401 – XF0401

F0111 – XF0111

F0115 – XF0115

F0116 – XF0116

In some cases, the unique key to a file has changed. To take this into account, you need to make changes to your program accordingly. Move blanks to any new key fields.

1. F0030 Previous: AN8, BKTP, CRCD, TNST, CBNK. CRCD was added to the middle of this unique key. In addition, BKTP has gone from one alpha to three alpha.

2. F0116 Previous: AN8, EFTB. IDLN and ADTP were added to this unique key after AN8 and before EFTB. This is the order: AN8, IDLN, ADTP, and EFTB.

3. F0417 Previous: PYIN, GLBA. CRCD was added to the end of this unique key.

All custom programs should at least be compiled even if you make no changes.

# Distribution and Manufacturing Systems

This section provides database and system changes for the Distribution and Manufacturing systems.

## Database Changes

This chapter lists file level database changes for the Distribution and Manufacturing systems. New Physical Files and their Dependent LFs or Copy Members

### Summary of Database Changes

Double-click the following icon to open the Manufacturing and Distribution Database Changes Summary.

### Changed Physical File Layouts

Double-click the following icon to open the Manufacturing and Distribution Changed Physical Files Detail.

### New Logical Files for Existing Physical Files

Double-click the following icon to open the Manufacturing and Distribution New Logical Files for Existing Physical Files.

### Changed Logical Files

Double-click the following icon to open the Manufacturing and Distribution Changed Logical Files.

# Copy Member Changes (non-Database)

| Rptg Code | Copy Member | Func-tion | Description | Action | Release/ Cume |
|---|---|---|---|---|---|
| 30 | C3007 | COPY | Back/Forward Schedule Operation Sequence Numbers | Changed | A73CU07 |
| 30 | C3007 | COPY | Back/Forward Schedule Operation Sequence Numbers | Changed | A73CU11 |
| 30 | C3007 | COPY | Back/Forward Schedule Operation Sequence Numbers | Changed | A73CU13 |
| 30 | C3007L | CPYL | Back/Forward Schedule Operation Sequence Numbers | New | A73CU16 |
| 30 | C3007L | CPYL | Back/Forward Schedule Operation Sequence Numbers | Changed | A91 |
| 31 | C3101 | COPY | Common Routine for AAI retrieval & Account validation | Changed | A73CU11 |
| 31 | C3101 | COPY | Common Routine for AAI retrieval & Account validation | Changed | A91 |
| 31 | C3102 | COPY | Common Routine for Inventory Transfer | New | A91 |
| 31 | C3103 | COPY | Common Routine for Components Issue | New | A91 |
| 31 | C3104 | COPY | Common Routine for Work Order Completion | New | A91 |
| 31 | C3105 | COPY | Common Routine for Work Order Hour and Quantity Posting | New | A91 |
| 31 | C3106 | COPY | Common Routine for De-commitment | New | A91 |
| 32 | C3294 | COPY | Configured Item Segment Editing | Changed | A73CU11 |
| 32 | C3294 | COPY | Configured Item Segment Editing | Changed | A73CU12 |
| 32 | C3294 | COPY | Configured Item Segment Editing | Changed | A91 |
| 34 | C341161 | COPY | File Server Key List - XF3411 | Changed | A73CU10 |
| 34 | C341161 | COPY | File Server Key List - XF3411 | Changed | A91 |
| 37 | C3701 | COPY | Return numeric value of Quality Management test results | New | A91 |

| Rptg Code | Copy Member | Func-tion | Description | Action | Release/ Cume |
|---|---|---|---|---|---|
| 37 | C371181 | COPY | File Server Key Lists - XF371181 | New | A91 |
| 40 | C0043 | COPY | Scan and Replace Character | New | A73CU11 |
| 40 | C3002 | COPY | Determine Recursive Component Items | Changed | A73CU12 |
| 40 | C3002 | COPY | Determine Recursive Component Items | Changed | A73CU16 |
| 40 | C4030071 | COPY | File Server Key List - XF40300 | Changed | A91 |
| 40 | C4102A | COPY | Common Subroutine - Calculate Available Quantity | Changed | A91 |
| 40 | C4102AL | CPYL | Common Subroutine - Calculate Available Quantity (ILE) | New | A73CU14 |
| 40 | C4102AL | CPYL | Common Subroutine - Calculate Available Quantity (ILE) | Changed | A91 |
| 40 | C73020 | COPY | /COPY Load Vertex Sales Tax Parameters | Changed | A73CU07 |
| 40 | C73020 | COPY | /COPY Load Vertex Sales Tax Parameters | Changed | A73CU11 |
| 40 | C73020 | COPY | /COPY Load Vertex Sales Tax Parameters | Changed | A91 |
| 40 | C73020L | CPYL | /COPY Load Vertex Sales Tax Parameters | New | A91 |
| 41 | C410871 | COPY | File Server Key List - XF4108 | Changed | A91 |
| 41B | C41512KL | COPY | IO Server Key List(s) - F41512 | Changed | A91 |
| 41B | C41DECL | CPYL | Determine Decimal Position for Temperature/Density | New | A91 |
| 42 | C005A | COPY | Common Subroutine for ECS functions | New | A91 |
| 42 | C005AL | CPYL | Common Subroutine for ECS functions (ILE) | New | A91 |
| 42 | C005E2 | COPY | Common Subroutine - Edit Manual Date Entry | New | A91 |
| 42 | C005E2L | CPYL | Common Subroutine - Edit Manual Date Entry (ILE) | New | A91 |

| Rptg Code | Copy Member | Func-tion | Description | Action | Release/ Cume |
|---|---|---|---|---|---|
| 42 | C005K | COPY | Common Subroutine - Currency Conversion | New | A91 |
| 42 | C005KL | CPYL | Common Subroutine - Currency Conversion (ILE) | New | A91 |
| 42 | C005Z1 | COPY | Common Subroutine - Currency Conversion | New | A91 |
| 42 | C005Z1L | CPYL | Common Subroutine - Currency Conversion (ILE) | New | A91 |
| 42 | C00VL | COPY | Common Subroutine - Retrieve Data Dictionary Defaults(P4211) | New | A91 |
| 42 | C00VLL | CPYL | Common Subroutine - Retrieve Data Dictionary Defaults(P4211) | New | A91 |
| 42 | C0101 | COPY | /COPY Retrieve Address Number | New | A91 |
| 42 | C0101L | CPYL | /COPY Retrieve Address Number (ILE) | New | A91 |
| 42 | C42119KL | COPY | IO Server Key List(s) - F42119 | Changed | A91 |
| 42 | C4211KYL | CPYL | Key List - P4211 (ILE) | New | A73CU14 |
| 42 | C4211KYL | CPYL | Key List - P4211 (ILE) | Changed | A73CU15 |
| 42 | C4211KYL | CPYL | Key List - P4211 (ILE) | Changed | A73CU16 |
| 42 | C4211KYL | CPYL | Key List - P4211 (ILE) | Changed | A91 |
| 42 | C4219 | COPY | Common Subroutine F4211 to F42199 | Changed | A73CU10 |
| 42 | C4219 | COPY | Common Subroutine F4211 to F42199 | Changed | A91 |
| 42 | C4219L | CPYL | Common Subroutine F4211 to F42199 (ILE) | New | A73CU14 |
| 42 | C4219L | CPYL | Common Subroutine F4211 to F42199 (ILE) | Changed | A91 |
| 42 | C998A | COPY | Common Subroutine - Retrieve Data Dictionary Defaults(P4211) | New | A91 |
| 42 | C998AL | CPYL | Common Subroutine - Retrieve Data Dictionary Defaults(P4211) | New | A91 |

| Rptg Code | Copy Member | Func-tion | Description | Action | Release/ Cume |
|---|---|---|---|---|---|
| 42 | CCHKDGT | COPY | UCC128 Check Digit Calculation | New | A91 |
| 43 | C4319 | COPY | CSR - Load F4311 Fields to F43199 | Changed | A91 |
| 43 | C4319L | CPYL | CSR - Load F4311 Fields to F43199 | New | A73CU14 |
| 43 | C4319L | CPYL | CSR - Load F4311 Fields to F43199 | Changed | A91 |
| 43 | C43291L | CPYL | Reverse foreign landed costs | New | A91 |
| 43 | C73022L | CPYL | /COPY Load Vertex Tax Parameters - Purchasing | New | A73CU14 |
| 43 | C73022L | CPYL | /COPY Load Vertex Tax Parameters - Purchasing | Changed | A91 |
| 43 | C73025 | COPY | /COPY Load Vertex Tax Parameters - Receipts | Changed | A73CU07 |
| 43 | C99843L | CPYL | Common Subroutine - Retrieve DD Defaults (P4312) | New | A91 |
| 43 | CWRTHISM | COPY | Write Receipt History - Match time | New | A91 |
| 43 | CWRTHISRL | CPYL | Write Receipt History - Receipt time | New | A91 |

## Server Changes

Double-click the following icon to open the Manufacturing and Distribution New and Changed Server Programs.

## Distribution Changes

This chapter describes changes made to the Distribution system.

# A8.1 Changes

## Barcode Enhancements for Selected Reports

This project adds barcode capability to selected reports. By doing so, clients are able to record their activities more accurately and more quickly.

The programs affected by this enhancement are:

- Stock Tags
- Inventory Count Sheet
- Pick Slips Print
- Purchase Receivers Print
- Print Simple Work Order
- Equipment Work Order Print
- Work Order Generation

Other programs in the Warehouse Management System, such as Movement Slips and Move Tags, are supported as well.

The major changes to these reports include new processing options for barcodes. Users have a choice of which symbology to use.

### Map of Specifications

| Program ID | Description | Batch/Interactive | Type | New/Changed |
|---|---|---|---|---|
| P31410 | Print Pick List | Batch | Print | Changed |
| P41410 | Inventory Count Sheets | Batch | Print | Changed |
| P41531 | Stock Tags | Batch | Print | Changed |
| P42520 | Print Pick Slips | Batch | Print | Changed |
| P43510 | Print Purchase Receivers | Batch | Print | Changed |
| P48415 | Print Work Orders | Batch | Print | Changed |
| P48425 | Print Pick List | Batch | Print | Changed |

| Report | Description | Type | New/Changed |
|---|---|---|---|
| R31410 | Work Order Exception Report | Print | Changed |
| R41410 | Inventory Count Sheet | Print | Changed |
| S41531 | Stock Tags | Print | Changed |
| R42520 | Pick Slips Print | Print | Changed |

| Report | Description | Type | New/Changed |
|---|---|---|---|
| R43510 | Purchase Receivers Print | Print | Changed |
| R48415 | Print Simple Work Order | Print | Changed |
| R48425 | Equipment Work Order Print | Print | Changed |

## Compile Specifications for Barcoded Printer Files (Example)

```
                        Create Printer File (CRTPRTF)

Type choices, press Enter.

File . . . . . .   R41410        Name
Library  . . . . . .    PGFOBJ73    Name, *CURLIB
Source file  . . . . . .   JDESRC        Name, *NONE
Library  . . . . . .    PGFSRC73    Name, *LIBL, *CURLIB
Source member  . . . . . .   R41410        Name, *FILE
Generation severity level  . . .   20           0-30
Flagging severity level  . . . .   0            0-30
Device:
Printer  . . . . . .   *JOB          Name, *JOB, *SYSVAL
Printer device type  . . . . . .   *IPDS         *SCS, *IPDS, *LINE...
Text 'description' . . . . . .   *SRCMBRTXT
```

```
                          Additional Parameters

 Source listing options . . . . .                 *SRC, *NOSRC, *SOURCE...
              + for more values
 Page size:
   Length--lines per page . . . . > 51             .001-255.000
   Width--positions per line  . .   198            .001-378.000
   Measurement method . . . . . .   *ROWCOL        *ROWCOL, *UOM
 Lines per inch . . . . . . . . .   6       6, 3, 4, 7.5, 7,5, 8, 9, 12
 Characters per inch  . . . . . .   15     10, 5, 12, 13.3, 13,3, 15...
 Front margin:
   Offset down  . . . . . . . . .   *DEVD          0-57.790, *DEVD
   Offset across  . . . . . . . .                  0-57.790

Back margin:
  Offset down  . . . . . . . . .   *FRONTMGN 0-57.790, *FRONTMGN, *DEVD
   Offset across  . . . . . . . .                  0-57.790
 Overflow line number . . . . . .   47             1-255
 Fold records . . . . . . . . . .   *NO            *NO, *YES
 Unprintable character action:
   Replace character  . . . . . .   *YES           *YES, *NO
   Replacement character  . . . .   ' '           40-FE, ' '
 Align page . . . . . . . . . . .   *NO            *NO, *YES
 Control character  . . . . . . .   *NONE         *NONE, *FCFC, *MACHINE
 Channel values:
  Channel  . . . . . . . . . . .   *NORMAL *NORMAL, 1, 2, 3, 4, 5, 6...
  Line number for channel:
   Line . . . . . . . . . . . . .                  1-255
              + for more values
 Fidelity . . . . . . . . . . . .   *CONTENT       *CONTENT, *ABSOLUTE
```

```
AFP Characters . . . . . . . . .   *NONE          Character value, *NONE
             + for more values
Degree of page rotation  . . . . > 90       *AUTO, *DEVD, *COR, 0, 90...
Pages per side . . . . . . . . .   1              1-4
Reduce output  . . . . . . . . .   *TEXT          *TEXT, *NONE
Print text . . . . . . . . . . .   *JOB
Hardware justification . . . . .   0              0, 50, 100
Print on both sides  . . . . . .   *NO            *NO, *YES, *TUMBLE, *FORMDF
Unit of measure  . . . . . . . .   *INCH          *INCH, *CM
Front side overlay:
  Overlay  . . . . . . . . . . .   *NONE          Name, *NONE
    Library  . . . . . . . . . .                  Name, *LIBL, *CURLIB
  Offset down  . . . . . . . . .                  0-57.790
  Offset across  . . . . . . . .                  0-57.790
```

## Lot Number Conversion for Century Compliance (SAR 1401789)

Clients who use lot-processing types 1 or 5 for generating date specific lot numbers or serial numbers, need to run this conversion. The date-generated lot number uses a year, month, day, and next number format for the lot number. This lot number is not century-specific.

### Criteria for Running Conversion

Run this job to convert existing data (year, month, day, and next number lot number) to a century-specific format. To do this, the century is added as a prefix to all existing lot number data in the system. For example, the lot number 9612310001 will be converted to 199612310001.

### Conversion Specifications

Specify the library that contains the Lot Master file (F4108) to be used to drive the data conversion through all the files that contain log number. Then, specify the libraries that contain the data you want the driving library and Lot Master file to convert. Generally, production data is stored in one library.

For this scenario, specify the production library for both the F4108 library in processing option 1 and for the first library in processing option 2.

**Note:** If you do not have any items that use lot-processing type 1 or 5, you do not need to run this conversion.

**Note:** If the client chooses to use lot-processing type 1 or 5, you may not need to run this conversion after this Cumulative Update is installed. Contact Global Support Center for more information about the conversions and the issues surrounding this conversion.

**Note:** Users must have QSECOFR authority to run this conversion job.

## Route Type Code (ROTP)

This new data element was added to provide for additional routing types within the receipt routing process. Only one route was previously available per supplier/item relationship. This type code was added to allow for multiple route types for a particular supplier/item relationship. The UCC-128 Compliance and Transfer Orders to Receipt Routing enhancements created the need for multiple route types. The two route types added for these enhancements are ASN and Transfer Order.

## BCRC – Base Company Currency Code

This field was added to the F4311, F4301, F43199, F43121, F4343, F43800, F4211, F4201, F42199, F42119, and F42019 files. It stores the currency for the base company on a purchase order or sales order. The base company is the one associated with the header branch/plant for the order.

### Affected Programs

Any program that writes to files that contain the BCRC field.

### Implementation Steps for header files (F4301, F4201)

- If currency is off, BCRC is blank
- If currency is on, the program must locate the header Branch/Plant (MCU) in the Business Unit Master File (F0006) to find the company (CO field). Then the program finds the company in the Company Constants file (F0010) to load the Company Currency field (CRCD) to BCRC

### Implementation Steps for detail files (F4311, F4211, and so on)

- If currency is off, BCRC is blank
- If currency is on, load the value from the header Base Company Currency Code field to BCRC for the detail line

## Meter Reading on Bulk Delivery Confirm (Milk Run)

Users want to be able to specify the meter readings on a Milk Run trip when delivered (Bulk Delivery Confirmation for Milk Run). Currently, the Bulk Delivery Confirmation program (P49711) has the meter reading fields in the fold area, but they are just informational. The quantity delivered will be calculated based on these meter readings. In addition, should the ending meter reading be less than the beginning reading, it is assumed that the meter rolled over. This logic was included into the calculation of quantity delivered.

## Order-Centric Delivery Confirmation

Previously, the Bulk Delivery Confirmation program was a compartment-oriented program; however, in many cases, the delivery confirmation process is order based. When a bulk order is shipped in multiple vehicle compartments, you can now confirm order delivery without having to specify which compartments were unloaded. Bulk Delivery Confirmation relieves quantities from the first compartment in which the order was loaded and proceeds sequentially through the compartments until the full quantity delivered is recorded. Because this may not reflect the actual vehicle compartments relieved, you have the ability to make necessary corrections during Bulk Disposition. A new compartment allocation window enables you to rapidly list the compartments for all quantities left on board. System-created adjustment records update both the Load and Delivery Item Balance and Transaction files.

Map of Specifications

| Program ID | Description | Batch/Interactive | New/Changed |
|---|---|---|---|
| P49710W | Bulk Delivery Confirmation Window | Interactive | New |
| P49710 | Bulk Delivery Confirm | Interactive | Changed |
| P4911X | Trip Detail Conversion | Batch | New |
| P49110X | Annual Trip Detail Conversion | Batch | New |
| P49510 | Bulk Load Confirmation | Interactive | Changed |
| P49510W | Bulk Load Confirmation Window | Interactive | Changed |
| P49572 | Upload Gantry Data | Interactive | Changed |
| P49731 | Match Confirmation Batch Process | Batch | Changed |
| P49911 | Purge Trip Files | Batch | Changed |
| XT49799 | Load and Delivery Transaction Server | Interactive | Changed |
| P415403 | Throughput Reconciliation Report | Batch | Changed |
| P49620 | Bulk Invoice | Batch | Changed |
| P49730 | Mass Delivery Confirmation | Interactive | Changed |

## Batch Bulk Load Confirm

This enhancement lets the users run the load-confirm process in batch mode. The program is driven by DREAM Writer. It lets the user enter the trips or orders to be load-confirmed.

This new program performs the same processes that the interactive Bulk Load Confirm does. The program is driven by DREAM Writer, and it can be run in proof or final mode. The proof version generates an error report and performs no file updates. The final version generates an error report, and performs file updates for those trips/orders that passed all validations and edits.

The DREAM Writer allows the flexibility of choosing the trips or the sales orders to load-confirm. Whether you are load-confirming trips or orders based on a processing option. Delivery confirmation is done to orders only. The trips have to be delivery-confirmed through the interactive program.

Because there is no area to input 'quantity loaded', it is assumed that the quantity shipped is the same as the quantity loaded. In addition, all defaults are used for the values of temperature, density, density type, and so forth. If the user wants to load-confirm an order/trip with values other than the defaults, each order/trip have to be load-confirmed individually through the current interactive version of the program.

## Journal Entry Generation with Receipt Routing (SAR 1326483)

Journal entry processing in receipt routing was enhanced to allow journal entries at any step within a route. The enhancement also provides the ability to designate any step as the operation at which the order is eligible for payment. With this added functionality, it is possible to eliminate the extensive time lag that potentially could occur between liability transfer and representation on the general ledger. This allows for more timely order payments, and makes it possible to accurately recognize the value of inventory in receipt routes.

This enhancement required few database changes. Journal entry fields for the general ledger were added to the Receipt Routing Ledger file (F43199) to adequately record information for journal entries created at a particular point of item processing along the receipt route.

The added fields include:

| Field Name | Description | Size |
|---|---|---|
| PPKCO | Document Company | 5 |
| PPDOC | Document Number | 8 |
| PPDCT | Document Type | 2 |
| PPDGL | G/L Date | 6 |

## Multiple General Ledger Distribution on a Purchase Order Line

The Purchase Order System was enhanced (SAR #631048) to allow multiple accounts in the general ledger to be attached to a single purchase order line. The enhancement also supports the multiple accounts through the distribution process (order, commitment, receipt, and voucher processing). Changes include adding two files, and adding a new data dictionary item (MACT) to existing files.

### Purchase Order Multiple Account File (F4316)

This file supports individual multiple account entries, and determines how the original purchase order line is distributed.

### Model Purchase Order Multiple Account Model File (F4316M)

This file allows the definition of pre-defined multiple account entries.

### Multiple Accounts Flag (MACT)

This new data dictionary item should allow only a value of 1' if multiple account records in F4316 are attached to the purchase order line.

# A9.1 Distribution Changes

## Advanced Lot Management

Additional Lot Expiration Dates and a Lot Effective Date

The Lot Master File (F4108) has seven new lot expiration dates and a lot effective date in addition to an existing lot expiration date. These dates are calculated based on shelf life days or effective days, which are stored in the Item Master (F4101) and the Item/Branch Master (F4102) when a lot master record is created at inventory transactions. These dates are used to determine if a lot is good at inventory commitment.

Additional lot expiration dates:

- Best Before Date
- Sell By Date
- User Defined Date 1
- User Defined Date 2
- User Defined Date 3
- User Defined Date 4
- User Defined Date 5

**Note:** Each item can have information on which date is used as an expiration date. For instance, Item-A uses best before date as an expiration date while Item-B uses a lot expiration date.

The dates must be smaller or equal to a lot expiration date.

A lot is effective from an effective date through an expiration date, which is one of eight expiration dates.

A lot effective date must be smaller or equal to any lot expiration dates.

When an effective date is zero, the lot is considered effective unless it is expired.

When an expiration date is zero, the lot is considered to be expired.

The lot date calculations are done by X41LOTC. For information on X41LOTC, see the online program help of X41LOTC.

### New Fields

### Item Master (F4101)

| | | | |
|---|---|---|---|
| IMCMDM | Commitment Date Method | A(1) | Associated with UDC(40/DM) |

| | | | |
|---|---|---|---|
| IMBBDD | Best before default days | P(6,0) | Used to calculate best before date |
| IMSBDD | Sell by default days | P(6,0) | Used to calculate sell by date |
| IMU1DD | User date 1 default days | P(6,0) | Used to calculate user-defined date 1 |
| IMU2DD | User date 2 default days | P(6,0) | Used to calculate user-defined date 2 |
| IMU3DD | User date 3 default days | P(6,0) | Used to calculate user-defined date 3 |
| IMU4DD | User date 4 default days | P(6,0) | Used to calculate user-defined date 4 |
| IMU5DD | User date 5 default days | P(6,0) | Used to calculate user-defined date 5 |
| IMLEDD | Effective default days – Manufacturing | P(6,0) | Used to calculate a lot effective date in manufacturing transactions |
| IMPEFD | Effective default days – Distribution | P(6,0) | Used to calculate a lot effective date in distribution transactions |

## Item/Branch Master (F4102)

| | | | |
|---|---|---|---|
| IBCMDM | Commitment Date Method | A(1) | Associated with UDC(40/DM) |
| IBBBDD | Best before default days | P(6,0) | Used to calculate best before date |
| IBSBDD | Sell by default days | P(6,0) | Used to calculate sell by date |
| IBU1DD | User date 1 default days | P(6,0) | Used to calculate user-defined date 1 |
| IBU2DD | User date 2 default days | P(6,0) | Used to calculate user-defined date 2 |
| IBU3DD | User date 3 default days | P(6,0) | Used to calculate user-defined date 3 |
| IBU4DD | User date 4 default days | P(6,0) | Used to calculate user-defined date 4 |
| IBU5DD | User date 5 default days | P(6,0) | Used to calculate user-defined date 5 |
| IBLEDD | Effective default days – Manufacturing | P(6,0) | Used to calculate a lot effective date in manufacturing transactions |
| IBPEFD | Effective default days – Distribution | P(6,0) | Used to calculate a lot effective date in distribution transactions |

## Lot Master (F4108)

| | | | |
|---|---|---|---|
| IOOHDJ | On hand date | S(6,0) | A lot has inventory for the first time on this date |
| IOBODJ | Based on date | S(6,0) | A lot starts expiring on this date |
| IOBBDJ | Best before date | S(6,0) | |

| | | |
|---|---|---|
| IOSBDJ | Sell by date | S(6,0) |
| IOU1DJ | User defined lot date 1 | S(6,0) |
| IOU2DJ | User defined lot date 2 | S(6,0) |
| IOU3DJ | User defined lot date 3 | S(6,0) |
| IOU4DJ | User defined lot date 4 | S(6,0) |
| IOU5DJ | User defined lot date 5 | S(6,0) |
| IOETDJ | Lot effective date | S(6,0) |

### Commitment Date Method in Sales Order Processing

Commitment Date Method (CMDM) has also been added to the F4211 (Sales Order Detail) file. The sales order entry programs (such as P4211) populate the field from the default value in the F4102 (Item Branch) at the time the sales order is created.

To allow a more flexible default of the Commitment Date Method for sales lines, the Override Commitment Date Method (OCDM) was added to the new Advanced Lot Management Preference (F40319). This is a user-defined code (UDC) stored in 40/OM that mirrors the codes of the Commitment Date Method UDC table 40/DM. The only difference is that the value of blank in 40/OM designates that the default Commitment Date Method is used.

The sales order entry programs resolve this preference at the time the order is created. If a preferred override commitment date method for the sales line is found, this is stored in SDCMDM in place of the default Commitment Date Method from the F4102. The code in SDCMDM is used by any Sales Order Processing program calculating availability or committing inventory for the sales line. For instance, P42997 (Inventory Commitment/Availability) uses SDCMDM to determine which lot date it looks at when deciding whether or not to commit inventory from that lot.

### Changes to C4102A – Common Subroutine – Calculate Available Quantity

#AMMEJ (Lot Expiration Date) is the work field that determines whether or not the lot is included in the availability calculation. It is the responsibility of the program executing C4102A to load #AMMEJ with one of the dates (Expiration Date, Best By Date, Sell By Date, and so on.) from the Lot Master. Sales Order Processing programs populate #AMMEJ based on the code in SDCMDM (e.g. if SDCMDM is set to '2', the Sell By Date is moved into #AMMEJ). As in earlier releases, #AMMEJ is ignored if #A4108 is set to blanks.

Two new work fields were added to allow lots not yet available because of the effective date to be excluded from the availability calculation:

- #AETDJ (Lot Effective Date)
- #ADAT2 (Pick Date)

#AETDJ should be populated from the Lot Effective Date (IOETDJ) in the Lot Master (F4108) record.

#ADAT2 should be populated with whatever date is to determine if the lot is in effect yet. For Sales Order Processing programs, it is SDPDDJ (Pick Date) from the F4211.

If the date in #AETDJ is greater than the date in #ADAT2, #ASEL is set to blanks. This causes the lot to be skipped in the availability calculation. If #ADAT2 is zero, then no check of the effective date in #AETDJ is performed and the lot is treated as though it were effective. Both work fields should be populated with the date in Julian format.

### Ascending Ship Date

The F4239 (Last Customer Shipment) was added to allow the tracking of lot date information from the last shipment of an item to a customer. Both the lot number and the dates from the actual last shipment and the lot number and dates from the shipment with the latest (highest) dates are stored in a single record for each item and customer.

The following fields in the file store the actual last shipment information:

- SMLLNO (Last Lot Number)
- SMLXDJ (Last Expiration Date)
- SMSLDJ (Last Sell By Date)
- SMLBDJ (Last Best Before Date)
- SMUSD6 (User Date 6)
- SMUSD7 (User Date 7)
- SMUSD8 (User Date 8)
- SMUSD9 (User Date 9)
- SMUSD0 (Last User Defined Date)

The following fields store the latest (highest) shipment information:

- SMLOTN (Lot/SN)
- SMMMEJ (Lot Expiration Date)
- SMSBDJ (Sell By Date)
- SMBBDJ (Best Before Date)
- SMUSD1 (User Date 1)
- SMUSD2 (User Date 2)
- SMUSD3 (User Date 3)
- SMUSD4 (User Date 4)
- SMUSD5 (User Date 5)

The F4239LA is keyed by SMSHAN (Ship To) and SMITM (Short Item Number). The logical is not defined as unique, but the uniqueness of one record per ship-to/item is enforced by P4205 (Order Confirmation). Branch/plant (SMMCU) is stored in the

file but the field is not a key to the record, so shipments are not tracked by branch/plant.

P4205 is the only application that writes or updates records to the file. P4205 does so only if CPSADR (Ship Ascending Constant) in the F4009 (Distribution/Manufacturing Constants) is set to '1'. The XF4239 (Input/Output Server – F4239) provides input/output to the file.

The X4239 (Validate Last Customer Shipment) compares the dates on a given lot to the dates on the F4239 record and indicate whether or not the dates from the lot are less than (before) the dates on the F4239 record. This allows the calling program to issue a warning or error that the lot currently being shipped from is older than the latest lot shipped to that customer.

The calling program passes a F4108 (Lot Master) record along with the following data structure:

| | | | |
|---|---|---|---|
| P1SHAN | 8 | Ship To | Short address number in alphanumeric. |
| P1OCDM | 1 | Override Commitment Date Method | Not used by X4239 |
| P1EDCK | 1 | Check Expiration Date | '1' turns on comparison of IOMMEJ to SMMMEJ. |
| P1SBCK | 1 | Check Sell By Date | '1' turns on comparison of IOSBDJ to SMSBDJ. |
| P1BBCK | 1 | Check Best Before Date | '1' turns on comparison of IOBBDJ to SMBBDJ. |
| P1USC1 | 1 | Check User Defined Date 1 | '1' turns on comparison of IOU1DJ to SMUSD1. |
| P1USC2 | 1 | Check User Defined Date 2 | '1' turns on comparison of IOU2DJ to SMUSD2. |
| P1USC3 | 1 | Check User Defined Date 3 | '1' turns on comparison of IOU3DJ to SMUSD3. |
| P1USC4 | 1 | Check User Defined Date 4 | '1' turns on comparison of IOU4DJ to SMUSD4. |
| P1USC5 | 1 | Check User Defined Date 5 | '1' turns on comparison of IOU5DJ to SMUSD5. |
| P1RCOD | 3 | Return Code | A return value of 'NSF' indicates that at least one date from the F4108 that was checked is less (earlier) than the date from the F4239. |

The server takes the address number from the data structure and the item number from the F4108 record and retrieves the F4239 record. The check date flags indicate which dates to check. The comparison is done against the latest (highest) date, not the date from the actual last shipment. If any date from the F4108 that is checked is less (earlier) than the corresponding date from the F4239, the return code is set to 'NSF'.

For example, if the flag is set to check Sell By Date and the sell by date on the lot is earlier than the latest sell by date shipped to that customer, the return code is set to

'NSF'. If all lot dates checked are the same as or later than the dates from the F4239 record, then the return code is set to blanks. If the server is called but none of the check date flags are set to '1', the return code is set to blanks. If no F4239 record is found, the return code is set to 'NR '.

The check date flags originate in the Advanced Lot Management Preference (F40319). The sales order entry programs resolve this preference when the order is created and store the check date flags in new fields added to the F4211 record. If no preference is found, the fields default to blank.

## Inventory Summarization

This enhancement put in another level of summarization of the Inventory Transactions (F4111) between it and the As Of File (F41112), creating the new Inventory Summary File (F41118). If the As Of generation is run as well as this one, the Inventory Transactions can be purged. If the As Of file needs to be regenerated, this system provides that capability. Reports are also provided to insure the integrity of all these files.

When the Inventory Summarization (P41547) is run, the Inventory Summary File (F41118) is created and the Inventory Transaction file (F4111) is updated as follows:

| | | | |
|---|---|---|---|
| ILSUMP | Posted to Summary | 1 | Hardcoded to '1' |
| ILTPRG | To Be Purged | 1 | Hardcoded to '1', based on the processing option |

The Inventory Summary file is the summary of Inventory Transactions based on century, fiscal year, document type, item, branch, location, lot serial, and G/L category code.

## Service Warranty Management

This enhancement allows service warranty information to be stored and attached to lines on sales orders. If the item needs to be returned by the customer for repair, replacement, or credit, a return order is created to manage the service warranty transaction.

The warranty information is stored in the F42401 (Service Warranty Header File) and the F42402 (Service Warranty Detail File). The F42401LA is keyed by the following fields:

| | | | |
|---|---|---|---|
| CHDOCO | Order Number | 8.0 (Signed) | |
| CHDCTO | Order Type | 2 | |
| CHKCOO | Order Company | 5 | Hard-coded by application to '00000'. |
| CHCOCH | Contract Change Number | 3 | Hard-coded by application to '000' |

The F42402LA is keyed by the following fields:

| | | | |
|---|---|---|---|
| CDDOCO | Order Number | 8.0 (Signed) | |
| CDDCTO | Order Type | 2 | |
| CDKCOO | Order Company | 5 | Hard-coded by application to '00000'. |
| CDCOCH | Contract Change Number | 3 | Hard-coded by application to '000' |
| CDLNID | Line Number | 6.0 (Packed) | |

A warranty can be selected and assigned to a specific line on a sales order either interactively (P42404W – Assign Service Warranty) or in batch (P42404 – Batch Assign Service Warranty). Once the warranty is assigned, the tie to the F42401 record is stored in the F4211 (Sales Order Detail) record:

| | | | |
|---|---|---|---|
| SDWORN | Warranty Order Number | 6.0 (Signed) | Updated from CDDOCO. |
| SDWCTO | Warranty Document Type | 2 | Updated from CDDCTO. |
| SDWKCO | Warranty Document Co. | 5 | Updated from CDKCOO. |
| SDWGNO | Warranty Line Number | 6.0 (Packed) | Updated from CDCOCH. |
| SDWCEJ | Warranty Expiration Date | 6.0 (Signed) | |
| SDWFLG | Warranty Action Flag | 1 | Updated to 'W' once Install Base Record was generated. |

After the sales order is invoiced, the P42403 (Write Install Base Records) generates a F42400 (Service Warranty Install Base File) record. One F42400 record is written for each F4211 record. The only exception to this is if basic serial number processing is being used (F4220 – Serial Number File) and there are multiple serial numbers assigned to the same sales line. In this case, a F42400 record is written for each serial number.

The F42400 stores the combination of sales detail and address information that manages any warranty transaction with the ultimate purchaser of the item under warranty. This file includes a complete record image of the F4211 plus address and other fields used for any future warranty return.

Some pertinent fields:

| | | | |
|---|---|---|---|
| WIKCOO | Order Company | 5 | Order Company from the original F4211 record. |
| WIDOCO | Order Number | 8.0 (Signed) | Order Number from the original F4211 record. |
| WIDCTO | Order Type | 2 | Order Type from the original F4211 record. |
| WILNID | Line Number | 6.0 (Packed) | Line Number from the original F4211 record. |

| | | | |
|---|---|---|---|
| WIOKCO | Original Order Company | 5 | Once a warranty return order is created, this is the Order Company from the return order. |
| WIOORN | Original Order Number | 8 | Once a warranty return order is created, this is the Order Number from the return order. |
| WIOCTO | Original Order Type | 2 | Once a warranty return order is created, this is the Order Type from the return order. |
| WIOGNO | Original Line Number | 7.0 (Packed) | Once a warranty return order is created, this is the Line Number from the return line on the order. |
| WIWORN | Warranty Order Number | 8.0 (Signed) | The number of the warranty (F42402) assigned to the original F4211 line. |
| WIWCTO | Warranty Document Type | 2 | The document type of the warranty (F42402) assigned to the original F4211 line. |
| WIWKCO | Warranty Document Company | 5 | The document company of the warranty (F42402) assigned to the original F4211 line (will always be '00000'). |
| WIWGNO | Warranty Line Number | 6.0 (Signed) | The line number of the warranty (F42402) assigned to the original F4211 line. |
| WIWSNB | Warranty Serial Number | 30 | The serial number or lot number for the item under warranty. If basic serial number processing (F4220) is being used, the serial number from the F4220 record will be stored here. |
| WIAREQ | Requested By | 8.0 (Signed) | Updated with the address book number of the person requesting the return when a warranty return is created. |
| WIMLNM | Mailing Name | 40 | Address fields for the holder of the warranty. Address info can be stored here but not added to the F0101 until the holder requests a return. |
| WIADD1 | Address Line 1 | 40 | |
| WIADD2 | Address Line 2 | 40 | |
| Etc. | | | |
| WIRETL | Return Line Number | 6.0 (Packed) | The line number of the return line once a return order is created. |
| WIRFRL | Repair Line Number | 6.0 (Packed) | The line number of the repair line from the return order. |
| WIRPLL | Replacement Line Number | 6.0 (Packed) | The line number of the replacement line from the return order. |

| | | | |
|---|---|---|---|
| WILN1L | Loaner Send Line Number | 6.0 (Packed) | The line number from the return order to send out a loaner. |
| WILN2L | Loaner Return Line Number | 6.0 (Packed) | The line number from the return order to return a loaner. |
| WIRFNB | Refund Line Number | 6.0 (Packed) | The line number of the refund line from the return order. |

If the item is returned, the F42400 record generates the warranty return order. The return order is created as a sales order, with a record in the F4210 (Sales Order Header) and record(s) in the F4211. If multiple items are being returned, a separate return order is generated for each item. A return line must be created. The other lines (repair, replacement, and so on.) depend upon the disposition of the return.

The original order fields (OKCO, OORN, OCTO, OGNO) on both the F42400 record and the new F4211 return record provide a tie between the two records. The order keys of the return order are stored in the original order fields on the F42400 record. In addition, the line number for each type of line added to the return order is also stored in the F42400 record (e.g. WIRFRL is updated with the line number of the repair line).

Likewise, the order keys from the F42400 record (WIDOCO, WIDCTO, WIKCOO, and WILNID) are stored in the original order fields on the F4211 return record. In addition, the reason (return, repair, replacement, and so on.) for the creation of the return line is stored in SDSO15. The number stored in the field correlates to the option exit taken from the P42402 (Service Warranty Workbench). The value in this field can be checked if processing specific to a type of return order line is required.

The possible values for SDSO15:

| | |
|---|---|
| 3 | Return line |
| 4 | Warranty repair line |
| 5 | Non-Warranty repair line |
| 6 | Replacement line |
| 7 | Loaner send line |
| 8 | Loaner repair line |
| 9 | Refund line |

# Manufacturing Changes

## Manufacturing Accounting

### Work Order Over/Under Completion

This enhancement supplies an alternative method for calculating production variances based on the completed plus scrapped parent quantities. This eliminates variances caused by over/under completion.

Modifications were made to minimize interruptions to existing client implementations. The enhancement let each customer choose either to maintain the current method of calculating other variances or to apply the new method.

The solution includes:

- Restating standard, current, and planned production costs based on the reported quantities completed and scrapped instead of the original work order quantity
- The option to allow customers to continue using the current calculation of other variances
- Adding a single field (FORQ) to file F3102 to indicate whether component cost is fixed or variable
- Incorporation into the existing Manufacturing Variance program (P31804)

### Work-In-Process (WIP) Revaluation

WIP Revaluation provides an automated method for recording cost-change effects on WIP and generating the corresponding journal entries. Modifications were made to minimize disruptions to existing client implementations. The enhancement allows customers choose either to maintain the current method of updating frozen costs or to adjust WIP balances automatically.

The solution includes the following functionality:

- Ability to revalue WIP when a cost update is performed by revaluing active work orders for items that have changed costs, including adjusting inventory values and generating journal entries
- Allowing customers to maintain current procedures (optional based on a processing option)
- Modification of file F3102 to include the component and production order branch (MMCU). This allows inter-branch costs to be handled appropriately
- A new program (P30837) and work file (T03835) were created for revaluation
- A new join file (F3102J) selects active production orders (F4801 WAPPFG = '1') that use items with new costs (T30835) for which work in process exists (F3102)
- Minor changes to the cost update program (P30835) were needed to create the work file (T30835) to be used by the new WIP Revaluation program

- The work order routing file (F3112) was changed to include current hours by operation sequence. This file is the basis for re-calculating labor, machine, and overhead costs by using the new work center rates. This file is not converted because there is no certainty that the routing values match those at the time the work order was created. Therefore, these fields are zero for existing work orders. This makes the re-calculation of current Bx and Cx costs impossible. JD Edwards World recommends that open work orders be run through accounting variances and closed before the new release is implemented.

### Repetitive Manufacturing

Several changes due to the Repetitive Manufacturing project also affect work order cost accounting:

- The Manufacturing Variance program (P31842) was obsolete in the A9.1 release. Accounting for repetitive production is handled by the work order programs P31802 and P31804 with minor adjustments
- Program P31804 generated cumulative variances. Therefore, users were advised to run it once only. The variance flag in the work order header was used for selection purposes. Before this enhancement, only work orders with a variance flag of 1' were selected. These are the ones for which journal entries have already been created in final mode, and for which variances have not yet been calculated. After running this program in final mode, the variance flag was set to '2'. It is now possible to generate variances more than once for rates and work orders. For this reason, the program was changed to store the previous variance calculated in the F3102 file. If the program is run more than once, it calculates only the net change instead of cumulative amounts. P31804 changes the value of the flag (WAPPFG) from '2' to '3' when the production document status is greater than or equal to the status code in the SRS2 field in F3009.
- Program P4801AX was changed to update the variances flag from a value of '2' to '3' for existing work orders. This new value indicates that the work order is closed and cannot have any move activity (new transactions). For related changes, see Repetitive Manufacturing later in this chapter.

## Commitments Processing

This enhancement involved changing the Manufacturing applications to use the same basic concepts of Distribution commitment processing. Changes include the following:

- Allow soft commits to be processed against primary locations only
- Force a hard-commit when a location other than the primary is introduced to a parts list line, whether the input is manual or programmatic
- Change the order of the controlling factors in determining hard- or soft-commits (first, the existence of a non-primary location on the parts list line, and second, the commitment control and hard-commit/soft-commit fields in Manufacturing Constants)

- Accomplish line level commitments through a field in the Parts List file. If a hard-commit is performed on a parts list line item, this field sets to 'H'. If a soft-commit is performed, the field sets to 'S'.

### Nature of Change

The solution for the problem described incorporates the Manufacturing current commitment processing by branch/plant with the Distribution concept of line level commitment processing. The manufacturing software processes as before, except in the following situations:

- A client manually inputs a non-primary location into a parts list. The line is hard-committed regardless of the hard-commit/soft-commit code in Manufacturing Constants
- A non-primary location is input programmatically into a Parts List. Locations other than the primary are entered into part lists when interfacing with the Warehouse Management system, when using work center locations, and when using the Multi-Location Selection window. In these cases, the line again is hard-committed
- The line level commitment field in the Parts List file is added or changed whenever commitments are processed

### Scope

The following programs were changed to initially look at the contents of the field WMCOMM when either changing or relieving a commitment. If WMCOMM is not filled, the programs continue to use Manufacturing Constants as they do currently. Because the current logic for processing commitments was left in place, no conversion programs are necessary.

| Program | Program Description | Status |
|---|---|---|
| P31042 | Component Item Substitution | Changed |
| P3111 | Work Order Parts List Revisions | Changed |
| P3111P | Work Order Process Resource Revisions | Changed |
| P311 IS | Substitute Availability Window | Changed |
| P31113 | Work Order Inventory Issues | Changed |
| P31115 | Co/By Product Completion Window | Changed |
| P3H23 | Super Backflush | Changed |
| P31410 | Work Order Generation | Changed |
| P31415 | Work Order Print Parts List | Changed |
| P31418 | Work Order Parts Shortage Report | Changed |
| P31420 | Work Order Automatic Batch Issue | Changed |

| Program | Program Description | Status |
|---|---|---|
| P3190 | Work Order Repost Commitments | Changed |
| P3197 | Batch Work Order Commitments | Changed |
| P4600 | Request Inquiry | Changed |
| P4617 | Online Confirmation | Changed |
| P4617W | Confirmation Overrides | Changed |
| P48013 | Manufacturing Work Order Entry | Changed |
| X3294 | Sales Order Detail Server | Changed |
| X4617 | Confirmation Server | Changed |

## Quality Management

Quality Management is a new system (37) designed to help companies capture and manage data related to the quality of their products. The emphasis of Quality Management is on material quality; specifically, whether material conforms to specifications. The module allows users manage and report the inspection and testing of material at the points of receiving, manufacturing, and shipping.

### Objectives

- Provide the ability to create and maintain tests. Test attributes should include allowed values, target measurements, and instructions related to a particular test
- Develop a method of grouping a test into specifications. In addition, create means of linking tests and specifications to an item at a particular point in the production process
- Support the use of preference profiles so that item/test specifications can be customized for a particular customer or customer group
- Create a program that allows the entry of test results at various stages of purchasing, sales, and the work order life cycle. This includes receipts, completions, any operations within the receipt-routing process, and ship and load confirmation
- Allow the input of one-time tests during results entry and, conversely, be able to delete predefined tests for an item after the result display is loaded with item-specific tests
- Display test results through reports and inquiry screens. Clients are able to review the results by lot or by work order
- Allow the ability to evaluate a lot for a particular customer at sales order entry
- Generate a Certificate of Analysis (COA) for customers who require documented verification of product quality

- Develop a lot-tracing capability for finding test results assigned to the components of an assembled good or an item that was re-classed. Support this functionality through both inquiry screens and the Certificate of Analysis
- Provide for the ability to track non-conforming products
- Allow the entry of generic text during test revisions, specification revisions, item/test specifications, and results entry

## Scope

Quality Management required the following new programs as well as modifications to existing programs:

| Program | Program Description | Status |
|---|---|---|
| P3112 | Work Order Routing Instructions Revisions | Changed |
| P31225 | Manufacturing Scheduling Workbench | Changed |
| P31114 | Manufacturing Completion | Changed |
| P31115 | Co/By Product Completion | Changed |
| P311221 | Work Order Employee Time Entry | Changed |
| P3H23 | Super Backflush | Changed |
| P3119 | Completions Workbench | Changed |
| P3701 | Test Revisions | New |
| P3701W | Test/Specification Search Window | New |
| P3702 | Specification Revisions | New |
| P3702W | Specification Revisions Selection Inquiry | New |
| P3703 | Non-Conforming Product | New |
| P3711 | Test Results Revisions | New |
| P37111 | Test Results Edit | New |
| P37111W | Test Status Revisions | New |
| P37112W | Lot Search Window | New |
| P37113W | Test Results Inquiry | New |
| P37114W | Test Results Selection Window | New |
| P37200 | Tested Lot Search | New |
| P37201 | Trace Test Results | New |
| P37202 | Test/Specification Where Used | New |

| Program | Program Description | Status |
|---|---|---|
| P37203 | Test Result Workbench | New |
| P37204 | Test Results Inquiry | New |
| P37205 | Test Results Report | New |
| P37410 | Test Definitions Report | New |
| P37415 | Specifications Report | New |
| P37420 | Item/Test/Operations Report | New |
| P37450 | Product Test Report | New |
| P37460 | Certificate of Analysis | New |
| P37800 | Batch Test Results Entry | New |
| P37900 | Certificate of Analysis Extract | New |
| P40ITM2 | Item Search/Return | Changed |
| P40SEL | Selection Criteria Window | Changed |
| P4007 | Preference Profiles | Changed |
| P40300 | Preference Revisions | Changed |
| P40400 | Preference Processing | Changed |
| P41001 | Branch Plant Constants | Changed |
| P4108 | Lot Master Revisions | Changed |
| P4113 | Inventory Transfers | Changed |
| P41200 | Item Search with Word Search | Changed |
| P41280 | Lot Availability | Changed |
| P4205 | Ship Confirm | Changed |
| P4211 | Sales Order Entry Detail | Changed |
| P4242 | Transfer Order Entry | Changed |
| P4312 | Receipts by PO/Item/Account | Changed |
| P43250 | Routing Movement/Status | Changed |
| P48013 | Manufacturing Work Order Entry | Changed |
| P49510 | Bulk Confirm | Changed |
| XF3711 | Input/Output Server - F3711 | New |
| XF40300 | Input/Output Server - Preference Profiles | Changed |

| Program | Program Description | Status |
|---|---|---|
| XF40303 | Input/Output Server - Preference Profiles | New |

### Important Note for ECS Clients

Clients upgrading from either A7.1 or A7.3 to A9.1 release and are using the Quality Preference available through Load and Delivery Management, have to run a conversion program. The conversion program reads data from Product Specifications (F4988) and writes data to the Quality Management Preference file (F40318) and Test Definitions (F3701). The historical data that resides in Vehicle Lab Results are not converted. A7.1 and A7.3 Load and Delivery Management programs and files related to the ECS Quality Preference were obsoleted.

## Forecasting Enhancements

As part of the analysis in converting the existing Forecasting functionality to EnterpriseOne, several areas for improvement in the JD Edwards World product were identified. Many of the concepts in question were consistent with client suggestions for improvements. Major areas identified included enhancing the forecast calculations, providing a new method of determining best fit, and creating a link between summary and detail records.

### Objectives

- Enhance the forecast calculations according to standard statistical formulas, and allow users to input parameters to include in the calculations
- Provide an option to use MAD as an alternate method of determining best fit
- Create detail forecast records when calculating summary forecasts
- Write zero and negative quantities to the forecasting files and change MRP logic accordingly
- Allow the forecast generation program to run in proof or final mode Provide a more accurate method of generating weekly forecasts
- Allow the Extract Sales Order History program to extract large customer forecast records by Ship To or Sold To address book number
- Link the amounts and quantities in both detail and summary forecasts so that a change in one would be reflected in the other by the original ratio between the two
- For large customer processing, allow best fit determination and forecast calculation by each customer
- Allow forecasts to be generated for more than one year
- Provide a method of creating different versions of a forecast for comparison

### Description of Changes

In the past, the three most recent periods were used in calculating forecasts. The calculations did not allow users to specify the number of periods to use in the calculations. To provide this flexibility, users now are allowed to enter the number of periods to be used in the forecast calculation for several of the existing methods. This also includes allowing entry of alpha and beta factors in Exponential Smoothing calculations.

Two new methods for forecasting were introduced. The first, Least Squares Regression, is similar to Linear Approximation, but utilizes a widely accepted statistical formula. The second, Exponential Smoothing with Trend, provides the ability to perform exponential smoothing when seasonality is not present.

Previously, the only basis for determining best fit was a method referred to as Percent of Accuracy (POA). This approach is not regarded as a standard method for calculating the accuracy of a forecast. One method, known as Mean Absolute Deviation (MAD) is based on a proven statistical formula for forecasting data points. MAD provides a much more consistent method for comparing alternate forecasting methods. To keep the existing functionality intact for current clients, a new processing option was added to let the user choose between POA or MAD for best fit analysis.

Before, when Summary Forecast Generation was run, detail forecast records were not created. This left no connection between the summary and detail records. The Summary Forecast Generation program was modified to create detail forecast records.

The previous forecasting process did not allow zero or negative quantities to be written to the Detail Forecast and Summary Forecast files. This caused confusion for clients when fewer records were created than the number specified. Writing a record for each period now provides a complete look at the resulting calculations.

The previous forecast generation did not give users a chance to analyze the forecast before updating the files. Both the Detail Forecast Generation and the Summary Forecast Generation programs now can be run several times in 'proof mode until acceptable results are obtained. The program then can be run in final mode to generate forecast records.

The way weekly forecasting had been calculated was too simplistic. Weekly forecasts were simply the result of dividing the monthly forecast by the number of weeks in the month. This did not truly represent weekly forecasting and tended to provide a very flat distribution of the forecast across the month.

To provide the ability to create multiple versions of forecasts, the Forecast Type field (TYPF) was expanded from two characters to eight. This allows users to create types with more descriptive titles, such as 01JUN97. Eventually, the requirement is to provide the ability to compare two versions of a forecast side-by-side on the same screen and provide some means for version control. The Version field (FVER) was added to both the F3400 and F3460 in anticipation of a possible enhancement in EnterpriseOne.

To solve the Year 2000 problem and to allow EnterpriseOne programs to access data on the AS/400, the year field (MFYR) was taken out of F3460. As a result, F3460LC is obsolete. All the programs in forecasting and planning that used F3460LC were modified accordingly. In addition, logic was modified to no longer write a 'year>

record to F3460. This primarily affected P34201, which now rolls up the detail records as required.

### Scope

Forecasting Enhancement required modification to the following existing programs:

| Program | Program Description | Status |
|---|---|---|
| P3400 | Summary Revisions | Changed |
| P34200 | Forecast Summary Inquiry | Changed |
| V34200 | Forecast Summary Inquiry | Changed |
| P3460 | Detail Forecast Maintenance | Changed |
| P34640 | Summary Forecast Generation | Changed |
| R34640 | Summary Forecast Generation | Changed |
| P3465 | Extract Sales Order History | Changed |
| P34650 | Forecast Generation | Changed |
| R34650 | Forecast Generation | Changed |
| R34651 | Exponential Smoothing | Changed |

## Lot Trace/Track

The following enhancements to Lot Trace/Track logic were included in version A7.2 and above:

- Top, Bottom, and Inventory Issue transactions are indicated in the second Description of the Trace/Track Inclusion Rules (UDC table 40/DC). The 2nd Description definitions previously were:
  - Top = C
  - Bottom = B
  - Issue = T

  Inclusion rules were changed to:
  - Top = C
  - Bottom = B
  - Completion = M
  - Issue = T
- The Transaction Reference field of the Item Ledger record is no longer used to identify issue or completion transactions

- The Parent Lot field of the Item Ledger record may or may not be filled in for inventory issue transactions. If it is populated for issue transactions, it is used to perform trace and track logic for serial-numbered items only. If it is not populated, the document number and document type will be used for trace and track logic
- Only two kinds of transactions populate the Parent Lot field in Item Ledger: Inventory Issues (sometimes) and Item Reclass
- A level is caused by an inventory issue, work order completion, or item reclass
- Inventory issue transactions are consolidated by lot number, document type, document number, item, branch, and parent lot
- Inventory completion transactions are consolidated by lot number, document type, document number, item, branch, and parent lot. For example, if lots A and B are the same issued part number and lot C and D are the same completed part number, then in A7.1 release does not exists to determine that lot A was used only to produce lot D. A track function on lot A shows lots C and D. A trace on lot D shows lots A and B. A7.3 release can provide direct component to parent associations, but only for serialized items
- Item reclass transactions are the only Item Ledger transactions to populate the from/to field. Reclass transactions involving lots populates the parent lot field
- The Lot Trace/Track program incorporates the use of two indexes. The first index is used for issue and completion consolidation, and the second one is used to define the bill of material-like tree structure for display purposes

## Process Blending

The integration of bulk requirements into the Manufacturing system included changes to the Shop Floor Control and Product Data Management modules. These enhancements make it possible for ECS clients to set up bills of materials and enter/process work orders that contain bulk products. Bulk products in the ECS industry require the entry, calculation, and tracking of various factors related to quantity measurements and blending specifications.

### Objectives

- Bill of Materials: There is a requirement to be able to enter a percent-based formula that does not add up to 100% within a given tolerance due to items that would cause the result to exceed 100% (such as PPM additives). Additionally, there could be a need for a bill of materials to contain several process steps with ingredients that are outside of the percent formula. These requirements are handled within the Ingredients List Revisions program (P3002P) and the Bill of Materials Entry (P3002), which allows the entry or change of the ingredients. These programs allow a percent definition to be added to 100% and additional ingredients to include the PPM additives or other quantities that are not based on percent. The Percent Bill Validation field (IAPBVD) in the Manufacturing Constants controls whether this is allowable.

- A function key is provided from the Work Order Ingredients List program (P3111) that calls a program that is defined and created by the user to optimize the ingredients
- Issues and Completion of work orders were modified to allow the entry of ambient quantities, current temperature, and density information. The standard quantity and weight are calculated and stored accordingly
- The Work Center Master allows the designation of the Blend/Fill Location (IWCOLO). This location is validated against the Tank Master if the work center is designated as a blending location
- The Allowed Products Matrix is checked when the parts list is generated for a work order, and a warning message is printed or displayed if an incompatibility exists between the parts list and the blending tank
- The Allowed Products Matrix is checked when inventory is issued to a blending/filling location. The allowed product from the blending tank is checked against the ingredients list at issue time and an error is printed or displayed if an incompatibility exists
- The tank capacity is checked when work orders are entered and when the product is completed to a blending, holding, or storage tank. An error message is displayed if the transaction would cause the tank's capacity to be exceeded
- The Engineering Change Management (ECO) menu contains approval routings by Branch/Plant and/or by specific ECO Work Order. A new menu was created and some minor program modifications have been made to allow these programs to be used to handle the work order approval

## Scope

Process Blending required the following new programs as well as modifications to existing programs:

| Program | Program Description | Status |
|---|---|---|
| P3002 | Enter/Change Bill | Changed |
| P3002P | Process Resource Revisions | Changed |
| P3006 | Enter/Change Work Center | Changed |
| P3009 | Manufacturing Constants | Changed |
| P3111 | Ingredients List Revisions | Changed |
| P3111P | Ingredients List Revisions | Changed |
| P31113 | WO Inventory Issues | Changed |
| P31114 | Work Order Completion | Changed |
| P31115 | Co/By Products Completion Window | Changed |
| P3H23 | Super Backflush | Changed |

| Program | Program Description | Status |
|---|---|---|
| P31410 | Work Order Generation | Changed |
| P31420 | Batch WO Inventory Issues | Changed |
| P41513 | Batch Write Bulk Item Ledger | New |
| P41513W | Bulk Product Information | New |
| P415104 | Weighbridge | Changed |
| P41514 | Bulk Manufacturing Gain/Loss and Exceptions | New |
| P41514E | Bulk Manufacturing Gain/Loss and Exceptions | New |
| P48013 | Enter/Change Order | Changed |
| P48185 | WO Approval/Audit Review | Changed |
| X41515 | Product Compatibility Server | New |
| X41516 | Tank Capacity/Compatibility Server | New |
| XF41511 | Bulk Cardex Server | Changed |

## Engineering Change Orders

The enhancements made to the ECO system include improved revision level control, a new inquiry to display future BOMs by including pending ECOs, and ability to process ECOs across multiple plants. Other enhancements include the ability to perform all Bill of Material (BOM) maintenance through the use of Engineering Change Orders (ECO) and improvements to the approval process.

### Objectives

- Simplify revision level control
- Tighten security surrounding the approval process
- Send e-mail to the originator when an ECO is rejected or implemented
- Issue a warning when an order-specific approval list is being deleted
- Display revision levels consistently in screens and reports Add Item Revision Level to the Purchase Order Detail screen Allow ECOs to impact multiple branch/plants
- Provide for an inquiry to a BOM at a specified date in the future, incorporating all pending ECOs
- Expand the Drawing Revision Level field from two to three bytes throughout the system
- Allow maintenance of all BOM fields through the ECO Population program

- Provide an option to update the parent's Component Revision Level in all structures in which the parent is a subassembly when its Item Revision Level changes

### Scope

The scope of the ECOs included the design, development, and documentation of the creation and modifications of related ECO programs, screens, and files. Functionality requires modifications within the Product Data Management (PDM), Procurements, and Inventory Management systems.

| Program | Program Description | Status |
|---|---|---|
| P3002 | Bill of Material Revisions | Changed |
| P3002W | BOM Selection Inquiry | Changed |
| P3013 | ECO Parts List | Changed |
| P30131 | ECO Related Items | Changed |
| P30200 | Bill of Material Inquiry | Changed |
| P30201 | Where-Used Inquiry | Changed |
| P30210 | Future Bill Inquiry | New |
| P30410 | Single-Level Bill of Material Report | Changed |
| P30415 | Multi-Level Bill of Material Report | Changed |
| P30420 | Where-Used Report (Single and Multi-Level) | Changed |
| P30510 | Bill of Material Population Program | Changed |
| P34500 | Purchase Order Print | Changed |
| P4101 | Item Master Revisions | Changed |
| P41026 | Item Branch Information Revisions | Changed |
| P430112 | Purchase Order Detail Additional Information | Changed |
| P4818 | ECO Approval | Changed |
| P48182 | ECO Order Specific Routing | Changed |
| P48185 | ECO Approval/Audit Review | Changed |

## Repetitive Manufacturing

Repetitive manufacturing is often characterized by entire production lines being dedicated to a family of products. Product families share similar components and routings. Similar components require less inventory movement to and from the line. Similar routings allow work center set up and changeover times between related

products to be kept to a minimum. With constantly increasing competition, repetitive manufacturers are continually seeking tools to reduce non-value-added functions and lower inventory levels while improving product quality.

Many repetitive manufacturers use rate-based scheduling as a method to control production. However, as they become more sophisticated, the next step is to produce based on actual demand rather than forecast. The authorization to produce is derived from demand rather than the traditional use of work orders or rate schedules. Scheduling production lines requires tools to schedule, sequence, and balance production based on capacity for each production line. The fundamental paradigm shift is thinking in terms of production lines rather than traditional work orders or rates.

Material movement is also driven by demand and controlled by visual cues known as kanbans. Kanbans are predetermined quantities of components at specified locations on the production line. They are designed to minimize work-in-process (WIP) inventories. Movement of kanbans is oriented around the production line itself instead of specific rates or work orders.

In addition to enhancing the product to support repetitive manufacturing, consideration must be given to manufacturers at all points on the discrete to repetitive continuum. Where possible, the gap between rate-based scheduling and the discrete work order functionality should be bridged to provide more consistency. This results in not only a path to take the next step toward true repetitive manufacturing, but also an improved solution for manufacturers currently in mixed-mode environments.

### Objectives

- Develop a workbench to schedule production in a repetitive environment. This should include the ability to classify products into families and use the classification as a basis for sequencing production. Scheduling should be able to balance production across multiple lines as well as within minimum and maximum rates defined for a specific line
- Support production capacity and load in units per hour in addition to hours per unit
- Implement the ability to define shop floor calendars by production line
- Allow for the definition of multiple replenishment points for a production line
- Provide tools to support electronic kanban control at the consuming and supplying locations. Incorporate enhancements to Procurements that allow control over the creation of purchase orders
- Implement EDI transactions to enable sending supplier schedules and releases
- Provide a method to delay payment for purchased items until consumption (Pay on Consumption)
- Combine functionality between work orders and rate schedules to provide a more seamless solution. This should include developing a comprehensive inquiry structure with drill down capabilities

## Scope

Repetitive Manufacturing required the following new programs as well as modifications to existing programs:

| Program | Program Description | Status |
|---|---|---|
| P00071 | Shop Floor Calendar | Changed |
| P0007AX | F0007 File Conversion | Changed |
| P061181 | Payroll Time Transactions | Changed |
| P3003 | Enter/Change Routing | Changed |
| P3006 | Work Center Revisions | Changed |
| P3007 | Resource Units Revisions | Changed |
| P3007G | Resource Units Refresh | Changed |
| P3009 | Manufacturing Constants | Changed |
| P3016 | Kanban Master Revisions | New |
| P30450 | Kanban Calculation | New |
| P3102AX | F3102 File Conversion | Changed |
| P3104 | Enter/Change Rate Schedule | Obsolete |
| P3104AX | F3104 File Conversion | Obsolete |
| P3109 | Enter/Change Rate Schedule | New |
| P31093 | Line Item Relationships | New |
| P31H | Parts List | Changed |
| P31113 | Inventory Issues | Changed |
| P3112 | Work Order/Rate Routing | Changed |
| P3112AX | F3112 File Conversion | Changed |
| P31121 | Order Hours Status | Obsolete |
| P31122 | Order Quantities Status | Obsolete |
| P311221 | Hours and Quantities Trans. | Changed |
| P3H23 | Super Backflush | Changed |
| P31124 | Line Balancing Review | Obsolete |
| P3114 | Rate Schedule Workbench | Obsolete |
| P3H9 | Rate Schedule Workbench | New |

| Program | Program Description | Status |
|---|---|---|
| P31220 | Dispatch List | Changed |
| P31223 | Production Status | Changed |
| P31224 | Line Balancing Review | Obsolete |
| P31225 | Shop Floor Workbench | Changed |
| P31410 | Work Order Processing | Changed |
| P31422 | Post Hours and Quantities | Changed |
| P31440 | Kanban Cards | Obsolete |
| P3152 | Line Balance Review | New |
| P3153 | Rate Schedule Workbench | New |
| P3154W | Split Lines Window | New |
| P3155W | Alternate Line Window | New |
| P3156 | Line Sequencing Workbench | New |
| P3157 | Kanban Supply | New |
| P3158 | Kanban Consumption | New |
| P3159 | Line Dispatch List | New |
| P31802 | Manufacturing Accounting | Changed |
| P31804 | Variance Generation | Changed |
| P31842 | Rate-Based Accounting | Obsolete |
| P3190 | Work Order/Rate Repost | New |
| P3197 | Production Order Commitments | Changed |
| P3411 | Message Review | Changed |
| P3482 | Master Schedule Planning | Changed |
| P3483 | Multi-Plant MPS | Changed |
| P4021 | Supply and Demand Inquiry | Changed |
| P4051 | Supply/Demand | Changed |
| P41013 | Manufacturing Data | Changed |
| P41027 | Manufacturing Data | Changed |
| P4312 | Purchase Order Receipts | Changed |
| P43500 | Purchase Order Print | Changed |

| Program | Program Description | Status |
|---|---|---|
| P47152 | EDI 862 Transaction | New |
| P470621 | EDI 830 Transaction | New |
| P48013 | Work Order/Rate Revisions | Changed |
| P48016 | Work Order Cat Codes | Changed |
| XT310911 | Quantity Detail Server | New |

## Notes

### Shop Floor Calendar

The shop floor calendar was modified to include Shift and Calendar Name as part of the key. These fields are used in conjunction with the Work Center Master to allow alternate calendars for work centers. Existing data in the Work Day Calendar file (F0007) is converted with blanks in the new fields. When both Shift and Calendar Name are blank, it is considered to be the base calendar.

Shift-specific and named calendars are overrides to the base calendar. Only those calendars that are set up for specific shifts are considered active. For example, if calendars are only set up for shifts 1 and 2, then shifts 3, 4, 5, and 6 are considered inactive.

Named calendars are referenced by work centers. If a named calendar is specified in the Work Center Master, the system looks only for calendars with the specified name for that work center. Therefore, there is no default back to the base calendar. To use a named calendar in Repetitive Manufacturing, a separate calendar for each shift is required (including the calendar name in each).

### Manufacturing Constant Hours

The Manufacturing Constants field Work Hours Per Day was expanded to six shift hours fields. The hours are used in conjunction with the shop floor calendar to generate the resource units. Additionally, the field is used in Lead-Time Calculation, Message File Revisions, Master Planning Schedule, Work Order Entry, SOE Work Order Processing server, and the Back/Forward Schedule copy subroutine. The work hours for a given day is assumed to be the total of the first three shift hours fields: WRHR, WRH2, and WRH3. This total is not stored, but is displayed on the Manufacturing Constants screen.

### Replenishment Location

The new Replenishment Location field was added to the Routing Master and Shop Routing files. Previously, when the parts list was built for a work order, the issue from location was determined by the Commitment Control field logic. This could

have been superseded by entering a location value for the work center. The new field allows for multiple locations to be defined within a parent item's routing. Each component location on the Parts List is determined by the routing step referenced in its operations sequence field. This location supersedes the value in the Work Center Master field if one had been entered also.

### Capacity in Units/Hour

The Work Center Master file was modified to include standard, minimum, and maximum rates per hour and the related unit of measure (UOM) when defining production lines. These values, their calculated extension in the Resource Units file, and the new Line/Item Relations file are used to support the statement of capacity and its consumption in terms of units/hour. Work orders and rates entered on production lines represent load against the line in terms of units /hour as well. However, for purposes of minimizing impact on the cost roll-up and scheduling processes, the values stored in the Resource Units and Routing files is converted to hours/unit.

### Online Receipts

Flags were added to the Item Master/Item Branch and kanban Master files. These flags, when turned on, cause the Issues and kanban Check-In programs to invoke the Online Blind Receipts process. The quantities received equal the issue quantity and the kanban size respectively.

### Rate Schedule Master Redesign

The Rate Schedule Master file (F3104) is obsolete. Rates now are stored in the Work Order Header file (F4801). A new file, Line/Item Relationship (F3109), was added. The purpose of the file is to define the parameters for an item when built on a given line. In addition, one record can be used to identify the default line information for use by planning. This is similar in nature to the Rate Generation Rules functionality.

### Change to EDI 830

The existing Outbound 830 transaction was previously built by using the Forecast file (F3460). A processing option was added to use the Supplier Release Schedule file (F3430) instead. The same output files are used.

## Work Center Master

Work centers now are associated with branch/plants. This required the addition of the existing MMCU field to the key. Therefore, the work center number by itself is no longer a unique value. This modification also required a new field, Issue Branch/Plant (LOMC), to be used with the Issue Location field (LOCN).

This change affects all programs that access the Work Center Master file (F30006).

To convert the existing files, the Routing Master file (F3003) determines which branch/plants to create work center records for. Each unique occurrence of the work center within a branch/plant defined in the F3003 results in a record in the converted F30006.

The Work Center Rates file (F30008) is converted to match the F30006. If the work center was used in multiple branch/plants, a copy of the rates is created in each.

# Human Resources and Payroll systems

This section provides database and system changes for the Human Resources and Payroll systems.

## Database Changes

This chapter lists file level database changes for the Human Resources and Payroll systems.

### Summary of Database Changes

Double-click the following icon to open the Human Resources and Payroll Database Changes Summary.

### Changed Physical File Layouts

Double-click the following icon to open the Human Resources and Payroll Changed Physical Files Detail.

### New Logical Files for Existing Physical Files

Double-click the following icon to open the Human Resources and Payroll New Logical Files for Existing Physical Files.

### Changed Logical Files

Double-click the following icon to open the Human Resources and Payroll Changed Logical Files Detail.

## Copy Member Changes (non-Database)

| Rptg Code | Copy Member | Func-tion | Description | Action | Release/ Cume |
|---|---|---|---|---|---|
| 07 | C06012 | COPY | Derive Payroll Account Number if using Position Control | Changed | A73CU10 |
| 07 | C06083 | COPY | Calculate Fiscal/Anniversary Date | Changed | A73CU09 |
| 07 | C0609 | COPY | Create Acrued Wage DBA | New | A91 |
| 07 | C06106 | COPY | Load EE labor distribution Instruction to Data Structure | Changed | A73CU10 |
| 07 | C06106 | COPY | Load EE labor distribution Instruction to Data Structure | Changed | A73CU16 |
| 07 | C06106 | COPY | Load EE labor distribution Instruction to Data Structure | Changed | A91 |
| 07 | C06391 | COPY | Calculate Fiscal/Anniversary Date | Changed | A73CU06 |
| 07 | C06391 | COPY | Calculate Fiscal/Anniversary Date | Changed | A73CU09 |
| 07 | C06391 | COPY | Calculate Fiscal/Anniversary Date | Changed | A91 |
| 08 | C06INTL | COPY | International Company Determination Copy Module | Changed | A73CU09 |
| 08 | C0803 | COPY | Calculate Salary and Hourly Rate | Changed | A73CU07 |
| 08 | C0803 | COPY | Calculate Salary and Hourly Rate | Changed | A73CU12 |
| 08 | C0803 | COPY | Calculate Salary and Hourly Rate | Changed | A73CU13 |
| 08 | C0803 | COPY | Calculate Salary and Hourly Rate | Changed | A91 |

## Server Changes

Double-click the following icon to open the Human Resources and Payroll New and Changed Server Programs.

## Human Resources and Payroll Changes

This chapter describes changes made to Human Resources and Payroll systems.

# A8.1 Changes

## Server Changes

### X08100 - Position Control Monitor (Changed)

This server records position activity in the Position Activity File (F08111). This server was modified to receive the Contract/Calendar Code (CNCL), the Daily Rate of Pay (DROP), and the Contract/Calendar Salary (CTSL) from the programs that call it, and to include them when updating the Position Activity File (F08111).

### X08101 - Projected at Year End Values Server (Changed)

This server calculates the impact of position activity on a position budget at year-end. It was modified for employees with a contract/calendar to calculate percentage of days worked in the contract/calendar for the fiscal year and the projected salary for those same days.

### X08102 - Position Budget Edit Server (Changed)

This server performs the budget check when the position budget edit constant is set to Y in the HR Constants File (F08040). It was changed to base edits on a contract/calendar when employees are attached to one.

### X060116 - Update Employee Master Server (Changed)

This server updates the appropriate record in the F060116 file to match any changes that were made to the primary job in the F060118 file. It was changed to update the F060116 fields that do not exist in F060118. The use of the F060118 file was removed so that the calling program can use the F060118 file in any key order.

### X089321 - Contract/Calendar Salary Server (New)

This server provides all of the salary calculations and synchronization necessary when the contract salary, calendar, or pay stop date is changed for a contract/calendar employee.

### X089302 - 'Synch' Server (New)

This server was modified to only change the keys on the F08932 file for secondary jobs. Secondary jobs should not affect the Accrual file.

### X089303 - Salary Paid Server (New)

This server was modified to use the start and stop dates from the Contract Salary Work File (F08932) instead of the Contract/Calendar File (F08930) when determining the earliest start date and latest stop date for calendar records attached to an employee job.

### X08932 - Update F08932 Server (New)

This server was modified to use the new PENDING member in the Contract Salary Work File (F08932) rather than a user space to update F08932.

# Localization

This section provides database and system changes for the Localization systems.

## Database Changes

This chapter lists file level database changes for the Localization systems.

### Summary of Database Changes

Double-click the following icon to open the Localization Database Changes Summary.

### Changed Physical File Layouts

Double-click the following icon to open the Localization Changed Physical Files Detail.

### New Logical Files for Existing Physical Files

Double-click the following icon to open the Localization New Logical Files for Existing Physical Files.

### Changed Logical Files

Double-click the following icon to open the Localization Changed Logical Files.

# Copy Member Changes (non-Database)

| Rptg Code | Copy Member | Func-tion | Description | Action | Release/ Cume |
|---|---|---|---|---|---|
| 70 | C700400 | COPY | Enable Tax Withholding - USRIDX | New | A91 |
| 70 | C700401 | COPY | A/B Tax Withholding Codes USRIDX | New | A91 |
| 70 | C700402 | COPY | Tax Codes Server | New | A91 |
| 70 | C700403 | COPY | Tax Code Definition Server | New | A91 |
| 70 | C700404 | COPY | Tax Code Concept Server - User Index | New | A91 |
| 70 | C700405 | COPY | Tax Code Precedent Server - User Index | New | A91 |
| 70 | C700406 | COPY | Base Reduction Server - User Index | New | A91 |
| 70 | C700407 | COPY | Tax Withholding Limit Amount Server - User Index | New | A91 |
| 70 | C700408 | COPY | Supplier Reduction Server - User Index | New | A91 |
| 70 | C700411 | COPY | Voucher Tax Withholding Entry Server | New | A91 |
| 70 | C700412 | COPY | Address Book Members - User Index | New | A91 |
| 70 | C700415 | COPY | Payments done to other companies Server - User Index DS | New | A91 |
| 70 | C700416 | COPY | Accumulated Amount Server - User Index | New | A91 |
| 70 | C700418 | COPY | Address Book Category Codes Tag File Server | New | A91 |
| 70 | C700419 | COPY | Payments to other companies Detail Doc. Server | New | A91 |
| 70 | C700430 | COPY | Withholding - Structure of input data | New | A91 |
| 70 | C700431 | COPY | Tax Withholding Calculation Server | New | A91 |
| 70 | C7004570 | COPY | A/P Common Routine User Index Pre-Payment | New | A91 |

| Rptg Code | Copy Member | Func-tion | Description | Action | Release/ Cume |
|---|---|---|---|---|---|
| 70 | C700476 | COPY | /COPY for Pre-Payment Processing | New | A91 |
| 70 | C700477 | COPY | /COPY for Pre-Payment Processing | New | A91 |
| 70 | C700482 | COPY | Void payment - Tax Voucher accumulation | New | A91 |
| 70 | C70DOC | COPY | /COPY for User Index IXTWDOC | New | A91 |
| 70 | C70ERR | COPY | Errors in Payment group | New | A91 |
| 74R | C74R0321 | COPY | /COPY for User Index of P74R0321 | New | A91 |
| 74R | C74R0411 | COPY | Country Server - Voucher Entry User Index | New | A91 |
| 74R | C74R0415 | COPY | /COPY for User Index of X76A0415 | New | A91 |
| 74R | C74R0900 | COPY | Correspondence Constants - USRIDX | New | A91 |
| 74R | C74R0901 | COPY | Correspondence Account set up - USRIDX | New | A91 |
| 74R | C74R0902 | COPY | Correspondence Method - USRIDX | New | A91 |
| 74R | C74R0911C | COPY | Correspondence Detail - Credit - USRIDX | New | A91 |
| 74R | C74R0911D | COPY | Correspondence Detail - Debits - USRIDX | New | A91 |
| 74R | C74R0911E | COPY | Correspondence Detail - Errors - USRIDX | New | A91 |
| 74R | C74R0911H | COPY | Correspondence Detail - Intercompany Accounts by CO - USRIDX | New | A91 |
| 74R | C74R0911I | COPY | Correspondence Detail - Intercompany Accounts - USRIDX | New | A91 |
| 74R | C74R0911K | COPY | Correspondence Detail - Debits/Credits NN - USRIDX | New | A91 |
| 74R | C74R11A | COPY | Petty Cash Desk | New | A91 |
| 74R | C74R11B | COPY | Petty Cash Desk | New | A91 |

| Rptg Code | Copy Member | Func-tion | Description | Action | Release/ Cume |
|---|---|---|---|---|---|
| 74R | C74R11C | COPY | Petty Cash Desk | New | A91 |
| 74R | C74R11D | COPY | Petty Cash Desk | New | A91 |
| 74R | C76A4115 | COPY | Country Server - Recalculate Pay Item Amount | New | A91 |
| 76 | C7615 | COPY | Edit Transaction Nature w/ICMS Tax Substitution Mark-up | New | A73CU10 |
| 76 | C7615 | COPY | Edit Transaction Nature w/ICMS Tax Substitution Mark-up | Changed | A73CU11 |
| 76 | C7615 | COPY | Edit Transaction Nature w/ICMS Tax Substitution Mark-up | Changed | A73CU13 |
| 76 | C7615 | COPY | Edit Transaction Nature w/ICMS Tax Substitution Mark-up | Changed | A73CU16 |
| 76 | C7615 | COPY | Edit Transaction Nature w/ICMS Tax Substitution Mark-up | Changed | A91 |
| 76 | C76STT | COPY | /COPY for Brazilian - Algorithm Tax Situation | Changed | A73CU13 |
| 76 | C76STT | COPY | /COPY for Brazilian - Algorithm Tax Situation | Changed | A73CU16 |
| 76 | C76STT | COPY | /COPY for Brazilian - Algorithm Tax Situation | Changed | A91 |
| 76A | C42PRT | COPY | SOP Invoice Print - CSE Print Fields | New | A73CU15 |
| 76A | C42PRTECL | CPYL | ECS Invoice Print - CSE Print Fields | New | A91 |
| 76A | C76A0110I | COPY | /COPY for User Index of X76A0110 | New | A73CU15 |
| 76A | C76A03105I | COPY | Definition of User Index - AR Voucher Entry | New | A73CU15 |
| 76A | C76A0411I | COPY | Country Server - Voucher Entry User Index | New | A73CU15 |
| 76A | C76A0411I | COPY | Country Server - Voucher Entry User Index | Changed | A91 |
| 76A | C76A0413I | COPY | User index for F76A0413 | New | A91 |

| Rptg Code | Copy Member | Func-tion | Description | Action | Release/ Cume |
|---|---|---|---|---|---|
| 76A | C76A0493I | COPY | /COPY for User Index of X76A0493 | New | A73CU15 |
| 76A | C76A18I | COPY | /COPY for User Index TX76A18 | New | A73CU15 |
| 76A | C76A19I | COPY | User Index Drivers for F76A19 | New | A73CU15 |
| 76A | C76A81DRPT | COPY | DREAM Writer - Dynamic Report Processing | New | A73CU15 |
| 76A | C76ACUIT | COPY | CUIT Edition Routine | New | A73CU15 |
| 76A | C76AMINI | COPY | /COPY for User Index TXMIN | New | A73CU15 |
| 76A | C76ATAXI | COPY | Store Tax Calculation Information | New | A73CU15 |
| 76B | C76B11 | COPY | Right Justify Short Item Number | New | A73CU16 |

# Server Changes

Double-click the following icon to open the Localization New and Changed Server Programs.

# Localization Changes

This chapter describes changes made to Localization systems.

## A9.1 Changes

## Argentina

### Enhancements

- To include Argentina localization as part of pristine environment. SAR 7901194
- New withholding computing module. SAR 7901207

  A new withholding module was developed and replaces the existing localization from previous releases. This new withholding computing module

was designed to be very flexible and user configurable. It allows implementing most future legal changes related to localizations without the need of installing an upgrade or waiting for the release of a localization PTF changes to programs.

This new withholding module is not just a redesign of the existing solution; it also introduces enhancements to it:

- Withholding computing by company and/or vendor tax Id (CUIT) or by company and/or vendor number
- User-defined document type to identify withholdings, not hard coded
- UTE composition can vary for each withholding type
- Withholding rates can be defined by A/B withholding category codes
- Withholding log file to track how each withholding was computed with record of all amounts involved in the computing (taxable amount, rate, special deductions that took part, and so on.)
- Withholding certificate printing can be done by a user-defined program. No programming changes are required to get it executed
- Withholding registry can create automatic record to fiscal authority A/B number and payments can be done from those records

For customers upgrading from previous releases, the upgrade process takes care of converting the set up and transactions from the old withholding localization to the new one. The 95 % of the set up work for the new withholding module is solved by the upgrade process.

- New withholding certificate next number definition. SAR7901194

  The withholding certificate next number set up was consolidated into a new set of files. This new scheme works together with the new generic withholding module and it is more flexible to allow users to switch on/off the elements that takes part of the next number individually (fiscal year, issue place, state). This new next number scheme is used by all withholding types.

  It also simplifies the maintenance of the next number for the certificates in comparison to the existing solution in previous releases.

- New withholding certificate printing program. SAR 7901194

  A new withholding certificate printing program was developed to take advantage of the new withholding database, new withholding certificate next number and to add flexibility for user-defined changes to header and footer text.

  All withholding types are now printed or re-printed by using the same program.

  Some redesign to the format was done to add flexibility but without removing information from it which is mandatory by the national legislations. An exception to this rule is the gross income-withholding certificate. As it is not ruled by a national legislation, the one that is being printed by the localization is intended to be used as a model for custom creating the specific withholding certificates required by the states where the company operates.

It is very simple to set up the use of the custom withholding certificate, just add its name to the set up for the particular withholding type and that is all. No custom programming to localization is required to activate it.

## Brazil

### Enhancements

- Computing of new taxes PIS / COFINS / ISS. SAR 7952071

  Localization to compute and register PIS / COFINS / ISS taxes for purchases and sales is provided. A new set up was created to provide a flexible way to define those taxes and to simplify future changes to them.

- Transaction Nature Code (CFOP). SAR 7952039

  The four digits CFOP code handling is included in A9.1 release. It replaces the previous solution based on a three digits CFOP.

  Fiscal reporting is done based on the four digits CFOP too.

  The upgrade process takes care of converting the three digits to four digits CFOP if upgrading from A8.1 or A7.3 SP15 or earlier.

  **Note:** Customers with A7.3 SP 16 release already have this localization enhancement

- Tax / Withholding computing. SAR 7901207

  Companies in Brazil must calculate income and social security taxes for the Fiscal Notes (supplier voucher) that they receive. These taxes are calculated on an aggregate basis or on a retention basis. When calculated on an aggregate basis, the tax is added to the basis of the total amount on the Nota Fiscal or voucher. The tax is remitted to the government, not to the supplier. When calculated on a retention basis, the tax is an amount that is withheld from the Nota Fiscal/voucher payment. In this situation, a portion of the amount that is due to the supplier (the tax) is remitted to the government instead of the supplier.

  Withholding localization was included in the A9.1 release. A very flexible module was created to be able to compute payment withholdings or additional taxes for the voucher.

  The taxes or withholdings supported by the localization are:

  - IR (Imposto de Renda): A federal income tax that is levied on services that are provided by individuals or legal entities
  - ISS (Imposto sobre Serviços): A municipal tax varies according to the city of origin of the job or service
  - INSS (Instituto Nacional do Seguro): The national social security tax affects to individuals or legal entities

- FUNRURAL (Fundo de Assistência e Previdência do Trabalhador Rural): The income tax for agricultural businesses. It is a rate to apply to the Nota Fiscal amount
- PIS/PASEP (Programa de Integração Social/Programa de Formação do Patrimônio do Servidor Público): A mandatory contribution that is levied as a percentage of monthly billings. The PIS contribution is done to Brazilian social programs by private companies and enterprises. The PASEP contribution is made to Brazilian social programs by public or government entities
- COFINS (Contribuição para Financiamento da Seguridade Social): It is a mandatory contribution that is levied as a percentage of monthly billings on merchandise and services
- CSLL (Contribuição Social sobre o Lucro Líquido): A tax on net gains

- General Ledger reports. SAR 7952047, 7952055

  New General Ledger reports are included in the A9.1 release. They are:

  - General ledger
  - Transaction journal
  - Account balance
  - Accounts Receivable transaction ledger
  - Accounts Payable transaction ledger

  **Note:** Customers with the A7.3 SP 16 release already have this localization enhancement.

- Fiscal Reporting. SAR 7952047, 7952055, 7952063

  New Fiscal reports for input / output fiscal notes transactions were included in the localization. The following reports are included:

  - Municipal DIPAN declaration
  - Municipal DECLAN declaration
  - ICMS statement
  - IPI statement
  - GIA tax collection
  - ICMS by state for input and output transactions
  - Input / Output transactions register for Industry | Commerce
  - DIPI register
  - GIA ICMS Input / Output transactions
  - Inventory register
  - Goods coding table
  - Stock production register
  - Flat file generation for ICMS

- Flat file generation for inter state transactions
- Flat file generation for collection national guide

- IN86 reporting. Localization to extract, update, and flat file generation to fulfill IN86 requirement is provided

**Note:** Customers with the A7.3 SP 16 release already have this localization enhancement.

## Spain

### Enhancements

- To include Spain localization as part of pristine environment. SAR 8088027

# Programmer's Tools and Considerations

This section provides the following:

- Performance considerations for programmers
- National language support
- Double-byte enablement
- Scanning tool for year-2000 date fields

## Performance Considerations for Programmers

### What Makes an Application Run Slowly

The way programs use computer resources, deeply affect response times (or batch run time). If a program does many expensive instructions for each transaction, it uses a large part of the CPU and the response time is poor. If a program does many input/output (I/O) operations, it spends a large amount of time waiting for data to be transferred between the disk and the CPU, and response time is slow.

Sometimes it is difficult to determine what turns out to be "a lot of I/O" when the program runs at a customer site. Customers run our software against thousands or millions as many records as we have in our test data files. Customers set up their operation such that they incur many expensive instructions for each transaction record processed. Customers often run our software on computers much slower than our development machine.

You need to be aware of how JD Edwards World can make programs run faster and minimize the use of expensive resources. Because the software architecture is modular (this is a popular concept in the software industry under the buzzword object-oriented), it is possible to cause a lot of work to be done inadvertently that should be simple processing.

### Program Calls/Initialization

JD Edwards World uses program calls extensively. Some heavily used programs are:

| | |
|---|---|
| **X0028** | Date Conversion |
| **X0005** | UDC Server |
| **XF0901** | Account Master File Server |
| **X09031** | Compute Period Number |

Some common subroutines perform program calls (for example C00161 and C0000). On the AS/400, calling a program is relatively expensive. If the program being

called is a CL program, it is even more expensive. The best way to minimize this expense is to check whether we really need to call the program (again). For example, if the transaction date has not changed, do not convert it again. Use the results of the previous conversion. This is called conditioning the calls.

If you are writing a file server or application server, make it as inexpensive as possible. A call to RPG is less expensive than a call to CL. Within the RPG program, end it by setting on RT instead of LR. If you set on LR, next time the program is called, it goes through program initialization again. If you set on RT, the program stays in an initialized state.

**Note:** The variables within the program do not clear automatically. For example, if Indicator 83 was on, it stays on. You may need to add code to initialize some fields that the code currently assumes will contain zeroes and blanks the first time around. Any files that were open when the program last ended stays open.

## Common Subroutines

Some of the common subroutines can be expensive as well. The following subroutines show up in many measurements as hot spots, which are areas of code in the programs that use up more CPU time.

| Subroutine Name | Explanation |
|---|---|
| COO 12 Right Justify Numeric Fields | This general-purpose subroutine does scrubbing for many different situations. It moves your input field to a 22-entry array and scrubs that array. It looks for decimal points and date separators. Obviously, these functions are unnecessary and expensive if what you are scrubbing is a two-byte subfile option field. Even more unnecessary is to scrub an input parameter coming from another program where the other program is reading a numeric field from the data base and passing it to this program.<br><br>The subroutine formats the field in three different ways: as a 29P9 field (#NUMR); as a 15P9 field (#NUMR9); and as a 15P2 field (#NUMR2). Look at the definition of your output field to decide which of the formatted fields you should use. If your output field is 11P2, for example, and you move #NUMR9 into it, you lose three significant digits. #NUMR9 only has six significant digits. Use #NUMR instead.<br><br>If the field does not originate from a user typing it on a screen, it probably does not need to be put through COO 12. |

| Subroutine Name | Explanation |
|---|---|
| C9822/3 Double-Byte Truncation | This subroutine moves your alpha input field into an 80-entry array and scrubs it. The subroutine should only do the scrubbing if the program is running on a double-byte system, but sometimes it executes unnecessarily. It tests a flag field #DBL, which is loaded from the QJDF data area. If the option in the data area is set to '0', the subroutine does the scrubbing. (It tests for ' '). This subroutine was changed in A7.3 to test for '0' or ' '. |
| C0042/3 Right Adjust Alpha Field | This subroutine moves your input field to an 80-entry array, and painstakingly right-adjusts it. It is used every time a business unit field is typed in on a screen. (The business unit field is 12 characters.) Make sure you condition the execution of this subroutine so that it executes only if the business unit field changes. |
| COO 161 Format Numeric Fields for Output | This subroutine calls an Assembler program that examines and edits your output (inserting decimal points). There are programs that execute this subroutine for alpha fields. It does not harm the alpha fields, but it is an unnecessary expense. JD Edwards World also recommends that if a field is unlikely to change during a batch run (for example, check date), check whether it has changed before formatting it again. |

## Double-Byte Scrubbing

All JD Edwards World programs include a routine (C9822) to scrub alpha fields as needed to accommodate double-byte character data. JD Edwards World programmers analyze any new or changed RPG code for potential double-byte problems. The Double-Byte Analysis program flags lines of code where an alpha field is moved to a shorter alpha field. Because imbedded control characters exist in double-byte fields, problems occur if an alpha field in a double-byte system is truncated.

The analysis program examines the fields involved to determine whether they could contain double-byte data. If the field is defined in the data dictionary as type 'A' (contains an alpha code, such as MSA [Marital Status Actual]), or if the field is two bytes long or less, it does not contain double-byte characters and should not be scrubbed. If the field is defined in the data dictionary as type 'O' (Open), it can contain double-byte characters and may need to be scrubbed. An example of an open field is ALPH (Alpha Description).

At times, the analysis program cannot determine what the field is. Perhaps the programmer moved the field into a work field earlier in the program (for example, moving several code fields into a data structure, then later moving the data structure into a work field). If, for example, the program moves $WK22 to $KY20 (a 22-byte work field to a 20-byte work field), the analysis program flags that statement with DBX20 in the left margin. The 'X' indicates that programmer must investigate further. In this case, the programmer would find that $WK22 is made up of several smaller code fields that would not contain double-byte characters.

If an open field (or work field) is moved to an array, or an array is moved to an open field, the analysis program flags it with DBArara in the margin. The analysis program cannot determine whether truncation will occur. The programmer should analyze the code again to determine whether truncation could occur. For example, moving a 40-byte open field to an array with 30 one-byte entries would cause truncation.

If the programmer determines that there will be no truncation, the programmer should change the DBXnn or DBAnn flag to DBNnn. No extra code should be added to handle truncation.

If the programmer determines that there will be truncation, the programmer should change the DBXnn or "DBAnn to "DBYnn. The 'Y' indicates that double-byte scrubbing is needed. The programmer should make sure that the nn portion of the flag reflects the correct output length for the field after the move. This is usually needed for moves to an array for which the analyzer cannot determine the correct output length. The programmer, then, runs the source code through the double-byte conversion process. This changes the moves, which causes truncation as follows:

Original code (ABALPH is an open field 40-bytes long, and RRALPH is a report field 30-bytes long):

```
MOVE   ABALPH       RRALPH
```

New code after converting the program:

```
/COPY       JDECPY,E9822

/COPY       JDECPY,C9822

                    CLEAR       @UA
                    MOVEA       ABALPH              UA
                    Z-ADD       30          #OUTLG
                    EXSR        C9822
                    MOVEA       @UB         RRALPH
```

@UA and @UB are 80-byte alpha arrays.

Customers who modify JD Edwards World code should keep the double-byte scrubbing routine in mind. It can be expensive and should be used only where needed. If a customer is not on a double-byte system, they probably should not add double-byte scrubbing to any custom code they write.

## Database Read/Write

Database requests usually are the most expensive thing JD Edwards World programs do. The most effective way to reduce cost is to reduce the number of requests.

For batch, use sequential I/O where possible. Add record blocking for sequential I/O. One batch program was changed, it did many writes to use sequential-blocked writes. This change reduced the run time by two thirds. See *Sequential I/O* for further discussion.

### Do not Read Records Again

For example, if company number has not changed since the last transaction, there is no need to re-read the company file. See *Caching Control Files* for further discussion.

In a subfile program from which a user may select a particular record to see more detail, store the extra fields for the detail as hidden fields in the subfile. In this way, the detail code does not have to re-read the record. If necessary, the information can be passed to another program in a data structure.

### Do not Read Records Unnecessarily

Do not scan the database. If users are allowed to type record selection criteria in a subfile program, there should be supporting logical files to read only the records users are interested in. Do not read 200 records, discarding 185 of them to present 15 records to the user. The problem with adding additional logical files is that the system has to maintain them. This puts an extra load on the system.

If there are multiple subfiles defined for a program and the user can choose the ones to look at (this is usually controlled by a processing option), do not fill all the subfiles ahead of time. Fill the subfile only if the user asks for it. Customer/Vendor Ledger Inquiry programs now have this logic in place.

Where a file was normalized (for example, the F0101 Address Book file was split into eight files in A7.1), do not assume that all the new file information is needed. Do not just replace CHAIN 10101 with CHAIN 10101, CHAIN 10112, CHAIN 10113, CHAIN 10114, CHAIN 10115, CHAIN 10116, CHAIN 10301, and CHAIN 10401. Look at the fields that needed and only CHAIN to the formats that contain those fields. This is especially important for file server programs.

### Avoid Setting on the Fail Indicator

If you normally expect a record not to be there, do not CHAIN every time to find it. An example is when you use next numbers to allocate a key for your file. Next numbers usually come up with a key that does not exist in your file. If your file is defined as having a unique key, it is much less expensive to WRITE the new record with an error indicator on the WRITE (usually the write will succeed) than to CHAIN with the new key first (usually the CHAIN will fail).

Even more expensive is using SETLL where the SETLL positions past the end of the file. This forces an FEOD (RPG-abbreviated CLOSE and OPEN of the file).

For example, a control file has company as the major key. The customer has set up their control data for company 00000. The transaction records are for company 12345. For every transaction record, do a SETLL to the control file with a key of company 12345. This causes an FEOD because no records exist with a key higher than 00000. Do not find an eligible record and do the SETLL again with company 00000.

Because customers are encouraged to do generic set up with company 00000, check the control file at the beginning of the program to see if there are ANY records with a key greater than company 00000. See *Caching Control Files* for further discussion.

## Sequential I/O

If the program reads a file that has the record selection and sequencing done through DREAM Writer, you should be able to use sequential processing with that file. RPG allows the file to be opened for sequential-only processing if one of the following statements is true:

- The only OPCODE against that file in the program is a READ
- The only OPCODE against that file is a WRITE **

If the program does a SETLL, CHAIN, READE, UPDAT, or DELET against that file, it will not be opened for sequential-only processing. Check your RPG compile listing for the message "RPG will block/unblock file xxxxx" to see whether you have achieved sequential-only processing for that file. Sequential-only processing saves substantially on both CPU and I/O.

If you are reading a file, and only every tenth or twentieth record is updated (or deleted), it would be best to open the file twice. (For example, open the physical for the READ, and open the logical for the update). Read the physical. When you get to the record that needs updating, retrieve and update it through the logical.

To get the record blocking, you must add an override in the CL program:

```
OVRDBF Ynnnnnn SEQONLY (*YES xxxx)
```

Where *xxxx* is the number of records to block. To calculate this number, divide the record length into 32767 (32K) and round down. The AS/400 defaults to a block size of 4096 (4K) for sequential.

> **Note:** If there already is an override for that file in the CL, you must change the existing override by adding the extra parameters to it.

If you cannot achieve true sequential processing for a file, you still can get some benefit from blocking the file with a different override:

```
OVRDBF Ynnnnn NBRRCDS (xxxx)
```

**Note:** Even if the only OPCODE against a file is WRITE, the operating system database does not allow sequential-only processing if there is a unique key against that file.

For example, you read a master file, and for each master record, you do SETLL and READE in a transaction file. You do not get sequential-only processing on the transaction file, but you can still block it. If you read the master file in employee sequence, and access the transaction file by employee number (that is, you process both files in a similar key sequence), calculate the blocking factor as above.

If the key sequence of the two files is different, block the transaction file with something closer to the average number of records per master file record.

**Note:** If you use a much-larger blocking factor than the number of records, you are reading sequentially at a time, the job runs slower than a job with no blocking added.

### Expert Cache

The customer can do something to provide record blocking without changing any code. If the customer separates the batch jobs into their own shared pool (for example, by changing the QBATCH subsystem description to use *SHRPOOL1), they can turn on expert cache in that pool. This is an operating system function that dynamically looks at the way jobs are reading or writing data, and does its own blocking where appropriate. This can have the same effect on batch run times as adding the OVRDBF. NBRRCDS(wz) CL statement. It does not reduce CPU usage like SEQONLY processing.

To change to expert cache, do a WRKSHRPOOL and change the paging option column to *CALC.

The operating system ignores this paging option if it determines that there are not enough memory or CPU cycles to use it.

## Caching Control Files

Programs often access multiple control files for each transaction record processed. To cut down on reads to the control files, many programs have caching logic for them. Most files have caching logic as well. This usually consists of an array of 100 or more entries in which valid codes are stored. For example, to validate a deduction type in Payroll, the program first does a LOKUP in the deduction-type array. If it does not find the deduction type there, it does a CHAIN to the deduction-type file. If it finds the deduction type in the file, it adds it to the array.

Caching can be the single most-significant improvement to a program to reduce database reads and speed up the program. It also can go spectacularly wrong. In the worst case, for every transaction record, a program may search a 250-entry array (3 x 250 machine instructions), and then still have to CHAIN to the control file.

Some customers stripped the caching logic out of a program and see significant improvement. Other customers added their own caching logic to a program that had none, and see significant improvement.

### What Goes Wrong?

The biggest problem JD Edwards World faces is that customers set up and use a variable number of control records. For example, if every customer set up no more than 100 pay types, the caching would work fine. Customers who set up more than 100 types may get no benefit from a cache with 100 entries. If we increase the size of the array to more than 100, it is no longer efficient. If JD Edwards World starts trying to use smart logic to keep the 100 types most recently used in the cache, JD Edwards World ends up spending more time managing the cache than would have by just CHAINing to the control file each time.

The second problem is that customers tend to set up control files as generically as possible. For example, if customers can get away with setting up a sales tax rate for company 00000 for the USA, they will. If not, customers set it up by state for company 00000. If that is not specific enough, customers set it up by county for company 00000. Only as a last resort, customers set it up for each company separately. Because JD Edwards World does not know how granular their definition is, JD Edwards World programs look for the most specific value first, then gradually work down to the most general value as each CHAIN or SETLL fails.

A program may do more than ten failed CHAINs or failed SETLLs to find the applicable control code for one field in a transaction record. If the program has caching for this file, and it stores the answer as it found it in the database (that is, the sales tax rate for company 00000 for Jefferson County is .034), go through a cache search followed by all the failed CHAINs all over again for each transaction record. Store the answer in the cache the way the question was asked. When looking for the sales tax rate for company 12345 for Jefferson County, the answer was company 00000 for Jefferson County. Put Company 12345/Jefferson County/3.4% in the cache.

Also try to set flags as you go along that tell you what to look for. For example, test the sales tax rate file in S999 to see if there are ANY records for companies greater than 00000.

### User Indexes

Some programs use user indexes instead of arrays to cache control records. These have the advantage of being able to grow as needed (unlike arrays). If you can populate the array correctly (for example, if JD Edwards World stores the answers in the array the way the question gets asked), a user index READ performs about the same as a successful LOKUP on a 1000-entry array. If there is a higher failure rate on the array LOKUP, the user index starts looking attractive for a smaller number of codes (about 250).

### Pre-Loading Arrays

You can tell the number of records in a control file by looking in the file information data structure when you open the file. (See copy book I00INFDS -field FIRCNT). If you are using a 100-entry array cache, and there are 100 records or less in the control file, you can load all entries in S999 into a sorted array. This speeds up every LOKUP to the array. The downside to this technique is that you have to define two

arrays: one sorted and one not sorted. If you define the array as a sorted array, but the contents are not in the correct sequence, the LOKUP will fail.

```
 * SORTED ARRAY
E                 A     100   3     A
 * UNSORTED ARRAY
E                 B     100   3
```

## Expensive Instructions

There are some instructions in RPG, which are surprisingly expensive. You need to be cautious when you use these instructions in the subroutines that are executed repetitively. For example, be particularly careful with subroutines S004 and S005, which execute the body of instructions for each transaction record.

### Array Handling

Anything to do with array processing can be expensive because the machine executes instructions repeatedly for each array entry. If you are keeping running totals in arrays, and you initialize the arrays each time you read a new master record, pay attention to how you initialize the arrays.

Using MOVE or Z-ADD is the most expensive way to initialize an array. MOVEA is better than MOVE/Z-ADD. RESET is most efficient for a numeric array. CLEAR is most efficient for an alpha array.

Searching large arrays is expensive. If the array has to be larger than 250 entries, consider a user index instead. Whenever possible, define arrays as sorted. This speeds up the search greatly. Although LOKUP can be expensive, doing it yourself is even more expensive. For example:

```
      Z-ADD       1     #A
#A    DOWLE 100
SRCH  IFEQ        ARR,#A
      GOTO              T66
      ELSE
      ADD         1     #A
      END IF
      ENDDO
T66   TAG
```

The following code runs much faster than the above:

```
      Z-ADD       1      #A
SRCH  LOKUPARR,#A
```

Try to make your search argument the same length and type as the array element definition. If the array is defined as packed, define your search field for the LOKUP as packed, not zoned. Otherwise, the computer has to convert the field for each comparison it does.

When you need to refer to a particular entry in an array multiple times, it is more efficient to move that element into a work field and refer to the work field multiple times. Each time you refer to an element of an array, the system calculates the actual offset of the beginning of that field in the array. For example:

```
                                        30
SRCH          LOKUP         ARR,#A
*IN30         IFEQ          '1'
              ADD           ARR,#A      TOT
ARR,#A        MULT          PCT         RATE
ARR,#A        DIV           FACT        TAX
```

Performs poorly compared to:

```
                                        30
SRCH          LOKUP         ARR,#A
*IN30         IFEQ          '1'
              Z-ADD         ARR,#A      WRK
              ADD           WRK         TOT
WRK           MULT          PCT         RATE
WRK           DIV           FACT        TAX
```

The XFOOT opcode sums all the entries in an array. If you have sized your array for many more entries than it usually contains, any arithmetic operation that runs against the whole array runs slower than necessary. For example, you have given two arrays 250 entries. Most customers only use 50 entries. You add the arrays together. The system does 250 ADDs, even though 200 of the entries in each array contain zero.

```
E             A     250   150
E             B     250   150
E             C     250   150
C     A       Add   B     C
```

**Note:** It is important to size your arrays carefully.

### String Handling

When you search for one character, it is faster to scan a field than to perform an array LOKUP. When you search for a three-byte code in a list of 50 possible three-byte codes, LOKUP is faster than SCAN. In this case, it either is even better to define the array as a sorted array, and load it in ascending sequence, or sort it once loaded (SORTA opcode). If the array is defined as a sorted array, make sure the contents are sorted to avoid unpredictable results. Sometimes it is not possible to do an array lookup, such as when you search for three characters in a 40-byte field that could start anywhere in the field.

### Data Structures

If you are initializing data structures repetitively, pay attention to using the most efficient method. For a large data structure with many subfields, the CLEAR opcode generates individual instructions to move blanks to the entire data structure, then moves BLANKS and ZEROES to each individual field. RESET is a less expensive instruction to use, because it overlays the data structure with a saved copy of the initialized version (only executing one instruction). If the data structure has only a few numeric fields, consider moving BLANKS to the data structure name followed by individual Z-ADD *ZERO instructions for the numeric sub-fields.

```
IEXAMP      IDS
I                                       1  100NUM1
I                                      11  4 0 ALPH1
I                                    P 41   430PKD1
I                                      44  73 ALPH2
I                                      74  810NUM2
C                   CLEAREXAMP

                                      VS

C                             RESETEXAMP
```

**Note:** If you are using RESET and the data structure contains numeric fields, you must initialize the data structure with T on the DS statement.

Multiple-occurrence data structures are more efficient than storing related fields in multiple arrays. For example, if you are caching the UDC codes and descriptions in four arrays:

```
E     KEY   150  16     Current Values
E     D1    150  30     Current Descr.
E     D2    150  30     Current Descr2
E     SP    150  10     Special Hand.
E*
I*
I*     PROGRAM INPUT SPECIFICATIONS AND DATA STRUCTURES
I*
IDRDS DS
I                       1     4  DRSY
I                       5     6  DRRT
I                       7     16  DRKY
I*
C     DRDS  LOKUPKEY,#E             66
C     *IN66 IFEQ '1'
C           MOVELD1,#A  SFDL01
C           MOVELD2,#A  SFDL02
C           MOVELSP,#A  SFSPHD
```

In handling the relevant entries for each of the arrays, the system has to calculate where the entry is in each array. If a multiple-occurrence data structure is used instead, the system finds the start of the relevant entry once. The more related arrays there are, the more significant this becomes.

```
E                 KEY   150  16           Current Values
I*
IDRDS       DS
I                                  1  4 DRSY
I                                  5  6 DRRT
I                                  7 16 DRKY
I*
IDRDESC     DS                   150
I                                  1  30 DRDL01
I                                 31  6 0 DRDL02
I                                 61  70 DRSPHD
```

```
C        DRDS       LOKUPKEY,#A                66
C        *IN66      IFEQ '1'
C        #A        OCCUR DSDESC
C                 MOVE DRDL01       SFDL01
C                 MOVE DRDL02       SFDL02
C                 MOVELDRSPHD       SFSPHD
```

## Arithmetic

Multiplication and division are more expensive than addition and subtraction. You can make these even more expensive by causing an overflow condition. This provokes system error handling. Error handling is always expensive. Older versions of X0028 (date conversion) had a deliberate overflow as part of the code that determines whether the year was a leap year:

```
C      $FMTYR      DIV   4           $NBRV9      99
C      $NBRV9      IFEQ  .000000000
C                  MOVE  '1'         $LEAP
```

The division operation stored the result in a field with no integers defined. For example, 1985 divided by 4 equals 496.3, but this would be stored in the program as .3 because $NBRV9 was defined with no leading numbers. This is an overflow condition. The result field is too small to hold the calculated result.

Removing the overflow improved the performance of X0028 by 46%. Replacing the division operation with other operation codes resulted in an additional 10% improvement in performance. Removing the TESTN opcode (also expensive) resulted in yet another 10% improvement.

An example of using multiplication operation excessively is in the CLONE-generated code for S998:

```
CSR          MOVE F@AD    #A
CSR          DO           #A
CSR          MULT 10      #@AD
CSR          END
```

It would be more efficient to do the following:

```
CSR          MOVE F@AD    #A
CSR          SELEC
```

```
CSR   #A    WHEQ  1
CSR         Z-ADD10     #@AD
CSR   #A    WHEQ  2
CSR         Z-ADD100    #@AD
CSR   #A    WHEQ  3
…
CSR   #A    WHEQ  9
CSR         Z-ADD1000000000#@AD
CSR         ENDSL
```

Fortunately, S998 is only executed once, but be aware of how you do your arithmetic in the code which is executed repeatedly.

### Error Handling

You need to have error-handling logic in your programs. You must be careful not to make the error-handling logic main stream. For example, a CL program creates a work environment for a batch job. It checks to see if the work library is there. If not, it creates it. It checks to see if the work versions of 20 files are in the library. If not, it creates them.

```
CHKOBJ  ABC  *LIB
MONMSG  CPF9801  EXEC(DO)
CRTLIB    ABC
ENDDO

CHKOBJ  ABC/DEF  *FILE
MONMSG  CPF9801  EXEC(DO)
CRTDUPOBJ  DBF  PRODLIB  *FILE  ABC
ENDDO
```

If the normal course of events is that the batch program creates the environment and then uses it, the above logic is back to front. It is provoking error-handling logic every time it runs instead of provoking it only when there is an error. The program instead should create the new library and objects, checking for errors if they exist already.

```
CRTLIB            ABC
MONMSG  CPF2111
CRTDUPOBJ  DEF  PRODLIB  *FILE  ABC
```

```
MONMSG  CPF5813
```

## Printing

If your application requires much printing throughout the day (for example, printing pick slips), printing can be expensive. Order Entry application offers customers their choice of whether they want to print picking slips interactively (by pressing a function key), print in batch by submitting a job for each picking slip, or print in a subsystem. The third option is the best for performance.

The problem with printing interactively is that your program ties up precious resources while it prints.

Printing in batch is very expensive if the customer has a large volume of print requests. Job initiation is expensive. If a print job is submitted every five minutes, the job initiation uses a significant amount of CPU resources.

Printing in a subsystem allows one job to run all day. The job usually starts when the dedicated subsystem starts. The online program communicates with the subsystem job by way of a data queue. It puts print requests onto the data queue. The subsystem job waits on the data queue. It sleeps between print requests. It wakes up when a request arrives on the data queue, and produces the picking slip.

## Display Files (Screens)

Several customers have remote locations. Subtle changes in the way you define display files can have a big impact on response time.

Display files created before A5.2 all had the PUTOVR/OVRDTA keywords as a standard. This requires the use of the RSTDSP(*YES) option when creating the display file. The old-style windows that we used in A5.2 required the use of RSTDSP(*YES) as well. RSTDSP(*YES) can slow down response time for remote users. It also affects local users when there are remote users on the same system. With RSTDSP(*YES), the system saves a copy of everything on the screen before it puts out a new display file on that screen.

For example, a user is looking at a sales order. P4211 writes the V4211 display file to the screen. The user positions the cursor in the customer number field and presses HELP. JD Edwards World program calls the Customer Name Search program (P01NS), which writes V01NS to the screen.

If these display files are defined with RSTDSP(*YES), the system places the contents of the V4211 screen (data and constants) into a save area on the CPU before sending the V01NS image to the screen. Then when the user selects a customer and returns to P4211, the system places the contents of the screen into another save area before re-displaying (restoring) the V4211 image that it saved previously.

If this is a remote user, the save action creates a lot of extra traffic on the communications line. A larger problem for everyone is that while the save and restore actions are going on, the user's job stays in memory, using an activity level. This means that other users potentially cannot get memory to do their work. Now this user's wait for the save/restore action to complete could be added to other users' response time, because they have to wait for it to complete before they get a

turn at the CPU. If the user is on a 9600-baud line, the save/restore could take a relatively long time to complete.

This scenario shows up in performance tool reports as high Short Wait/Short Wait Extended time.

It appears that many of our display files are defined with RSTDSP(*YES) unnecessarily. The only time RSTDSP(*YES) is needed is if there are PUTOVR/OVRDTA keywords in the display file DDS, or if there is an old-style window (does not use WINDOW keyword). Unfortunately, RSTDSP(*YES) is the default on the SVR record. If the MAINT/RSTDSP field is left blank, it compiles the display file with RSTDSP(*YES).

Changing the RSTDSP attribute during testing is easy. Use the CHGDSPF command with the RSTDSP(*NO) parameter. If you need the RSTDSP(*YES) attribute, the screen gets corrupted when you take a subfile option or press a function key that displays a different screen. It is easy to fix the problem by changing the display file back (CHGDSPF ... RSTDSP(*YES)). You must set this attribute correctly in the SVR record, because it can be very expensive for our customers if RSTDSP(*YES) is specified unnecessarily.

## General Batch Considerations

All of the preceding techniques for speeding up your application (except for display file considerations) also apply to batch jobs. Some additional techniques are unique to batch.

### Batch Window

The batch window is usually the off-shift time when most of the employees are home. Customers might want to get all batch jobs done before business opens for the day. This could be 6:00 p.m. to 7:00 a.m., or a shorter time. One of the ways to help the customer get through the batch work is to design for multi-threading. Try to design our batch jobs so that the customer can run two or more of them at once. Removing unnecessary dependencies allows multi-threading. (For example, you may need to have a copy of a data area in QTEMP instead of JDFOBJ if changes to that data area only affect this job).

Consider using SORT to speed up batch processing. The discussion on Sequential I/O describes how to use blocking. To see a benefit from blocking sequential input, the data must be physically on the disk in the sequence reading it. SORT would be appropriate for a work file that is, for example, created by this batch job, and that does not have many logical files over it.

If you are creating and updating summary records in a database file as part of the batch run, consider processing the input file in the correct sequence to do level break logic. For example, you want to create a summary record by salesperson in a territory. If you read the input records in that same sequence, you can accumulate totals for the salesperson until the salesperson changes. Write out the summary record for that salesperson, and so on. If you process the data in a different sequence, you have to repetitively retrieve and update the summary record for the salesperson. See *Database Read/Write*.

Look for opportunities to combine job steps to reduce passes through the data. If there are two reports that read the data in the same sequence, you could combine them into one program. Read the data once to produce both reports.

### Logical Files

In general, you should not set up a logical file solely to accommodate batch requirements. Use DREAM Writer or OPNQRYF to sequence and select the data for your program (or use SORT or RGZPFM).

### Sort

The CL command to run a sort is:

```
FMTDTA INFILE ((F0311))
OUTFILE (F0311WRK)
SRCFILE (QFMTSRC)
SRCMBR (GENMBR)
```

Sort can both select and sequence data for you. You need SORT control statements in a source member before you run the SORT. If these statements do not change, type them into a source member. You can prompt to get the format when you are editing the source member by using prompt types RH, RR, RF, or RC (for example, type IPRH in the sequence number field). If the selection/sequence varies, your program can write out the correct SORT statements. SORT can sort in place (the INFILE and OUTFILE can have the same name).

The following are some of the SORT control statements created by P062904. The first line is the header (prompt type RH). It defines the total length of sort fields, whether the sort is ascending or descending, and whether the sort fields should be included in the output record (an X means leave them out).

```
HFILE       88A         X
```

The next group of statements defines the sort fields (prompt type RF). The N in the second position means that it is a sort control field. It must be sorted according to the Header statement. An O specifies a sort control field to be sorted OPPOSITE to the Header statement. This way you can have some fields that sort in an ascending way and some that sort in a descending way. The third position defines the field type (character, packed, or zoned). The start and end positions are the start and end positions of this field in the input record.

```
FNC     3     7     CO
FNC    10    21    MCU
FNC    22    27    OBJ
```

```
FNC    28    35    SUB
FNU    61    S2     FY
FNU    S3    64     PN
FNU    65    70    DGJ
FNP   128   130   PDBA
FNU    71    78    AN8
```

The last statement also is a field definition (prompt type RF). It redefines the entire input record as one character field and specifies it must be written to the output file (the D in position two means that it is a data field).

```
FDC   1  164                    * *  OUTPUT COMPLETE RECORD  * *
```

### Reorganize

The CL command to reorganize a file is:

```
RGZPFM FILE (F0311)
KEYFILE (*LIBL/F0311LA F0311LA)
```

Reorganizing requires an existing access path over the data in the sequence you require. Reorganizing does not select data for you.

Be wary of using MAINT(*REBLD) or MAINT(*DLY). With MAINT(*REBLD), the entire access path will be rebuilt every time. It can slow down your testing. With MAINT(*DLY), the system monitors the percentage of records changed and dynamically changes the file to MAINT(*REBLD) if you change more than 20%. Consider removing the logical file member instead of MAINT(*REBLD). It is more efficient.

```
RMVM FILE (F0101LA)  MBR (F0101LA)
CALL  UPDPGM
ADDLFM FILE (F0101LA)  MBR (F0101LA)
```

## National Language Support

As of A7.3 release, JD Edwards World complied with IBM's V3R1 National Language Support features. A8.1 users can take advantage of IBM's Character Data Representation Architecture (CDRA), which is not supported by JD Edwards World releases prior to A7.3. This chapter discusses the aspects of CDRA that are relevant

to JD Edwards World software, the JD Edwards World approach to compliance, and guidelines for programmers. For a thorough introduction to CDRA, refer to IBM's AS/400 National Language Support for V3R1.

## CDRA Overview

CDRA is IBM's method of achieving consistent representation, processing, and interchange of coded characters (data) in AS/400 business computing systems and across IBM systems (AS/400 National Language Support, section 2.2). This is accomplished by tagging all character data with a coded character-set identifier (CCSID). Because character data is represented internally by hexadecimal code points, the data's CCSID tells the system how to interpret the hex values to arrive at its correct character representation. Therefore, if data coded to one CCSID is read by a job coded with a different CCSID, the integrity of the character data is preserved as CDRA handles the conversions of hex code points under the covers.

The invariant character set (see *Invariant Character Set* in this chapter) is a special set of characters that are mapped to the same code points in virtually all code pages. Therefore, their hex values never change even if they are transmitted across different CCSIDs. There are references to characters outside of this set as chameleons because their hex code points changes to adapt to different CCSIDs to retain the same graphic character representation. Chameleons common to the US code page are '$', '#', '@', and '!'.

## JD Edwards World Implementation

The JD Edwards World approach to tagging file data is to distinguish between textual data and non-textual data at the field level. Those fields that contain text (marked with an open data type in the data dictionary) are tagged with CCSID 00037, which is the CCSID of our development AS/400. This CCSID is generally used by North American AS/400s. Those character fields that are not used to store descriptive text (marked with an alpha data type in the data dictionary) are tagged with CCSID 65535. This CCSID indicates that the data is to be treated as hexadecimal data rather than coded graphic character data. In this way, the graphic character integrity of textual data is preserved as well as the hexadecimal integrity of non-textual character data.

Initially, CCSID keywords are inserted in the DDS for all non-numeric fields in physical files. The File Design Aid was enhanced to automatically insert these keywords when exiting so that the programmer does not need to be concerned about it. When IBM incorporates the CCSID parameter into the field reference file, the keywords are deleted from the file DDS and inserted into the field reference files instead.

In addition, the create option (14) was changed in the software versions repository to submit the creation with a job CCSID of 00037. This causes all source data for non-ILE programs, and constant data for display and printer files, to be interpreted through a 00037 lens. ILE program source data is interpreted according to the CCSID of the source file. Source files are tagged as 00037. Incidentally, the IBM compilers for non-ILE programs interpret data in source files through a 00037 lens. Most compilers expect syntactical operators and the naming convention for the source code to be in code page 00037; therefore, undesired mapping occurs if the

source is compiled with a CCSID other than 00037 or 65535 (AS/400 National Language Support Planning Guide section 3.4.10).

The display and printer file create commands were changed to include the *JOBCCSID value for the CHRID parameter (default is *DEVD). This preserves character data integrity for display and printer files where the user's job CCSID is different from the user's device description CHRID.

## Programming Guidelines

Follow these guidelines when you modify JD Edwards World objects, or when you create objects by using the JD Edwards World FDA, RDA, SDA, or CASE tool.

- Do not change any file CCSIDs either at the file or the field level
- If you create a source file to use for JD Edwards World programs or with JD Edwards World programming utilities, be sure to tag it with a CCSID of 00037. A source file CCSID defaults to the CCSID of the job that created it
- If you modify a source member through PDM:
    - For physical file DDS, do not change the CCSID keyword values. Otherwise, you can easily destroy data integrity. If you want to add a field, use File Design Aid, which automatically inserts the appropriate CCSID attribute for you
    - When you create an object through PDM, be sure your job CCSID is 00037. A user's job CCSID defaults to the user profile CCSID. To change it, enter CHGJOB CCSID(00037)
    - When you create a display file or printer file through PDM, be sure the CHRID keyword has value *JOBCCSID. This is not the default value
- When you create interactive programs, never require the user to enter a chameleon character to perform a particular function. Because the program looks for a specific hex value, the graphic character to enter varies according to the user's job CCSID. Therefore, this feature cannot be documented. Hints as to the location of any existing cases can be found in vocabulary overrides and processing options. (Example to avoid: "Enter '@' to view all types")
- Do not embed data dictionary item names in text (for example, glossary and processing option text) if they contain chameleon characters. (Example to avoid: "Default value from #CYR in data dictionary")
- Do not assign chameleon characters to data dictionary default or allowed values, or to processing options default values
- Do not use chameleons as or within literals in source code. (Example to avoid: FLDA IFEQ '$AB')

## Invariant Character Set

The following characters are included in the invariant character set:

- 26 unaccented uppercase letters 'A' through 'Z'
- 26 unaccented lowercase letters 'a' through 'z'

- Ten digits '0' through '9'
- Plus sign
- Less-than sign
- Equal sign
- Greater-than sign
- Percent sign
- Ampersand
- Asterisk
- Straight double quote
- Straight single quote
- Left parenthesis
- Right parenthesis
- Comma
- Underscore
- Hyphen
- Period
- Slash right
- Colon
- Semicolon
- Question mark

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| A | B | C | D | E | F | G | H | I |
| J | K | L | M | N | O | P | Q | R |
| S | T | U | V | W | X | Y | Z | a |
| b | c | d | e | f | g | h | i | j |
| k | l | m | n | o | p | q | r | s |
| t | u | v | w | x | y | z | 0 | 1 |
| 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | + |
| < | = | > | % | & | * | “ | ‘ | ( |
| ) | , | _ | - | . | / | : | ; | ? |

## DREAM Writer Printer Overrides

Be sure to run the Convert CL *LDA Lengths program (P98LDACVT) to include the double-byte parameters when overriding print files in DREAM Writer. For more

information, see *DREAM Writer Printer Overrides* in the *Double-Byte Enablement* chapter.

# Double-Byte Enablement

For A6 and later releases, all fields that can contain text material were redefined as open fields by changing the field-type attribute to "O" in the data dictionary. Technically, a field defined as open, allows the entry of single-byte data, double-byte data, or a combination of both. Whenever a string of double-byte characters is entered into a text field, the double-byte character string must be bracketed with special characters. These special characters, called shift-out and shift-in characters, indicate where the double-byte character string begins and ends. Without these delimiting characters, the system cannot resolve the correct meaning of the bits and bytes contained in a character string. An AS/400 system error results. Program modifications must be made to process all field truncations of open fields through a special subroutine (C9822).

## C9822 Subroutine

The C9822 subroutine oversees the process of truncating a field that contains double-byte data, making sure that a double-byte character is not divided in half or that one of the special shift-out or shift-in characters is not cut off. If the subroutine detects such a condition, the truncated character string is adjusted to contain the last whole double-byte character, and to insert a shift-in character.

### Affected Programs

You must run the C9822 subroutine. You must check all programs for any truncated open fields. If field truncations occur, you must edit them by using C9822.

## Procedure

**1.** Insert copy module for E9822.

```
E*************************************************************
E*
E*     Copy Composite Member for Subroutine - C9822 Common
E*
E/COPY JDECPY,E9822
E*************************************************************
```

**2.** Insert copy module for C9822.

```
C*********************************************************** C*
C*   Copy Common Subroutine - Double Byte Truncation Routine.
C*
C/COPY JDECPY,C9822
C***********************************************************
```

**3.** Modify field MOVEs for all fields being truncated as displayed below:

```
H/TITLE PMODELDBCS - MODEL - Execute Text Truncation
H*
C*
C*    Move Field xxx to Field zzz for Output. Execute C9822
C*
CSR   CLEAROUA
CSR   MOVEA xxx      @UA
CSR   Z-ADD yyy      #OUTLG
CSR   EXSR C9822
C*
CSR   MOVEAOUB    zzz   C*
C*
```

Explanation of variables:

| | | |
|---|---|---|
| @UA | = | Input processing array |
| @UB | = | Output processing array |
| xxx | = | Field name that contains text to be truncated |
| yyy | = | Length of output field, (numeric value) |
| zzz | = | Display field |

## DREAM Writer Printer Overrides

Additional fields were added to DREAM Writer printer overrides for double-byte support. Use them only when you operate the software on a double-byte machine.

### Implementation

The command string for printer overrides, which is built by DREAM Writer, is returned to CL programs by way of the local data area (*LDA) starting at position 257. The CL program retrieves the command string, then calls the command processor to run it. The additional double-byte parameters are returned in the last

256 bytes of the local data area, which increases the length of the printer file override string from 512 bytes to 768 bytes. To include them, the CL programs must be modified to retrieve the full length of the command string. Program P98LDACVT is provided to modify the CL programs that retrieve the local data area for printer overrides. It changes the si2e of the CL variable &PRTOVR from 512 bytes to 768 bytes, and retrieves the full length of the command string.

Modified declaration statements:

```
DCL   VAR(&PRTOVR) TYPE(*CHAR) LEN(76 8)
DCL   VAR(&OVRCLNG) TYPE(*DEC) LEN(15 5) VALUE(768.00000)
```

Modified statement to retrieve the full length of the command from the local data area:

```
CHGVAR      VAR(&PRTOVR) VALUE(%SST(*LDA 2 57 768))
```

Run this conversion only if your DREAM Writers require the additional printer overrides for double-byte capability.

### Running the Program

To run the program, select Convert CL *LDA Lengths from the A8 Conversion Utilities menu (BA6).

# Appendix

## Programs Converted to RPG IV

Double-click the following icon to open the A9.1 RPGL Programs.

## Obsolete files and other source members and objects

Double-click the following icon to open the Obsolete SVR Members and Objects.