ORACLE®

JD EDWARDS WORLD

JD Edwards World Programmer's Guide

Version A8.1 Base to A9.1

Revised - September 5, 2007

JD Edwards World

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 - September 5, 2007

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 The intended audience are 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 A7.3 release to the A9.1 release, use the *A73 to A91 Programmer's Guide*.

Note: The file changes for A8.1 base to A8.1 Cumulative Update 06 may or may not apply for you depending on what Cumulative Update level you are upgrading from.

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 to 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.

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  2 0  (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

N / A.

Copy Member Changes (non-Database)

Rptg Code	Copy Member	Function	Description	Action
73	C73012L	CPYL	/COPY Multi-Intra State Taxes	New
82	C82IDF	COPY	Query File ID Conversions	New

Rptg Code	Copy Member	Function	Description	Action
82	C82PARM	COPY	Query Parms for RPG Programs	New
82	E82FLDS	COPY	Query X82FLDS Definitions	New
82	E82PARM	COPY	Query Parms for RPG Programs	New
92	C92CAD	COPY	CAD - SDA / RDA Copy Module	New
92	C92CAD2	COPY	CAD - SDA / RDA Copy Module	New
92	C92CAD2PLI	COPY	CAD - SDA / RDA Copy Module	New
92	C92CAD3	COPY	CAD - SDA / RDA Copy Module	New
92	C92CAD3PLI	COPY	CAD - SDA / RDA Copy Module	New
92	C92CAD4	COPY	CAD - SDA / RDA Copy Module	New
92	C92CAD5	COPY	CAD - SDA / RDA Copy Module	New
92	C92CAD6	COPY	CAD - SDA / RDA Copy Module	New
92	C92CADPLI	COPY	CAD - SDA / RDA Copy Module	New
98	C0001T	COPY	Edit Action Code - Including PC Import / Export	New
98	C0001TL	CPYL	Edit Action Code - Including PC Import / Export	New
98	C0011L	CPYL	Center Descriptive Titles	New
98	C0016L	CPYL	Format Numeric Fields for Output with Overrides	New
98	C0040LE	CPYL	Alphanumeric Left Justify and Compress (ILE)	New
98	E0040LE	COPY	Alphanumeric Left Justify and Compress (ILE)	New
98	C0040XL	CPYL	Alphanumeric Left Justify and Compress - Language Support	New
98	C0042L	CPYL	Right Adjust Alphanumeric Field	New
98	C00E1	COPY	Setup Interactive Export	New
98	C00E1L	CPYL	Setup Interactive Export	New
98	C00E2	COPY	Perform Interactive Export	New
98	C00E2L	CPYL	Perform Interactive Export	New
98	C00EXWL	CPYL	Window Position Determination	New

Rptg Code	Copy Member	Function	Description	Action
98	C00I1	COPY	Import Data from CSV File	New
98	C00I1L	CPYL	Import Data from CSV File	New
98	C00I2	COPY	Import Data for subfiles	New
98	C00I2L	CPYL	Import Data for subfiles	New
98	C00IEM	COPY	Import/Export Messages	New
98	C00IEML	CPYL	Import/Export Messages	New
98	C00IESTS	COPY	Import/Export Status - Interactive	New
98	C00IET	COPY	Import/Export Termination	New
98	C00IETL	CPYL	Import/Export Termination	New
98	C00IEXP	COPY	Export Data to CSV File - Interactive	New
98	C00IIMP	COPY	Import Data from CSV File - Interactive	New
98	C00IMPSF	COPY	Import Data for Subfile - Interactive	New
98	C74S347	COPY	DREAM Writer - Dynamic Report Processing	New
98	C74SFL	COPY	Format Fields left or right with blanks or no blanks	New
98	C81DRPT2	COPY	DREAM Writer - Dynamic Report Processing	New
98	C81DRPTL	CPYL	DREAM Writer - Dynamic Report Processing	New
98	C98208	COPY	Acquire a Transaction eSignature	New
98	C98208L	CPYL	Acquire a Transaction eSignature	New
98	C98209	COPY	Release a Transaction eSignature	New
98	C98209L	CPYL	Release a Transaction eSignature	New
98	E0001	COPY	Edit Action Code	Changed
98	E0001L	CPYL	Edit Action Code	Changed
98	E0011L	CPYL	Center Descriptive Titles	New
98	E0016L	CPYL	Format Numeric Fields for Output with Overrides	New
98	E0040L	CPYL	Alphanumeric Left Justify-Compress Blanks to 1st Character	Changed

Rptg Code	Copy Member	Function	Description	Action
98	E0042L	CPYL	Right Adjust Alphanumeric Field	New
98	E74SFL	COPY	Format Fields left or right with blanks or no blanks	New
98	E81DRPT2	COPY	DREAM Writer - Dynamic Report Processing	New
98	E81DRPTL	CPYL	DREAM Writer - Dynamic Report Processing	New
98	E9822L	CPYL	Double Byte Truncation Routine	Changed

Server Changes

Rpt Cde	Member ID	Func	Description	Action
81	X81DRPTD	RPG	DREAM Writer - Dynamic Report Description Retrieval	Changed
81	X81OPT	RPG	DREAM Writer - Retrieve Processing Options	Changed
81	X98790	RPGL	DREAM Writer - Copy / Delete Server	Changed
82	X82116	RPGL	Query Group Copy server copied from P82116	Changed
96	X0001M	RPGL	Encoding Server	Changed
96	X96CCV1	RPG	Cursor Control Value Processor	Changed
96	X96CCV2	RPG	Cursor Control Value Processor	Changed
96	X96CCV3	RPG	Cursor Control Value Processor	Changed
96	X96CCV4	RPG	Cursor Control Value Processor	Changed
96	X96CCV5	RPG	Cursor Control Value Processor	Changed
98	X0005	RPG	User-defined Codes Server	Changed
98	X0010	RPGL	Automatic Next Numbering	Changed
98	X0020CL	RPG	Load Soft Coding - CL Programs (Maximum 50 VTX Fields)	Changed
98	X0040L	RPGL	Alphanumeric Left Justify and Compress - Language support	Changed
98	X0090	RPG	Retrieve Dynamic Totaling Description	Changed

Rpt Cde	Member ID	Func	Description	Action
98	X98202	RPGL	Status code server	Changed
98	X98204	RPGL	Reason code server for trigger programs	Changed
98	X98204S	RPGL	Next serial number server	Changed
98	X98211	RPGL	Retrieve DBF active triggers	Changed
98	X98600W	RPG	Delete Member Verification	Changed
98	X98LSTFLD	RPGL	List file fields to JDE outfile	Changed
98	X98LSTOBJ	RPGL	List objects to JDE outfile	Changed
98	XBASETRG	RPGL	Trigger Program Template/w eSignature	Changed
98	XF99624	RPG	Server - Validate Systems	Changed
98	X0001	RPGL	Action Code Security Server	New
98	X98CVTHEX	RPGL	Convert char to hex or hex to char	New
98	X98EDTHLP	RPGL	Help Text - Edit Program ID	New
98	X0016A	RPGL	Generic Text Trigger Program (F01131)	Obsoleted
98	X0016B	RPGL	Generic Text Trigger Program (F00164)	Obsoleted
98	X0016C	RPGL	Generic Text Trigger Program (F00163)	Obsoleted
98	X0016D	RPGL	Generic Text Trigger Program (F00165)	Obsoleted
98	X0016E	RPGL	Generic Text Trigger Program (F4802H)	Obsoleted
98	X0016X	C	Strip RTF characters from string	Obsoleted
98	X00692	RPGL	Business Units Supp. Data Trigger (F00692)	Obsoleted
98	X01092	RPGL	Address Book Supp. Data Trigger (F01092)	Obsoleted
98	X08092	RPGL	HRM Supp. Data Trigger (F08092)	Obsoleted
98	X12092	RPGL	Asset Management Supp. Data Trigger (F12092)	Obsoleted
98	X41092	RPGL	Inventory Supp. Data Trigger (F41092)	Obsoleted
98	X48092	RPGL	Work Orders/ECO Supp. Data Trigger (F48092)	Obsoleted

Technical Foundations – Enhancements

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

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.

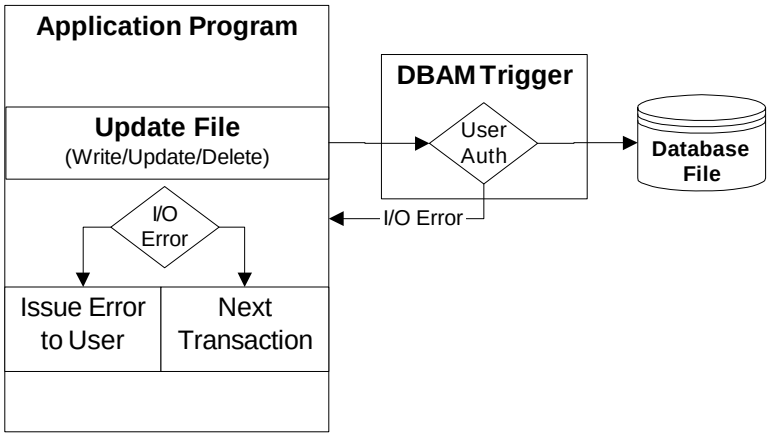
Implementation

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.

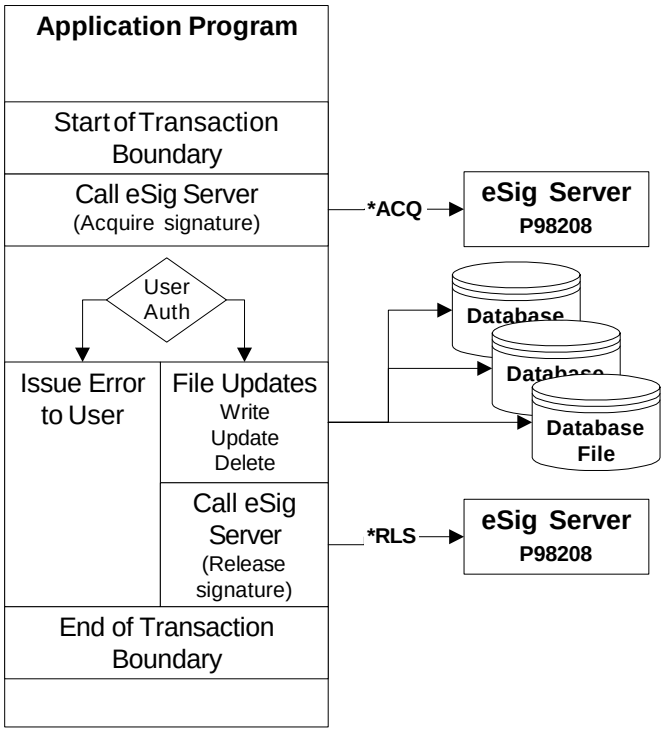


Transaction Level

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/Oand 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

Rptg Code	Physical File	Dependent File	Description
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
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

Rptg Code	Physical File	Dependent File	Description
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
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

Rptg Code	Physical File	Dependent File	Description
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
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

Rptg Code	Physical File	Dependent File	Description
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
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

Rptg Code	Member ID	Description	Function
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
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

Rptg Code	Member ID	Description	Function
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	Function	Description	Action
01	E01100L	CPYL	Retrieve Mailing Address	New
04	C73013L	CPYL	/COPY Perform Tax Calculations - Vertex Mode	New
04	C73GEOL	CPYL	/COPY Retrieve GeoCode - ILE	New

Server Changes

Rpt Cde	Member ID	Func	Description	Action
00	X00500CA	RPG	Convert Numeric to Words - MUST SEND SOURCE - Canadian	Changed
00	X00500FRC	RPG	Convert Numeric to Words - MUST SEND SOURCE - Canadian (84)	Changed
00	X00711	RPGL	Process Definition Server	New
00	X00712	RPG	Process Conflicts Definition Server	New
00	X0096	RPG	Batch Header Retrieve/Update/Create	Changed
00	X0097	RPG	Batch Control - Prompt Batch Control and Assign Number	Changed
00	X0099	RPG	Batch Control - Update	Changed
00	X00991	RPG	Create user objects for functional servers	Changed
00	X00PPAT1	RPG	E-Mail Message Server	Changed
00	X00PPAT2	RPG	E-Mail Message Server - Handles V4R3 Internet Mail	Changed
00	XF0030	RPG	Input/Output Server - F0030	New
00	XS0009	RPG	File Server - F0009 (Last Record Cache)	Changed
00	XS04572	RPG	File Server - F04572	Changed
00	XT0911Z1	RPG	Journal Entry Functional Server	Changed
01	X0030	RPG	Retrieve Next Number	Changed
01	X0101	RPG	Retrieve Address Number	Changed
01	X0101A	RPG	Audit Log Information	Changed

Rpt Cde	Member ID	Func	Description	Action
01	X0103	RPGL	Search Type / Action Code Security Analysis	New
01	X01100	RPG	Retrieve Mailing Address	Changed
01	X011101	RPG	Convert Full Alpha Name to First, Last, Middle Name	Changed
01	X0115	RPG	Phone Numbers - Retrieve Next Number	Changed
01	X01200	RPG	Address Book Server - Parent / Child Processing	Changed
01	X01850	RPG	Record Existence Check - Address Book	Changed
01	X01950	RPG	Address Book Delete Server	Changed
01	X0301	RPG	Customer Currency Amount Server	Changed
01	X0401	RPG	Vendor Currency Amount Server	Changed
01	X73014	RPG	Vertex - Update Changed GeoCode	Changed
01	XF0075	RPGL	Input / Output Server - F0075	New
01	XF0101	RPG	Input / Output Server - F0101	Changed
01	XF01014	RPGL	Input / Output Server - F01014	New
01	XF01017	RPGL	Input / Output Server - F01017	New
01	XF01018	RPGL	Input / Output Server - F01018	New
01	XF0111	RPG	Input / Output Server - F0111	Changed
01	XF0115	RPG	Input / Output Server - F0115	Changed
01	XF0116	RPG	Input / Output Server - F0116	New
01	XF0301	RPG	Input / Output Server - F0301	Changed
01	XF03015	RPGL	Input / Output Server - F03015	New
01	XF0401	RPG	Input / Output Server - F0401	Changed
01	XF04015	RPGL	Input / Output Server - F04015	New
01	XS0030	RPGL	File Server - F0030	New
01	XS0101	RPG	File Server - Address Book (F0101)	Changed
01	XS01016	RPGL	File Server - F01016	New
01	XS01017	RPGL	File Server - F01017 - Related Addresses	New
01	XS0101LA	RPG	File Server - F0101LA	Changed

Rpt Cde	Member ID	Func	Description	Action
01	XS0111	RPG	File Server - Who's Who (F0111)	Changed
01	XS03015	RPGL	File Server - F03015	New
01	XS04015	RPGL	File Server - F04015	New
01	XS4574	RPGL	File Server - F04574 - Return Spot Rate	New
01	XT0101Z1	RPG	Address Book Functional Server	Changed
01	XT0150	RPG	Address Book Organizational Structure Functional Server	Changed
03	X0302	RPG	Compute Due Date	Changed
03	X03021	RPG	Compute Due Date For Advanced Payment Terms	Changed
03	X0320	RPG	A/R Drafts - Company Level Processing	Changed
03	X03220	RPG	Convert Existing Draft to Draft Handling Files (F0320,1,2)	Changed
03	X035001	RPG	Draft Number Server - Statements	Changed
03	X03801	RPG	User Exit Interface Program	Changed
03	X03900	RPG	Summarize A/R User Space	Changed
03	X73031	RPG	Vertex - Link/Parm Area Server for A/R	Changed
03	XF0311	RPG	Input/Output Server - F0311	Changed
03	XT0311Z1	RPG	Accounts Receivable Functional Server	Changed
04	X0030I	RPG	International Bank Account Number Validation (IBAN)	Changed
04	X04111	RPG	A/P Ledger - Retrieve Next Pay Item Extension Number	Changed
04	X04131	RPG	A/P Matching Documents - Retrieve Next Payment ID	Changed
04	X0415	RPG	Build OPNQRYF Command for A/P Payment Processing	Changed
04	X042572	RPG	A/P Processing - Split Detail	Changed
04	X0430	RPG	A/P Payment - Retrieve/Update Bank Account Information	Changed
04	X0450	RPG	Payee Control Server	Changed
04	X0453	RPG	A/P Approval - Set Approval Flag for Voucher	Changed

Rpt Cde	Member ID	Func	Description	Action
04	X04570	RPG	A/P Payment - Pre-Payment Processor	Changed
04	X04572	RPG	A/P Payment - Retrieve Next Payment Control Number	Changed
04	X04801	RPG	User Exit Interface Program	Changed
04	X04900	RPG	Summarize A/P User Space	Changed
04	X09860	RPG	Suspended Tax - Reverse Entries	Changed
04	X73023	RPG	Vertex Tax - Link Parm Server for A/P and A/R	Changed
04	X7303	RPG	Vertex - Link/Parm Area Server	Changed
04	X73033	RPG	Vertex - Link/Parm Server for A/P	Changed
04	XT0411Z1	RPG	Accounts Payable Functional Server	Changed
09	X0903111	RPG	Functional Server - Update F0911 with corresponding record	Changed
09	X0904111	RPG	Functional Server - Update F0911 with corresponding record	Changed
09	X0909	RPG	Subledger Edit Module	Changed
09	X091103	RPG	Journal Entry Tax Server	Changed
11	X005ES	RPG	Convert Numeric Value to Spanish Words - MUST SEND SOURCE	New
11	X005FRC	RPG	Convert Numeric to French Canadian Words - MUST SEND SOURCE	Changed
12	X1200	RPG	Fixed Assets Date Sensitive Balance Retrieval	Changed
44H	X44503	RPG	Display JE From Model JE	Changed
44H	X445098	RPG	Takeoff Substitute Quantity	Changed
44H	X44731	RPG	JA Revenue Budget functional server	Changed
48S	X4809	RPG	Table Key Editing - Service Billing	Changed
48S	X48096	RPG	Cost Plus Mark Up Table File Server	Changed
48S	X48121	RPG	Component Work File Generation	Changed
51	X51013	RPG	Job Cost Projection Routine	Changed
51	XF51081	RPGL	Input/Output Server - F51081	New
52	X52NTE	RPG	Contract Billing Not-To-Exceed Server	Changed

Cross Industry Changes and Enhancements

This chapter describes changes made to Cross Industry systems.

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.

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 usually edits 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 you 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, 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 populates 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 the ones just mentioned) 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 users are not required 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 you 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, 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	Function	Description	Action
30	C3007L	CPYL	Back/Forward Schedule Operation Sequence Numbers	New
31	C3101	COPY	Common Routine for AAI retrieval & Account validation	Changed
31	C3104	COPY	Common Routine for Work Order Completion	Changed
31	C3105	COPY	Common Routine for Work Order Hour and Quantity Posting	Changed
32	C3294	COPY	Configured Item Segment Editing	Changed
37	C371181	COPY	File Server Key Lists - XF371181	Changed
40	C0043	COPY	Scan and Replace Character	New
40	C4102A	COPY	Common Subroutine - Calculate Available Quantity	Changed
40	C4102AL	CPYL	Common Subroutine - Calculate Available Quantity (ILE)	Changed
40	C73020	COPY	/COPY Load Vertex Sales Tax Parameters	Changed
40	C73020L	CPYL	/COPY Load Vertex Sales Tax Parameters	New
40	E0043	COPY	Scan and Replace Character	New
41	C410871	COPY	File Server Key List - XF4108	Changed
41B	C41DECL	CPYL	Determine Decimal Position for Temperature/Density	New
42	C005AL	CPYL	Common Subroutine for ECS functions (ILE)	New
42	C00VLL	CPYL	Common Subroutine - Retrieve Data Dictionary Defaults(P4211)	New
42	C4211KYL	CPYL	Key List - P4211 (ILE)	New
42	C4219	COPY	Common Subroutine F4211 to F42199	Changed
42	C4219L	CPYL	Common Subroutine F4211 to F42199 (ILE)	New
42	C998AL	CPYL	Common Subroutine - Retrieve Data Dictionary Defaults(P4211)	New

Rptg Code	Copy Member	Function	Description	Action
43	C43291L	CPYL	Reverse Foreign Landed Costs	New
43	C99843L	CPYL	Common Subroutine - Retrieve DD Defaults (P4312)	New
43	CWRTHISM	COPY	Write Receipt History - Match time	New
43	CWRTHISRL	CPYL	Write Receipt History - Receipt time	New

Server Changes

Rpt Cde	Member ID	Func	Description	Action
13	X13807	RPG	Parts List and Routing Copy	Changed
31	X3111	RPG	Issued Serial Numbers Editing	Changed
31	X3112	RPG	Create Configured Work Order Routings	Changed
31	X3112C	RPG	Cost Configured Routings	Changed
31	XT310911	RPG	Schedule Quantity Detail Functional Server	Changed
32	X32351	RPG	Configured Item Split Server	Changed
32	X3294	RPG	Sales Order Detail Server	Changed
32	X4201WO	RPG	Sales Order Entry - Work Order Processing	Changed
34A	X41021T	RPG	Inventory Balance OrderManageNotify trigger (F41021)	Obsoleted
34A	X4301T	RPG	Purchase Order Header OrderManageNotify trigger (F4301)	Obsoleted
34A	X4311T	RPG	Purchase Order Detail OrderManageNotify trigger (F4311)	Obsoleted
34A	X4801T	RPG	Work Order OrderManageNotify trigger (F4801)	Obsoleted
34B	X0401T	RPG	Supplier Master Trigger (F0401)	Obsoleted
34B	X0901T	RPG	Account Master Trigger (F0901)	Obsoleted
34B	X41001T	RPG	Inventory Constants Trigger (F41001)	Obsoleted
34B	X41002T	RPG	UOM Conversions Trigger (F41002)	Obsoleted
34B	X4101T	RPG	Item Master Trigger (F4101)	Obsoleted

Rpt Cde	Member ID	Func	Description	Action
34B	X4105T	RPG	Item Cost Trigger (F4105)	Obsoleted
34C	X0101T	RPG	Address Book Master trigger (F0101)	Obsoleted
34C	X0111T	RPG	Address Book - Who's Who trigger (F0111)	Obsoleted
34C	X0115T	RPG	Address Book - Contact Phone Numbers trigger (F0115)	Obsoleted
34C	X0116T	RPG	Address by Date trigger (F0116)	Obsoleted
34C	X0301T	RPG	Customer Master trigger (F0301)	Obsoleted
34C	X4201T	RPG	Sales Order Header OrderManageNotify trigger (F4201)	Obsoleted
37	XF3711	RPG	Input/Output Server - F3711	Changed
38	X3811	RPG	Agreement Search/Edit Server	Changed
38	XT38111	RPG	Agreement Transaction Server	Changed
39	X391202	RPG	Period Extraction - In transit Server	Changed
39	X391203	RPG	Period Extraction - Not-In-Stock Server	Changed
39	X391204	RPG	Period Extraction - Accommodations Server	Changed
39	X391214	RPG	Period Extraction - Replacement/Current Cost Calculations	Changed
40	X4001	RPG	Calculate Economic Order Quantity	Changed
40	X4006	RPG	Retrieve Order Address Information	Changed
40	X40216	RPG	Blanket Release Maintenance	Changed
40	X4040	RPG	Calculate Promised, Delivery and Pick Date	Changed
40	X4090	RPG	Retrieve Distribution/Manufacturing AAI Information	Changed
40	X4092	RPG	Load Group Codes	Changed
40	X40HDR	RPG	Load Header Fields Which Default To Detail File	Changed
40	X4106	RPG	Retrieve Base Price Server - F4106	Changed
40	X41LOCN	RPG	Location Scrub & Format	Changed
40	X41LOTC	RPGL	Calculate lot dates	New
40	X49080	RPG	Invoice Cycle Calculation of Scheduled Invoice Date	Changed

Rpt Cde	Member ID	Func	Description	Action
40	X73DISP	RPG	HAZOX (R) MSDS Dispatch server	Changed
40	XF40300	RPG	Input/Output Server - Preference Profiles	Changed
40	XF40302	RPG	Input/Output Server - Preference Profiles	Changed
40	XF41021	RPG	Input/Output Server - F41021	Changed
40	XF4108	RPG	Input/Output Server - F4108	Changed
41	X41002	RPG	Convert Unit of Measure w/Item Number	Changed
41	X4102DEF	RPG	Load Default Values - Item Branch Add	Changed
41	X4108	RPG	Lot Master Update	Changed
41	X4111	RPG	Write to Item Ledger File (F4111)	Changed
41	X41255	RPGL	Calc Life Remaining - Server pgm	New
41	XF4101	RPGL	Input/Output Server - F4101	New
41	XF4111	RPG	Input/Output Server - F4111	Changed
41	XT4102Z1	RPG	Item Balance Functional Server	Changed
42	X09901	RPG	Summarize G/L User Space - Overflow	New
42	X4096	RPG	Sales Flex Accounting Functional Server	Changed
42	X41351	RPG	Kit Server	Changed
42	X42119	RPG	Write to Flexible Sales History File (F42119)	Changed
42	X4211CFG	RPG	Sales Order Entry - Configured Items	Changed
42	X4211D	RPGL	Sales Order Cancel	Changed
42	X4211FG	RPG	Free Goods Processing	Changed
42	X4211KT	RPG	Sales Order Entry - Kit Processing	Changed
42	X4211PR	RPG	Sales Order Entry - Special Pricing	Changed
42	X4211TFR	RPG	Sales Order Entry - Interbranch Transfer Cost	Changed
42	X4228	RPG	Sales Order Credit Check Processing	Changed
42	X4239	RPGL	Validate Last Customer Shipment	New
42	XF42119	RPG	Input/Output Server - F42119	Changed
42	XF42199	RPG	Input/Output Server - F42199	Changed
42	XF4239	RPGL	Input/Output Server - F4239	New

Rpt Cde	Member ID	Func	Description	Action
42	XF42400	RPGL	Input/Output Server - F42400	New
43	X00COM	RPG	Update Commitment Ledger	Changed
43	X41061	RPG	Retrieve Price Information	Changed
43	X4250	RPG	Update Related Purchase Order	Changed
43	X43008	RPG	Approval Message Sender	Changed
43	X43091	RPG	Receipt Routing Initiator	Changed
43	X43092D	RPG	Receipt Routing Disposition Server	Changed
43	X43092M	RPG	Receipt Routing Movement Server	Changed
43	X4311PR	RPG	Purchase Order Entry - Special Pricing	Changed
43	X4311SO	RPG	Update the Sales Order Related to a Purchase Order	Changed
43	X4312SD	RPG	Load and Confirm Sales Order from Receipts	New
43	X4313	RPG	Edit and Retrieve Work File T4313	New
43	X43230	RPG	Supplier Analysis - Calculations	Changed
43	X4340	RPG	Purchase Rebate Server	Changed
43	X73022	RPG	Retrieve Supplier/Address Book Info - Vertex Link/Parm Area	Changed
45	X4070	RPG	Calculate Price Adjustment Server	Changed
45	X4074PA	RPG	Detached Adjustment F4211 Data Structure Server	Changed
45	X4076	RPG	Price Formula Server	Changed
45	X4211PA	RPG	Price Adjustment Server - Sales Order Detail	Changed
46	X46002	RPG	Convert to Unit of Measure Structure	Changed
46	X4602	RPG	Location Detail Update	Changed
46	X46140R	RPG	Reverse Receipt/Request	Changed
46	X46141	RPG	Select Putaway Locations	Changed
46	X46142	RPG	Location Validation	Changed
46	X46151	RPG	Select Picking Locations	Changed
46	X46161	RPG	Select Replenishment Locations	Changed
46	X4617	RPG	Confirmation Server	Changed

Rpt Cde	Member ID	Func	Description	Action
46	X46170	RPG	Create Requests	Changed
46	X46175	RPG	Create Suggestions	Changed
46	X46290	RPG	Location Availability / Utilization	Changed
46	X46450	RPG	Carton Recommendations	Changed
48	X4826	RPG	Work Order Activity Rules	Changed
49	X49002	RPG	Convert Unit of Measure w / Volume & Weight	Changed
49	X49580A	RPG	Document Print Control Batch Creation	Changed
49	X49750	RPG	Freight Server	Changed
49	XT4914	RPG	Delivery Document Set Server	Changed
49	XT49799	RPG	Load and Delivery Transaction Server	Changed
49	XT4999	RPG	Order Line Adjustments	Changed
D1D	X680097	RPG	WorldRF - Batch Cntrl - Prompt Cntrl / Assign Number	Obsoleted
D1D	X684617	RPG	WorldRF - Confirmation Server	Obsoleted
D1D	XT680911Z1	RPG	WorldRF - Journal Entry Functional Server	Obsoleted
D1D	XT684102Z1	RPG	WorldRF - Item Balance Functional Server	Obsoleted
D1D	XUI68DATC	CLP	WorldRF - Add days to date	Obsoleted
D1D	XUI68ITM	RPG	WorldRF - Item Number Conversion Routine	Obsoleted
D1D	XUI68RCTL	CLP	WorldRF - Run Control Functional Server	Obsoleted
D1D	XUIRFCHK	CLP	WorldRF - Check for Object Existence	Obsoleted
D1D	XUIRFDUP	RPG	WorldRF - Duplicate Records	Obsoleted
D1D	XUIRFTXT	RPG	WorldRF - Format Text to Specified Length	Obsoleted

Distribution and Manufacturing Changes

This chapter describes changes made to Distribution and Manufacturing systems.

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 on-line program help of X41LOTC.

New Fields

Item Master (F4101)

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

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.

P1SHAN	8	Ship To	Short address number in alphanumeric.
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 AsOf File (F41112), creating the new Inventory Summary File (F41118). If the AsOf generation is run as well as this one, the Inventory Transactions can be purged. If the AsOf 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 (is always ‘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 is stored here.

WIKCOO	Order Company	5	Order Company from the original F4211 record.
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	
And so on			
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

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

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.

Copy Member Changes (non-Database)

Rptg Code	Copy Member	Function	Description	Action
07	C06106	COPY	Load EE labor distribution Instruction to Data Structure	Changed
07	E06106	COPY	Copy Module for F06106 - Array	Changed
08	C0803	COPY	Calculate Salary and Hourly Rate	Changed

Server Changes

Rpt Cde	Member ID	Func	Description	Action
07	X0600	RPG	Address Number Validation	Changed
07	X06391	RPG	Calculate Rollover Balances	Changed
07	X06DYNAC	RPG	Dynamic Account Creation - Custom Server - ESS	Obsoleted
07	X06SSN	RPG	Social Security Number Change Program	Changed
07	XT06116Z1	RPG	Payroll Batch Server	Changed
08	X08301	RPG	Calculate Plan Amounts	Changed
08	X08301SS	RPG	Calculate Plan Amounts - Employee Self Service	Changed
08	X08303SS	RPG	Flexible Spending Accounts - Employee Self Service	Changed

Human Resources and Payroll Changes

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

None.

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	Function	Description	Action
70	C700400	COPY	Enable Tax Withholding - USRIDX	New
70	C700401	COPY	A/B Tax Withholding Codes USRIDX	New
70	C700402	COPY	Tax Codes Server	New
70	C700403	COPY	Tax Code Definition Server	New
70	C700404	COPY	Tax Code Concept Server - User Index	New
70	C700405	COPY	Tax Code Precedent Server - User Index	New
70	C700406	COPY	Base Reduction Server - User Index	New
70	C700407	COPY	Tax Withholding Limit Amount Server - User Index	New
70	C700408	COPY	Supplier Reduction Server - User Index	New
70	C700411	COPY	Voucher Tax Withholding Entry Server	New
70	C700412	COPY	Address Book Members - User Index	New
70	C700415	COPY	Payments done to other companies Server - User Index DS	New
70	C700416	COPY	Accumulated Amount Server - User Index	New
70	C700418	COPY	Address Book Category Codes Tag File Server	New
70	C700419	COPY	Payments to other companies Detail Doc. Server	New
70	C700430	COPY	Withholding - Structure of input data	New
70	C700431	COPY	Tax Withholding Calculation Server	New
70	C7004570	COPY	A/P Common Routine User Index Pre-Payment	New
70	C700476	COPY	/COPY for Pre-Payment Processing	New
70	C700477	COPY	/COPY for Pre-Payment Processing	New
70	C700482	COPY	Void payment - Tax Voucher accumulation	New
70	C70DOC	COPY	/COPY for User Index IXTWDOC	New
70	C70ERR	COPY	Errors in Payment group	New

Rptg Code	Copy Member	Function	Description	Action
70	E700476	COPY	/COPY for Pre-Payment Processing	New
74R	C74R0321	COPY	/COPY for User Index of P74R0321	New
74R	C74R0411	COPY	Country Server - Voucher Entry User Index	New
74R	C74R0415	COPY	/COPY for User Index of X76A0415	New
74R	C74R0900	COPY	Correspondence Constants - USRIDX	New
74R	C74R0901	COPY	Correspondence Account set up - USRIDX	New
74R	C74R0902	COPY	Correspondence Method - USRIDX	New
74R	C74R0911C	COPY	Correspondence Detail - Credit - USRIDX	New
74R	C74R0911D	COPY	Correspondence Detail - Debits - USRIDX	New
74R	C74R0911E	COPY	Correspondence Detail - Errors - USRIDX	New
74R	C74R0911H	COPY	Correspondence Detail - Intercompany Accounts by CO - USRIDX	New
74R	C74R0911I	COPY	Correspondence Detail - Intercompany Accounts - USRIDX	New
74R	C74R0911K	COPY	Correspondence Detail - Debits/Credits NN - USRIDX	New
74R	C74R11A	COPY	Petty Cash Desk	New
74R	C74R11B	COPY	Petty Cash Desk	New
74R	C74R11C	COPY	Petty Cash Desk	New
74R	C74R11D	COPY	Petty Cash Desk	New
74R	C76A4115	COPY	Country Server - Recalculate Pay Item Amount	New
76	C7615	COPY	Edit Transaction Nature w/ICMS Tax Substitution Mark-up	Changed
76	C76STT	COPY	/COPY for Brazilian - Algorithm Tax Situation	Changed
76A	C42PRT	COPY	SOP Invoice Print - CSE Print Fields	New
76A	C42PRTECL	CPYL	ECS Invoice Print - CSE Print Fields	New
76A	C76A0110I	COPY	/COPY for User Index of X76A0110	New

Rptg Code	Copy Member	Function	Description	Action
76A	C76A03105I	COPY	Definition of User Index - AR Voucher Entry	New
76A	C76A0411I	COPY	Country Server - Voucher Entry User Index	New
76A	C76A0413I	COPY	User index for F76A0413	New
76A	C76A0493I	COPY	/COPY for User Index of X76A0493	New
76A	C76A18I	COPY	/COPY for User Index TX76A18	New
76A	C76A19I	COPY	User Index Drivers for F76A19	New
76A	C76A81DRPT	COPY	DREAM Writer - Dynamic Report Processing	New
76A	C76ACUIT	COPY	CUIT Edition Routine	New
76A	C76AMINI	COPY	/COPY for User Index TXMIN	New
76A	C76ATAXI	COPY	Store Tax Calculation Information	New
76A	E76A81DRPT	COPY	DREAM Writer - Dynamic Report Processing	New
76A	E76ACUIT	COPY	Edition CUIT Arrays	New
76B	C76B11	COPY	Right Justify Short Item Number	New
76B	E76B11	COPY	Right Justify Short Item Number	New

Server Changes

Rptg Code	Member ID	Function	Description	Action
70	X700400	RPG	Tax Withholding Constants server	New
70	X700401	RPG	A/B Tax Withholding Codes Server	New
70	X700402	RPG	Tax Code Server	New
70	X700403	RPG	Tax Code Definition Server	New
70	X700404	RPG	Tax Code Concept Server - User Index	New
70	X700405	RPG	Tax Code Precedent Server - User Index	New
70	X700406	RPG	Base Reduction Server - User Index	New

Rptg Code	Member ID	Function	Description	Action
70	X700407	RPG	Tax Withholding Limit Amount Server	New
70	X700408	RPG	Supplier Reduction Server - User Index	New
70	X700412	RPG	Address Book Members - User Index	New
70	X700415	RPG	Payments done to other companies Server - User Index	New
70	X700416	RPG	Accumulated Amount Server - User Index	New
70	X700418	RPG	Address Book Category Code Tag File Server	New
70	X700430	RPG	Base Calculation	New
70	X700431	RPG	Tax Withholding Calculation	New
70	X700435	RPG	Voucher Entry Validation	New
70	X700436	RPG	Delete/Void Tax Voucher	New
70	X700438	RPG	Initialize USRIDX	New
70	X700439	RPG	Country Exit - Tax Processing	New
70	X700440	RPG	User Index Driver for Manual Payment	New
70	X700441	RPG	Manual Pmt w/Voucher Match - Edition of Pmt Group	New
70	X700442	RPG	Manual Pmt w/Voucher Match - Update Files and Accumulated	New
70	X700443	RPG	Manual Pmt w/Voucher Match - Retrieve Company	New
70	X700444	RPG	Manual Pmt w/Voucher Match - Update Tag File of F0414	New
70	X700445	RPG	A/P Manual Check w/o Match - Update Files and Accumulated	New
70	X700446	RPG	A/P Pre payment - Update Files and Accumulated	New
70	X700447	RPG	Withholdings of manual pmt. on Voucher/Suffix & tot.voucher	New
70	X700455	RPG	Control of Selected PCG to Print	New
70	X700456	RPG	A/P Functional Server - Country Server - Reset	New
70	X700457	RPG	Update Payment Document	New

Rptg Code	Member ID	Function	Description	Action
70	X700458	RPG	Check information about Tax withholding	New
70	X7004581AR	RPG	Check information about Tax withholding	New
70	X700459	RPG	Write tax withholding records	New
70	X700460	RPG	Verify if vendor is include in other payment	New
70	X700461	RPG	Get relation company	New
70	X700465	RPG	Delete records in USRIDX (X04570)	New
70	X700466	RPG	Delete records in tag file F700473	New
70	X700467	RPG	Delete records in tag file F700473	New
70	X700468	RPG	Print Certificates	New
70	X700470	RPG	Update tag file F700473	New
70	X700471	RPG	Errors and Warnings	New
70	X700472	RPG	Retrieve information to related voucher	New
70	X700473	RPG	Calculate Withholding	New
70	X700474	RPG	Check amount of payment and write in work file	New
70	X700475	RPG	Errors and Warnings	New
70	X700476	RPG	Reorganize usridx X04570	New
70	X700480	RPG	Void tax Withholding	New
70	X700481	RPG	Check voucher of Tax Withholding	New
70	X700482	RPG	Void payment - Tax Voucher accumulation	New
70	X700491	RPG	A/P Functional Server - Tax Withholding edit and Update	New
70	X700492	RPG	A/P Speed release selection option validation	New
70	X700493	RPG	A/P Speed release -Update related Tax Voucher	New
70	X700494	RPG	Duplicate Tax/Withholding codes from split voucher	New

Rptg Code	Member ID	Function	Description	Action
70	XF700411	RPG	F700411 Functional Server	New
70	XF700414	RPG	F700414 Functional Server	New
70	XF700416	RPG	F700416 Functional Server	New
70	XF700417	RPG	F700417 Functional Server	New
70	XT0413	RPG	Accounts Payable Payment Functional Server	Changed
70	XT700411	RPG	F700411 Functional Server	New
74	X00LCTL	RPG	Country Server Program Designator	Changed
74	X045701NL	RPG	Pre-Payment Processing - User Exit Program	Changed
74	X045704DE	RPG	Accounts Payable Functional Server - Country Server	Changed
74	X04570ANL	RPG	Pre-Payment Processing - User Exit Program	Changed
74	X74030	RPG	Italian Bank ID File Validation Server	Changed
74	X74080	RPG	Suspended VAT - As Of Calculation	Changed
74	XT0311Z3IT	RPG	Accounts Receivable Functional Server - Country Server	Changed
74	XT0311Z3TR	RPG	Turkish G/L Date Edit	Changed
74	XT0411Z2NL	RPG	Accounts Payable Functional Server - Country Server	Changed
74	XT0411Z3IT	RPG	Accounts Payable Functional Server - Country Server	Changed
74	XT0411Z3TR	RPG	Turkish G/L Date Edit	Changed
74	XT0411Z3UK	RPG	A/P Functional Server - Country Server	Changed
74	XT0411ZAIT	RPG	Accounts Payable Functional Server - Country Server	Changed
74	XT0411ZAUK	RPG	Accounts Payable Functional Server - Country Server	Changed
74	XT0411ZCIT	RPG	Accounts Payable Functional Server - Country Server	Changed
74	XT0411ZCUK	RPG	Accounts Payable Functional Server - Country Server	Changed

Rptg Code	Member ID	Function	Description	Action
74	XT0411ZDIT	RPG	Accounts Payable Functional Server - Country Server	Changed
74	XT04135IT	RPG	Accounts Payable Payment Functional Server - Country Server	Changed
74	XT04136IT	RPG	Accounts Payable Payment Functional Server - Country Server	Changed
74	XT0413DUK	RPG	Post - Delete Voucher Receipt Entries - UK Withholding	Changed
74R	X045703RU	RPG	A/P Payment - Pre-Payment Processor-CSE Process group end	Changed
74R	X74R0311	RPG	Obtain current record of cash applied F0311 - RUSSIA	Changed
74R	X74R0411	RPG	Add Records in F0411 tag file	Changed
74R	X74R0415	RPG	Speed Release - Recalc Pay Item Amounts	Changed
74R	X74R0905	RPG	Prepayment processing	Changed
74R	X74R0972	RPG	Correspondence Integrity and Rebuild - Detail Processing	Changed
74R	X74R0973	RPG	Correspondence Integrity and Rebuild - Document Editing	Changed
74R	X74R0981	RPG	Correspondence Initialization	Changed
74R	X74R0982	RPG	Correspondence Detail Processing	Changed
74R	X74R0983	RPG	Correspondence Editing	Changed
74R	X74R0985	RPG	Correspondence Transaction Creation	Changed
74R	X74R0986	RPG	Correspondence Assign JELN to AE records	Changed
74R	X74R0987	RPG	Get Correspondence Method	Changed
74R	X74R0988	RPG	Edit Correspondence Rules	Changed
74R	X74R0989	RPG	Correspondence Intercompany Accounts	Changed
74R	X74R0990	RPG	Correspondence Intercompany Journal Entry Processing	Changed
74R	X76A4115	RPG	Speed Release - Recalculate Pay Item Amounts	New

Rptg Code	Member ID	Function	Description	Action
74R	XT0411ZARU	RPG	Accounts Payable Functional Server - Country Server Add	Changed
74R	XT0411ZCRU	RPG	Accounts Payable Functional Server - Country Server Chg	Changed
74R	XT74R0411	RPG	Functional server F74R0411	Changed
74S	X74SC036	RPG	Remittance Effect Suppression	New
74S	X74SC040	RPG	Profits / Expenses Exchange Rate Calculation and Creation	New
74S	X74SC105	RPG	Remittances in Magnetic Tape Server	New
74S	X74SC311	RPG	Matching Document Access	New
74S	X74SC801	RPG	F0911 Last Line Load	New
74S	X74SF000	RPG	Expiration Date Calc Server	New
74S	X74SF001	RPG	XT0311Z1 Version Validation Routine	New
74S	X74SF502	RPG	Numeric to Word Conversion (New)	New
74S	X74SF503	RPG	Numeric to Word Conversion (EURO))	New
74S	X74SFDCB	RPG	Digit Control Bank Account (Validation Dig. Control)	New
74S	X74SFNIF	RPG	NIF / CIF Validation Routine	New
74S	X74SL275	RPG	A / P invoices Data Load - Same as PGRFL651	New
74S	X74SL276	RPG	A / P invoices Data Load - Same as PGRFL652	New
74S	X74SL495	RPG	General Journal per Category Code	New
74S	X74SL555	RPG	Journal Entry Expenses Transfer Creation	New
74S	X74SL630	RPG	Alternate Payer File Load	New
74S	X74SL902	RPG	General Ledger Date Sensitive Balance Retrieval	New
74S	XT0311Z6ES	RPG	Accounts Receivable Functional Server C.S.	New
74S	XT0411Z7ES	RPG	Accounts Payable Functional Server C.S.	New
74S	XT04131ES	RPG	Accounts Payable Payment Functional Server C.S	New

Rptg Code	Member ID	Function	Description	Action
75	X00500RU	RPG	Convert Numeric Value to Words - MUST SEND SOURCE	Changed
75	X00500RU2	RPG	Convert Numeric Value to Words - MUST SEND SOURCE	Changed
75	X045704JP	RPG	Pre-payment Processing - Bank Charge Calculation	Changed
75	X40750	RPG	Retrieve Unit Price	Changed
75	XT04132BR	RPG	A/P Funct. Server - Country Server - Write tag file F700414	New
75	XT04134JP	RPG	A/P Payment Functional Server - Payment Detail Delete-Japan	Changed
75	XT04137JP	RPG	Accounts Payable Payment Functional Server - Country Server	Changed
75	XT04138JP	RPG	A/P Payment Functional Server - Draft Processing	Changed
75	XT0413AJP	RPG	Accounts Payable Payment Functional Server - Country Server	Changed
76	X76B199	RPG	Generate Sales Ledger from Nota Fiscal	Changed
76	XT0311Z1BR	RPG	A/R Brazilian Tag Maintenance	Changed
76	XT0311Z4BR	RPG	A/R Brazilian Tag Maintenance	Changed
76	XT0311Z5BR	RPG	A/R Brazilian Tag Maintenance	Changed
76	XT0911ZDBR	RPG	Reverse Cardex Entry - Country server	Changed
76A	X045703AR	RPG	A/P Payment - Pre-Payment Processor-CSE Process group end	New
76A	X045705AR	RPG	A/P Payment - Editing	New
76A	X045708AR	RPG	A/P Payment - Pre-Payment Processor	New
76A	X045709AR	RPG	A/P Payment - Verify if vendor is include in other payment	New
76A	X49580A1AR	RPG	Control Print Argentina	New
76A	X49580A2AR	RPG	C.Server - Creation of *DTAARA Argentina	New
76A	X49580A3AR	RPG	C. Server - Recovery values of *DTAARA Argentina	New
76A	X7004301AR	RPG	Base Calculation CS for overrides	New

Rptg Code	Member ID	Function	Description	Action
76A	X7004302AR	RPG	Filter F0411 records from base computing	New
76A	X7004741AR	RPG	Check amount of payment and write in work file	New
76A	X76A00	RPG	Retrieve initial tax setup	New
76A	X76A001	RPG	Next Number Resolution RG/3419	New
76A	X76A002	RPG	Identify withholding record	New
76A	X76A0022	CLP	Credit Invoice Print - CLRPFM WF	New
76A	X76A003	RPG	Update payment user index	New
76A	X76A0051	CLP	Create Physical File	New
76A	X76A0060	RPG	Calculation of Check Digit for Bar Code	New
76A	X76A01	RPG	Next Number Resolution RG/3419	New
76A	X76A0101	RPG	Edit CUIT Number	New
76A	X76A0110	RPG	A/B Store EDI Transaction Data In User Index	New
76A	X76A020C	RPG	Edit cash credit invoice RG/3419	New
76A	X76A020N	RPG	Edit not cash credit invoice	New
76A	X76A0302	RPG	Legal invoice number set up verification	New
76A	X76A0303	RPG	Generate NC for financial discount - ARG	New
76A	X76A03031	RPG	Financial Discount Credit Note (from SOP)	New
76A	X76A0304	RPG	Edit Invoice Emission RG.100	New
76A	X76A0305	RPG	Discount credit note void	New
76A	X76A0306	RPG	Update Invoice RG 100	New
76A	X76A0307	RPG	Update Legal Number	New
76A	X76A0308	RPG	Profit withholdings calculation (only for AR)	New
76A	X76A0309	RPG	Save Legal Number in Memory - ARG	New
76A	X76A0311	RPG	Retrieve AR Invoice from F0311	New

Rptg Code	Member ID	Function	Description	Action
76A	X76A0312	RPG	Get internal number using the legal number	New
76A	X76A0313	RPG	A/R Printed Invoice Control	New
76A	X76A0314	RPG	A/R Update of Invoice Total (F76A09)	New
76A	X76A0315	RPG	Verify tax data on F76A19Z1	New
76A	X76A0316	RPG	Validate tax fields	New
76A	X76A0317	RPG	Generate F76A19 based on F76A19Z1	New
76A	X76A0318	RPG	Validate status change - ARGENTINA	New
76A	X76A0319	RPG	Minimum ERROR, update invoice entry USRSPC	New
76A	X76A0320	RPG	Delete Minimum USRIDX when called from P03110Z	New
76A	X76A0321	RPG	Reverse Credited Draft	New
76A	X76A0322	RPG	Financial discount credit note void	New
76A	X76A0360	RPG	Retrieve voided amount DCTM='RE'	New
76A	X76A0399	RPG	Legal Invoice Number Update (F0311/RPVINV)	New
76A	X76A0400	RPG	Retrieve Withholding Type of Tax Code	New
76A	X76A0403	RPG	Validate supplier in hold status - ARGENTINA	New
76A	X76A0407	RPG	Update voucher entry User Index	New
76A	X76A0409	RPG	Authorization Editing Routine	New
76A	X76A041	RPG	Withholding overrides by PYIN	New
76A	X76A0413	RPG	Depurate F76A30 - ARGENTINA	New
76A	X76A0414	RPG	Verify Existence in AFIP file	New
76A	X76A0415	RPG	Delete Records File From F760472A	New
76A	X76A0416	RPG	Prepayment Verification	New
76A	X76A0418	RPG	Gross and Taxable Amount Recovery	New
76A	X76A042	RPG	Legal Next Numbers Entry	New
76A	X76A0420	RPG	Promissory Note Generation	New

Rptg Code	Member ID	Function	Description	Action
76A	X76A0421	RPG	Void prepayment transaction - Argentina	New
76A	X76A0422	RPG	Edit prepayment status	New
76A	X76A0423	RPG	Retrieve promises due date	New
76A	X76A0424	RPG	Retrieve the greatest due date of withholding payment	New
76A	X76A0429	RPG	Withholding certificate set up editing	New
76A	X76A0430	RPG	Purge of F760411A (Tag File)	New
76A	X76A0435	RPG	Document Duplicity Edition	New
76A	X76A0436	RPG	Retrieve Legal Type Document from F76A30	New
76A	X76A0437	RPG	Edit duplicate legal invoice number for legal type document	New
76A	X76A0440	RPG	Void Receipt Invoice	New
76A	X76A0453	RPG	Retrieve service date	New
76A	X76A0461	RPG	Credit Invoice Void	New
76A	X76A0464	RPG	Void and reclassification of fiscal credit (Credit Invoice)	New
76A	X76A0470	RPG	Calculate and Update Due Date of Check Deferred.	New
76A	X76A0471	RPG	Update differed check - Argentina	New
76A	X76A0472	RPG	A/P Payments - Save Information in User Index of Payments	New
76A	X76A0475	RPG	Reset of F760472A	New
76A	X76A0476	RPG	A/P Payments - F04572 processing	New
76A	X76A0480	RPG	Unchangeable Pay Status verification	New
76A	X76A0480A	RPG	Unchangeable Status Verification for delete	New
76A	X76A0482	RPG	Initial Pay Status Of Voucher Verification	New
76A	X76A0483	RPG	Add Records in F0411 tag file	New
76A	X76A0484	RPG	Edit and Verify A/P Invoice Legal Number	New

Rptg Code	Member ID	Function	Description	Action
76A	X76A0485	RPG	Update Prepayment Negative Item	New
76A	X76A0486	RPG	Valid Prepayment Tax Area	New
76A	X76A0487	RPG	Approver / Treasurer Verification	New
76A	X76A0488	RPG	Verify User Authorization	New
76A	X76A0489	RPG	Filter Records of Limit Date	New
76A	X76A0490	RPG	Edit Pay Status of New Pay Item	New
76A	X76A0491	RPG	Generate Tag File of F0413 for manual payments	New
76A	X76A0492	RPG	Update RG 100 flag	New
76A	X76A0493	RPG	Speed Release - Storage Pay Item Information on User Index	New
76A	X76A0494	RPG	Verify Payment with Check Dif.	New
76A	X76A0496	RPG	Update differed check flag when void the payment	New
76A	X76A081	RPG	Get Gross Income state / agreement supplier set up	New
76A	X76A0901	RPG	Verify document post	New
76A	X76A091	RPG	Write file F76A09 (AR)	New
76A	X76A092	RPG	Write File F76A09 (SOP)	New
76A	X76A101	RPG	Calculate tax control fields	New
76A	X76A12	RPG	Tax Minimum Control	New
76A	X76A13	RPG	Retrieve tax value	New
76A	X76A17	RPG	Retrieve Letter of invoice / shipment note	New
76A	X76A18	RPG	I / O in File F76A18	New
76A	X76A19	RPG	I / O in file F76A19	New
76A	X76A21	RPG	Credit Invoice Header Addition	New
76A	X76A30	RPG	Process F0411Z1 tag file	New
76A	X76A391	RPG	Get last purchase adjusted price	New
76A	X76A392	RPG	Get AS OF PPP	New
76A	X76A393	RPG	Get adjusted prices	New

Rptg Code	Member ID	Function	Description	Action
76A	X76A394	RPG	Calculate Totals	New
76A	X76A395	RPG	Void journal entry	New
76A	X76A396	RPG	Delete adjustments	New
76A	X76A397	RPG	Apply inflation index	New
76A	X76A4008C	RPG	Tax Calculator - Argentina	New
76A	X76A4200	RPG	Edit for Taxable Amount	New
76A	X76A4201	RPG	Save/Restore Tax Values in DTAARA	New
76A	X76A4202	RPG	Edit for Tax Amount	New
76A	X76A4203	RPG	Validate Tax Rates	New
76A	X76A4204	RPG	Clear Minimum User Index	New
76A	X76A4205	RPG	Process Total Tax Rate	New
76A	X76A4260	RPG	Initial setup and validations for invoice print	New
76A	X76A4261	RPG	SOP Invoice Print - Local Update Data Base - ARGENTINA	New
76A	X76A4270	RPG	Hold Control	New
76A	X76A42800	RPG	Sales update - Order Validation	New
76A	X76A428004	RPG	Local Updates	New
76A	X76A4900	RPG	Save/Retrieve Control Values in DTAARA	New
76A	X76A4901	RPG	Get Document Type	New
76A	X76A4960	RPG	Initial setup and validations for invoice print	New
76A	X76A4962	RPG	Bulk Invoice - Line Processing Control	New
76A	X76A50	RPG	Create PCG Edit Credit Invoice	New
76A	X76A500S	RPG	Convert Num to Words - Spanish, Male 2 dec.	New
76A	X76A51	RPG	Edit Credit Invoice Voucher Condition	New
76A	X76A52	RPG	Edit Credit Invoice Status Rules	New
76A	X76A53	RPG	Edit Voucher to be added or updated	New
76A	X76A54	RPG	Deleted Credit Invoice information	New

Rptg Code	Member ID	Function	Description	Action
76A	X76A55	RPG	Edit Voucher w/Credit Invoice to be deleted	New
76A	X76A56	RPG	Changed Voucher/Delete Credit Invoice Info	New
76A	X76A565	RPG	Retrieve amounts and offsets of tax	New
76A	X76A57	RPG	Edit Status of Credit Invoice to voided the pay - ARGENTINA	New
76A	X76A58	RPG	Change Status of Credit Invoice	New
76A	X76A60	RPG	Advance Status of Credit Invoice in Payment	New
76A	X76A61	RPG	Verify if corresponds include impositive reports	New
76A	X76A6201	RPG	Routine of Edition of Changes of State for User	New
76A	X76A6204	RPG	Routine of Upgrade of the File of Login	New
76A	X76A6205	RPG	Compute invoice pending amount with valores	New
76A	X76A6206	RPG	Editing routine	New
76A	X76A6210	RPG	Drafts application amount calculation	New
76A	X76A6214	RPG	Login File Update Routine - Deposit Slip	New
76A	X76A6214H	RPG	Login File Update Routine - Deposit Slip	New
76A	X76A6215	RPG	Recovery of Receipt Number	New
76A	X76A6216	RPG	Determines if a Draft is Posted	New
76A	X76A6217	RPG	Submit receipt printing	New
76A	X76A6218	RPG	Invoice Open Amount Recovery	New
76A	X76A6220	RPG	Total Amount of Receipt A/R	New
76A	X76A6222	RPG	Credit Note/Invoice Open Amount Calculation Routine	New
76A	X76A6223	RPG	Obtain Internal Number Routine	New
76A	X76A6224	RPG	Check if any valor was credited	New
76A	X76A6225	RPG	Check if any journal entry is posted	New

Rptg Code	Member ID	Function	Description	Action
76A	X76A6226	RPG	Routine of authorization validation	New
76A	X76A6240	RPG	Void Invoices that have been applied	New
76A	X76A6241	RPG	Void credited valores	New
76A	X76A6270	RPG	Application and Draft Total Calculation Routine	New
76A	X76A6274	RPG	Draft Receipts Open Amount Calculation Routine	New
76A	X76A6275	RPG	Invoice Open Amount Recovery Excluding Receipt in process	New
76A	X76A6276	RPG	Invoice Open Amount Recovery Routine	New
76A	X76A800	RPG	Generate record into F76A19 after Sales Update	New
76A	X76A8012	RPG	RG 726 /615 supplier type edit/override	New
76A	X76A8013	RPG	RG 726 /615 supplier type override	New
76A	X76A8014	RPG	Voided Voucher Validation	New
76A	X76A8030	RPG	EDI Batch Process	New
76A	X76A8081	RPG	SICORE Files Generation - First Step $G	New
76A	X76A8082	RPG	SICORE Files Generation - First Step $G	New
76A	X76A96CCX	CLP	Cursor Control Function Processor	New
76A	X76A96CCX1	RPG	Cursor Control Function Processor	New
76A	X76A98OPT	RPG	Processing Option Recovery (one by one)	New
76A	X76AA01	RPG	Retrieves Processing Options of P03105AR	New
76A	X76AA02	RPG	A/R Default fields Initialization	New
76A	X76AC01	RPG	Retrieve and Validate PO - P035001 - ARG	New
76A	X76AC02	RPG	Format Legal Number (VINV) - ARG	New
76A	X76AC03	RPG	Print Legal Number - P035001 - ARG	New
76A	X76AD01	RPG	A/R Retrieve PO P0381011AR - ARG	New

Rptg Code	Member ID	Function	Description	Action
76A	X76AD02	RPG	A/R Generate F76A19 records - ARG	New
76A	X76AD03	RPG	A/R Generate F76A09 record - ARG	New
76A	X76AE01	RPG	Update Due Date in Draft	New
76A	X76AF01	RPG	Retrieve P03103AR processing options	New
76A	X76AF02	RPG	Check duplicate receipt number in F0311	New
76A	X76AF03	RPG	Verify if Valores module is installed - ARG	New
76A	X76AF04	RPG	A/R Update RPVR01 field in F0311	New
76A	X76AF05	RPG	Assign Receipt Number	New
76A	X76AG01	RPG	Retrieve Processing Options - P03550AR - ARG	New
76A	X76AG02	RPG	Check Legal Number for NC (for discount) - ARG	New
76A	X76AG03	RPG	Read and Update F0312 - ARG	New
76A	X76AG04	RPG	Process Receive Generated - ARG	New
76A	X76AG05	RPG	Save Batch Number - ARG	New
76A	X76AG06	RPG	Save Batch Key for final Update - ARG	New
76A	X76AG07	RPG	Update Values - ARG	New
76A	X76AG08	RPG	Print Batch Generated - ARG	New
76A	X76AG09	RPG	Init Control - ARG	New
76A	X76AG10	RPG	Unifies Records RC and RG - ARG	New
76A	X76AG11	RPG	Generate R1 in F0311 - ARG	New
76A	X76AG12	RPG	Process Unapplied Cash - ARG	New
76A	X76AH01	RPG	Retrieve Processing Options - P034201	New
76A	X76AH02	RPG	Driver User Index - P034201	New
76A	X76AI01	RPG	Retrieving and Editing of Valores Processing Options	New
76A	X76AI02	RPG	Verify if application is open in F76A62122	New
76A	X76AI03	RPG	Customer Ledger Inquiry - Calculation Draft Amounts	New

Rptg Code	Member ID	Function	Description	Action
76A	X76AI05	RPG	Additional data selection for Valores	New
76A	X76AI06	RPG	Manage Customer User Index for Draft	New
76A	XF76A20	RPG	File Server F76A20 (Invoice Credit)	New
76A	XF76A50	RPG	Input/Output Server - F76A50	New
76A	XS76A4201	RPG	File Server - F4201 Sales Order Header - ARGENTINA	New
76A	XS76A4211	RPG	File Server - F4211 Sales Order Detail - ARGENTINA	New
76A	XS76A49211	RPG	File Server - F49211 Sales Order Detail - Tag File - ARG	New
76A	XS76A49584	RPG	File Server - F49584 Document Print Control-Document Detail	New
76A	XT0101ZAAR	RPG	A/B Functional Server - Country Server Add - ARGENTINA	New
76A	XT0101ZDAR	RPG	A/B Functional Server - Country Server Delete - ARGENTINA	New
76A	XT0311Z0AR	RPG	A/R FS CS - Tax Calculation	New
76A	XT0311Z8AR	RPG	A/R FS CS - End of transaction processing	New
76A	XT0311Z9AR	RPG	A/R FS CS - Invoice footer Localization edition	New
76A	XT0311ZAAR	RPG	A/R FS CS - Localization Files Update	New
76A	XT0311ZCAR	RPG	A/R FS CS - Localization Files Update	New
76A	XT0311ZDAR	RPG	A/R FS CS - Localization Files Delete	New
76A	XT0411Z0AR	RPG	Edit Voucher - Voucher level	New
76A	XT0411Z2AR	RPG	A/P Functional Server - Edit Line - ARGENTINA	New
76A	XT0411Z3AR	RPG	CSE - Voucher Level Validation - Change action	New
76A	XT0411Z5AR	RPG	Delete to file F760411A Tag file F0411	New
76A	XT0411Z6AR	RPG	Edit Line Level for Delete Action	New
76A	XT0411Z8AR	RPG	Edit Invoice Legal Number	New
76A	XT0411ZAAR	RPG	CSE - Add Action - Argentina	New

Rptg Code	Member ID	Function	Description	Action
76A	XT0411ZCAR	RPG	CSE - Change Action - Argentina	New
76A	XT0411ZDAR	RPG	CSE - Delete Action - Argentina	New
76A	XT04132AR	RPG	A/P - Country Server - Write tag file F700414 - Argentina	New
76A	XT04134AR	RPG	A/P - Country Server - Void Payment - Argentina	New
76A	XT04135AR	RPG	A/P Funct. Server - Country Server - Updated file F700417	New
76A	XT04136AR	RPG	Country Server XT0413 - Edit - ARGENTINA	New
76A	XT04138AR	RPG	Country server XT0413 - Update deferred check - Argentina	New
76A	XT04139AR	RPG	A/P Funct. Server - Country Server - Check Tax withholding	New
76A	XT0413AAR	RPG	A/P - Country Server - Write tag file F0413	New
76A	XT0413DAR	RPG	Country server XT0413 - Delete - ARGENTINA	New
76A	XT700411AR	RPG	A/P Functional Server - Call Functional Server F760411A	New
76A	XT76A0413	RPG	F76A0413 - File server	New
76A	XT76A411	RPG	F76A0411 Functional Server	New
76B	X045708BR	RPG	A/P Payment - Pre-Payment Processor	New
76B	X045709BR	RPG	A/P Payment - Verify if vendor is include in other payment	New
76B	X7601B	RPG	Recalculate Nota Fiscal Header	Changed
76B	X7602B	RPG	Impressao Termos de Abertura e Encerramento.	Changed
76B	X7615B	RPG	Server - Calculate Transaction Nature	Changed
76B	X76291B	RPG	Perform GL entries for ICMS Freight Tax	Changed
76B	X76B04	RPG	Edition Routine	New
76B	X76B094	RPG	Subroutine of Input/Output Registration	New

Rptg Code	Member ID	Function	Description	Action
76B	X76B100	RPG	Verify if CFOP should be 3 digits or 4 digits	New
76B	X76B23	RPG	Retrieve Balance for Journal	New
76B	X76B29	RPG	Accumulate Accounting Amount	New
76B	X76B293	RPG	Generate Accounting Amount - Operation Nature	New
76B	X76B4001	RPG	General Tax Calculator - Brazil	New
76B	X76B4005	RPG	Calculate tax control fields - Brazil	New
76B	X76B700	RPG	Routine for Address Book Registry	New
76B	X76B720	RPG	Routine for Account Plan File	New
76B	X76B740	RPG	Routine for Business Unit/Expenditure File	New
76B	X76B760	RPG	Routine for table Transaction Nature	New
76B	X76B780	RPG	Routine for Merchandise/Services table	New
76B	X76B790	RPG	Routine for Profit and Discounts File	New
76B	X76B96CCX	CLP	Cursor Control Function Processor	New
76B	X76B96CCX1	RPG	Cursor Control Function Processor	New
76B	X76BTX1	RPG	Sales Order Tax Calculator for Brazil	Changed
76B	X76TAXB	RPG	Purchase Order Tax Calculator for Brazil	Changed
76B	XT0411Z3BR	RPG	CSE - Voucher Level Validation - Change action	New
76B	XT0411Z6BR	RPG	CSE - Voucher Level Validation - Delete action	New
76B	XT0411ZABR	RPG	CSE - Add Action	New
76B	XT0411ZCBR	RPG	CSE - Change Action	New
76B	XT0411ZDBR	RPG	CSE - Delete Action	New
76B	XT04134BR	RPG	A/P Funct. Server - Country Server - Void Payment	New
76B	XT04135BR	RPG	A/P Funct. Server - Country Server - Updated file F700417	New

Rptg Code	Member ID	Function	Description	Action
76B	XT04139BR	RPG	A/P Funct. Server - Country Server - Check Tax withholding	New

Localization Changes

This chapter describes changes made to Localization systems.

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

Performance Considerations for Programmers

This section provides information about the 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 the software against thousands or millions as many records as JD Edwards World have in 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.

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 you 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. 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. 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.
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.

Subroutine Name	Explanation
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.

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 succeeds) than to CHAIN with the new key first (usually the CHAIN fails).

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 does not open 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 is recommended 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 works fine. Customers who set up more than 100 types may get no benefit from a cache with 100 entries. If you 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 fails.

```
 * 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. 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'
            ADD         ARR,#A        TOT
ARR,#A      MULT        PCT           RATE
ARR,#A      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
C      A     Add          B
```

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*     PROGRAM INPUT SPECIFICATIONS AND DATA STRUCTURES
T*
+
I*
IDRDS DS
I                         1     4  DRSY
I                         5     6  DRRT
I                         7     16  DRKY
I*
C     DRDS    LOKUPKEY,#E                  66
C     *IN6 6  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 you use a multiple-occurrence data structure 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 is 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 CSR CSR CSR          MOVE  F@AD  #A
                         DO    #A
                         MULT  10    #@AD
                         END
```

Performance Considerations for Programmers

It is more efficient to do the following:

```
CSR          MOVE F@AD                      #A
C            SELEC
C     #A     WHEQ  1
CSR          Z-ADD10                        #@AD
C     #A     WHEQ  2
CSR          Z-ADD10 0                      #@AD
C     #A     WHEQ  3
C     #A     WHEQ  9
CSR          Z-ADD10 0 0 0 0 0 0 0 0#@AD
C            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 DEF
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 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 is 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 is 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)
```

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.